

Rebar Addon for FreeCAD

Generated by Doxygen 1.8.11

Contents

1	Rebar Addon for FreeCAD	1
2	Namespace Index	3
2.1	Namespace List	3
3	Hierarchical Index	5
3.1	Class Hierarchy	5
4	Class Index	7
4.1	Class List	7
5	File Index	9
5.1	File List	9
6	Namespace Documentation	11
6.1	BentShapeRebar Namespace Reference	11
6.1.1	Function Documentation	11
6.1.1.1	CommandBentShapeRebar()	11
6.1.1.2	editBentShapeRebar(Rebar, f_cover, b_cover, l_cover, r_cover, diameter, t_cover, bentLength, bentAngle, rounding, amount_spacing_check, amount_spacing← value, orientation, structure=None, facename=None)	12
6.1.1.3	editDialog(vobj)	14
6.1.1.4	getpointsOfBentShapeRebar(FacePRM, l_cover, r_cover, b_cover, t_cover, bentLength, bentAngle, orientation)	14
6.1.1.5	makeBentShapeRebar(f_cover, b_cover, l_cover, r_cover, diameter, t_cover, bentLength, bentAngle, rounding, amount_spacing_check, amount_spacing← value, orientation=""Bottom Left"", structure=None, facename=None)	16
6.1.2	Variable Documentation	18

6.1.2.1	__author__	18
6.1.2.2	__title__	18
6.1.2.3	__url__	18
6.2	HelicalRebar Namespace Reference	18
6.2.1	Function Documentation	19
6.2.1.1	CommandHelicalRebar()	19
6.2.1.2	createHelicalWire(FacePRM, s_cover, b_cover, t_cover, pitch, size, direction, helix=None)	19
6.2.1.3	editDialog(vobj)	20
6.2.1.4	editHelicalRebar(Rebar, s_cover, b_cover, diameter, t_cover, pitch, structure=None, facename=None)	21
6.2.1.5	getpointsOfHelicalRebar(FacePRM, s_cover, b_cover, t_cover, pitch, edges, diameter, size, direction)	22
6.2.1.6	makeHelicalRebar(s_cover, b_cover, diameter, t_cover, pitch, structure=None, facename=None)	23
6.2.2	Variable Documentation	24
6.2.2.1	__author__	24
6.2.2.2	__title__	24
6.2.2.3	__url__	24
6.3	LShapeRebar Namespace Reference	24
6.3.1	Function Documentation	25
6.3.1.1	CommandLShapeRebar()	25
6.3.1.2	editDialog(vobj)	25
6.3.1.3	editLShapeRebar(Rebar, f_cover, b_cover, l_cover, r_cover, diameter, t_cover, rounding, amount_spacing_check, amount_spacing_value, orientation, structure=None, facename=None)	26
6.3.1.4	getpointsOfLShapeRebar(FacePRM, l_cover, r_cover, b_cover, t_cover, orientation)	27
6.3.1.5	makeLShapeRebar(f_cover, b_cover, l_cover, r_cover, diameter, t_cover, rounding, amount_spacing_check, amount_spacing_value, orientation=""Bottom Left"", structure=None, facename=None)	28
6.3.2	Variable Documentation	29
6.3.2.1	__author__	29
6.3.2.2	__title__	30
6.3.2.3	__url__	30

6.4	PopUpImage Namespace Reference	30
6.4.1	Function Documentation	30
6.4.1.1	showPopUpImageDialog(img)	30
6.4.2	Variable Documentation	30
6.4.2.1	__author__	30
6.4.2.2	__title__	31
6.4.2.3	__url__	31
6.5	RebarDistribution Namespace Reference	31
6.5.1	Function Documentation	31
6.5.1.1	getCustomSpacingString(amount1, spacing1, amount2, spacing2, amount3, spacing3, frontCover, size)	31
6.5.1.2	gettupleOfCustomSpacing(span_string)	32
6.5.1.3	removeRebarDistribution(self)	32
6.5.1.4	runRebarDistribution(self)	32
6.5.2	Variable Documentation	33
6.5.2.1	__author__	33
6.5.2.2	__title__	33
6.5.2.3	__url__	33
6.5.2.4	CustomSpacing	33
6.6	Rebarfunc Namespace Reference	33
6.6.1	Function Documentation	34
6.6.1.1	check_selected_face()	34
6.6.1.2	checkRectangle(edges)	35
6.6.1.3	extendedTangentLength(rounding, diameter, angle)	35
6.6.1.4	extendedTangentPartLength(rounding, diameter, angle)	36
6.6.1.5	facenormalDirection(structure=None, facename=None)	36
6.6.1.6	getBaseObject(obj)	37
6.6.1.7	getBaseStructuralObject(obj)	38
6.6.1.8	getEdgesAngle(edge1, edge2)	38
6.6.1.9	getFaceNumber(s)	39
6.6.1.10	getParametersOfFace(structure, facename, sketch=True)	39

6.6.1.11	getSelectedFace(self)	41
6.6.1.12	getTrueParametersOfStructure(obj)	42
6.6.1.13	showWarning(message)	43
6.6.1.14	translate(context, text, disambig=None)	44
6.6.2	Variable Documentation	44
6.6.2.1	__author__	44
6.6.2.2	__title__	44
6.6.2.3	__url__	44
6.6.2.4	FaceName	44
6.6.2.5	SelectedObj	45
6.7	RebarTools Namespace Reference	45
6.7.1	Variable Documentation	45
6.7.1.1	__author__	45
6.7.1.2	__title__	45
6.7.1.3	__url__	45
6.7.1.4	RebarCommands	45
6.8	Stirrup Namespace Reference	46
6.8.1	Function Documentation	46
6.8.1.1	CommandStirrup()	46
6.8.1.2	editDialog(vobj)	47
6.8.1.3	editStirrup(Rebar, l_cover, r_cover, t_cover, b_cover, f_cover, bentAngle, bentFactor, diameter, rounding, amount_spacing_check, amount_spacing_value, structure=None, facename=None)	47
6.8.1.4	getpointsOfStirrup(FacePRM, l_cover, r_cover, t_cover, b_cover, bentAngle, bentFactor, diameter, rounding, facenormal)	48
6.8.1.5	makeStirrup(l_cover, r_cover, t_cover, b_cover, f_cover, bentAngle, bentFactor, diameter, rounding, amount_spacing_check, amount_spacing_value, structure=None, facename=None)	50
6.8.2	Variable Documentation	51
6.8.2.1	__author__	51
6.8.2.2	__title__	51
6.8.2.3	__url__	52
6.9	StraightRebar Namespace Reference	52

6.9.1 Function Documentation	52
6.9.1.1 CommandStraightRebar()	52
6.9.1.2 editDialog(vobj)	53
6.9.1.3 editStraightRebar(Rebar, f_cover, coverAlong, rt_cover, lb_cover, diameter, amount_spacing_check, amount_spacing_value, orientation, structure=None, facename=None)	53
6.9.1.4 getpointsOfStraightRebar(FacePRM, rt_cover, lb_cover, coverAlong, orientation)	54
6.9.1.5 makeStraightRebar(f_cover, coverAlong, rt_cover, lb_cover, diameter, amount_spacing_check, amount_spacing_value, orientation=""Horizontal"", structure=None, facename=None)	55
6.9.2 Variable Documentation	57
6.9.2.1 __author__	57
6.9.2.2 __title__	57
6.9.2.3 __url__	57
6.10 UShapeRebar Namespace Reference	57
6.10.1 Function Documentation	58
6.10.1.1 CommandUShapeRebar()	58
6.10.1.2 editDialog(vobj)	58
6.10.1.3 editUShapeRebar(Rebar, f_cover, b_cover, r_cover, l_cover, diameter, t_cover, rounding, amount_spacing_check, amount_spacing_value, orientation, structure=None, facename=None)	59
6.10.1.4 getpointsOfUShapeRebar(FacePRM, r_cover, l_cover, b_cover, t_cover, orientation)	60
6.10.1.5 makeUShapeRebar(f_cover, b_cover, r_cover, l_cover, diameter, t_cover, rounding, amount_spacing_check, amount_spacing_value, orientation=""Bottom"", structure=None, facename=None)	61
6.10.2 Variable Documentation	62
6.10.2.1 __author__	62
6.10.2.2 __title__	62
6.10.2.3 __url__	62

7 Class Documentation	63
7.1 BentShapeRebar._BentShapeRebarTaskPanel Class Reference	63
7.1.1 Detailed Description	64
7.1.2 Constructor & Destructor Documentation	64
7.1.2.1 <code>__init__(self, Rebar=None)</code>	64
7.1.3 Member Function Documentation	64
7.1.3.1 <code>accept(self, signal=None)</code>	64
7.1.3.2 <code>amount_radio_clicked(self)</code>	66
7.1.3.3 <code>clicked(self, button)</code>	66
7.1.3.4 <code>getOrientation(self)</code>	67
7.1.3.5 <code>getStandardButtons(self)</code>	67
7.1.3.6 <code>spacing_radio_clicked(self)</code>	67
7.1.4 Member Data Documentation	67
7.1.4.1 <code>FaceName</code>	67
7.1.4.2 <code>form</code>	68
7.1.4.3 <code>Rebar</code>	68
7.1.4.4 <code>SelectedObj</code>	68
7.2 HelicalRebar._HelicalRebarTaskPanel Class Reference	68
7.2.1 Detailed Description	69
7.2.2 Constructor & Destructor Documentation	69
7.2.2.1 <code>__init__(self, Rebar=None)</code>	69
7.2.3 Member Function Documentation	69
7.2.3.1 <code>accept(self, signal=None)</code>	69
7.2.3.2 <code>clicked(self, button)</code>	70
7.2.3.3 <code>getSelectedFace(self)</code>	71
7.2.3.4 <code>getStandardButtons(self)</code>	71
7.2.4 Member Data Documentation	71
7.2.4.1 <code>FaceName</code>	71
7.2.4.2 <code>form</code>	72
7.2.4.3 <code>Rebar</code>	72

7.2.4.4	SelectedObj	72
7.3	LShapeRebar._LShapeRebarTaskPanel Class Reference	72
7.3.1	Detailed Description	73
7.3.2	Constructor & Destructor Documentation	73
7.3.2.1	__init__(self, Rebar=None)	73
7.3.3	Member Function Documentation	73
7.3.3.1	accept(self, signal=None)	73
7.3.3.2	amount_radio_clicked(self)	75
7.3.3.3	clicked(self, button)	75
7.3.3.4	getOrientation(self)	76
7.3.3.5	getStandardButtons(self)	76
7.3.3.6	spacing_radio_clicked(self)	76
7.3.4	Member Data Documentation	77
7.3.4.1	CustomSpacing	77
7.3.4.2	FaceName	77
7.3.4.3	form	77
7.3.4.4	Rebar	77
7.3.4.5	SelectedObj	77
7.4	RebarDistribution._RebarDistributionDialog Class Reference	77
7.4.1	Detailed Description	78
7.4.2	Constructor & Destructor Documentation	78
7.4.2.1	__init__(self, frontCover, size)	78
7.4.3	Member Function Documentation	78
7.4.3.1	accept(self)	78
7.4.3.2	setupUi(self)	79
7.4.4	Member Data Documentation	79
7.4.4.1	CustomSpacing	79
7.4.4.2	ExpandingLength	80
7.4.4.3	form	80
7.4.4.4	FrontCover	80

7.5	Stirrup._StirrupTaskPanel Class Reference	80
7.5.1	Detailed Description	81
7.5.2	Constructor & Destructor Documentation	81
7.5.2.1	<code>__init__(self, Rebar=None)</code>	81
7.5.3	Member Function Documentation	81
7.5.3.1	<code>accept(self, signal=None)</code>	81
7.5.3.2	<code>amount_radio_clicked(self)</code>	83
7.5.3.3	<code>clicked(self, button)</code>	83
7.5.3.4	<code>getStandardButtons(self)</code>	84
7.5.3.5	<code>spacing_radio_clicked(self)</code>	84
7.5.4	Member Data Documentation	84
7.5.4.1	<code>CustomSpacing</code>	84
7.5.4.2	<code>FaceName</code>	84
7.5.4.3	<code>form</code>	84
7.5.4.4	<code>Rebar</code>	84
7.5.4.5	<code>SelectedObj</code>	85
7.6	StraightRebar._StraightRebarTaskPanel Class Reference	85
7.6.1	Detailed Description	86
7.6.2	Constructor & Destructor Documentation	86
7.6.2.1	<code>__init__(self, Rebar=None)</code>	86
7.6.3	Member Function Documentation	86
7.6.3.1	<code>accept(self, signal=None)</code>	86
7.6.3.2	<code>amount_radio_clicked(self)</code>	87
7.6.3.3	<code>changeCoverAlong(self)</code>	88
7.6.3.4	<code>changeOrientation(self)</code>	88
7.6.3.5	<code>clicked(self, button)</code>	88
7.6.3.6	<code>getStandardButtons(self)</code>	89
7.6.3.7	<code>spacing_radio_clicked(self)</code>	89
7.6.4	Member Data Documentation	89
7.6.4.1	<code>CustomSpacing</code>	89

7.6.4.2	FaceName	89
7.6.4.3	form	89
7.6.4.4	Rebar	89
7.6.4.5	SelectedObj	89
7.7	UShapeRebar._UShapeRebarTaskPanel Class Reference	90
7.7.1	Detailed Description	90
7.7.2	Constructor & Destructor Documentation	91
7.7.2.1	__init__(self, Rebar=None)	91
7.7.3	Member Function Documentation	91
7.7.3.1	accept(self, signal=None)	91
7.7.3.2	amount_radio_clicked(self)	92
7.7.3.3	clicked(self, button)	92
7.7.3.4	getOrientation(self)	93
7.7.3.5	getStandardButtons(self)	93
7.7.3.6	spacing_radio_clicked(self)	94
7.7.4	Member Data Documentation	94
7.7.4.1	CustomSpacing	94
7.7.4.2	FaceName	94
7.7.4.3	form	94
7.7.4.4	Rebar	94
7.7.4.5	SelectedObj	94
7.8	RebarTools.BentShapeRebarTool Class Reference	94
7.8.1	Detailed Description	95
7.8.2	Member Function Documentation	95
7.8.2.1	Activated(self)	95
7.8.2.2	GetResources(self)	95
7.8.2.3	IsActive(self)	95
7.9	RebarTools.HelicalRebarTool Class Reference	96
7.9.1	Detailed Description	96
7.9.2	Member Function Documentation	96

7.9.2.1	Activated(self)	96
7.9.2.2	GetResources(self)	97
7.9.2.3	IsActive(self)	97
7.10	RebarTools.LShapeRebarTool Class Reference	97
7.10.1	Detailed Description	98
7.10.2	Member Function Documentation	98
7.10.2.1	Activated(self)	98
7.10.2.2	GetResources(self)	98
7.10.2.3	IsActive(self)	98
7.11	PopUpImage.PopUpImage Class Reference	99
7.11.1	Detailed Description	100
7.11.2	Constructor & Destructor Documentation	100
7.11.2.1	__init__(self, img)	100
7.11.3	Member Data Documentation	100
7.11.3.1	image	100
7.11.3.2	verticalLayout	100
7.12	RebarTools.StirrupTool Class Reference	100
7.12.1	Detailed Description	101
7.12.2	Member Function Documentation	101
7.12.2.1	Activated(self)	101
7.12.2.2	GetResources(self)	101
7.12.2.3	IsActive(self)	101
7.13	RebarTools.StraightRebarTool Class Reference	102
7.13.1	Detailed Description	102
7.13.2	Member Function Documentation	102
7.13.2.1	Activated(self)	102
7.13.2.2	GetResources(self)	103
7.13.2.3	IsActive(self)	103
7.14	RebarTools.UShapeRebarTool Class Reference	103
7.14.1	Detailed Description	104
7.14.2	Member Function Documentation	104
7.14.2.1	Activated(self)	104
7.14.2.2	GetResources(self)	104
7.14.2.3	IsActive(self)	104

8 File Documentation	105
8.1 BentShapeRebar.py File Reference	105
8.2 BentShapeRebar.py	106
8.3 HelicalRebar.py File Reference	110
8.4 HelicalRebar.py	111
8.5 LShapeRebar.py File Reference	114
8.6 LShapeRebar.py	115
8.7 PopUpImage.py File Reference	119
8.8 PopUpImage.py	119
8.9 README.md File Reference	120
8.10 README.md	120
8.11 RebarDistribution.py File Reference	120
8.12 RebarDistribution.py	121
8.13 Rebarfunc.py File Reference	122
8.14 Rebarfunc.py	123
8.15 RebarTools.py File Reference	127
8.16 RebarTools.py	127
8.17 Stirrup.py File Reference	129
8.18 Stirrup.py	130
8.19 StraightRebar.py File Reference	134
8.20 StraightRebar.py	135
8.21 UShapeRebar.py File Reference	139
8.22 UShapeRebar.py	140

Chapter 1

Rebar Addon for FreeCAD

Started as a Google Summer of Code (GSoC 2017) [project](#).

Documentation

This project is aimed at easing up the process of rebaring in [FreeCAD](#). In this project, list of rebars will be provided to user under Rebar tools in the form of dropdown. This project covers six different rebar shapes as given below:

- **Straight Rebar:** [wiki](#)
- **UShape Rebar:** [wiki](#)
- **LShape Rebar:** [wiki](#)
- **BentShpae Rebar:** [wiki](#)
- **Stirrup Rebar:** [wiki](#)
- **Helical Rebar:** [wiki](#)

Video Tutorial

Installation

Pre-requisites

- FreeCAD (version >= 0.17): [Installation guide](#)

Steps to install Rebar Addon in FreeCAD

1. Open the FreeCAD Addon Manager (Tool -> Addon manager).
2. When an addon manager will open, select Reinforcement from a list of workbenches shown by an addon manager.
3. After selecting, click on Install/Update button.
4. Restart FreeCAD.
5. Now you will see different rebars in a drop-down list of rebar tools (Arch -> Rebar tools -> Different rebars).

How it works

Each rebar tool has two files, one is Python file and second is there respective name UI file like `Straight↔Rebar.py` and `StraightRebar.ui` file). Let's take a straight rebar tool. In `StraightRebar.py` file, there are two functions. One is `makeStraightRebar()` function. This function creates straight rebar and adds new properties to the default Rebar object. Second function is `editStraightRebar`. This function is used when we want to change a new properties(which is created by `makeStraightRebar` function) of the rebar object and it will take Rebar object as input which is created by `makeStraightRebar` function. In `StraightRebar.py`, `_StraightRebarTaskPanel` class is present. This class loads UI(present in `StraightRebar.ui` file) in the task panel of FreeCAD. First time when a user clicks on Apply or Ok button, then `makeStraight↔Rebar` function is executed and after that when user want to change the properties of Straight rebar then `edit↔StraightRebar` function is excuted.

Extras

- [FreeCAD forum thread](#)
- [GSoC proposal](#)
- [Development logs](#)

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

BentShapeRebar	11
HelicalRebar	18
LShapeRebar	24
PopUpImage	30
RebarDistribution	31
Rebarfunc	33
RebarTools	45
Stirrup	46
StraightRebar	52
UShapeRebar	57

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

BentShapeRebar._BentShapeRebarTaskPanel	63
HelicalRebar._HelicalRebarTaskPanel	68
LShapeRebar._LShapeRebarTaskPanel	72
RebarDistribution._RebarDistributionDialog	77
Stirrup._StirrupTaskPanel	80
StraightRebar._StraightRebarTaskPanel	85
UShapeRebar._UShapeRebarTaskPanel	90
RebarTools.BentShapeRebarTool	94
RebarTools.HelicalRebarTool	96
RebarTools.LShapeRebarTool	97
QDialog	
PopUpImage.PopUpImage	99
RebarTools.StirrupTool	100
RebarTools.StraightRebarTool	102
RebarTools.UShapeRebarTool	103

Chapter 4

Class Index

4.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

BentShapeRebar._BentShapeRebarTaskPanel	63
HelicalRebar._HelicalRebarTaskPanel	68
LShapeRebar._LShapeRebarTaskPanel	72
RebarDistribution._RebarDistributionDialog	77
Stirrup._StirrupTaskPanel	80
StraightRebar._StraightRebarTaskPanel	85
UShapeRebar._UShapeRebarTaskPanel	90
RebarTools.BentShapeRebarTool	94
RebarTools.HelicalRebarTool	96
RebarTools.LShapeRebarTool	97
PopUpImage.PopUpImage	99
RebarTools.StirrupTool	100
RebarTools.StraightRebarTool	102
RebarTools.UShapeRebarTool	103

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

BentShapeRebar.py	105
HelicalRebar.py	110
LShapeRebar.py	114
PopUpImage.py	119
RebarDistribution.py	120
Rebarfunc.py	122
RebarTools.py	127
Stirrup.py	129
StraightRebar.py	134
UShapeRebar.py	139

Chapter 6

Namespace Documentation

6.1 BentShapeRebar Namespace Reference

Classes

- class [_BentShapeRebarTaskPanel](#)

Functions

- def [getpointsOfBentShapeRebar](#) (FacePRM, l_cover, r_cover, b_cover, t_cover, bentLength, bentAngle, orientation)
- def [makeBentShapeRebar](#) (f_cover, b_cover, l_cover, r_cover, diameter, t_cover, bentLength, bentAngle, rounding, amount_spacing_check, amount_spacing_value, orientation="Bottom Left", structure=None, face-name=None)
- def [editBentShapeRebar](#) (Rebar, f_cover, b_cover, l_cover, r_cover, diameter, t_cover, bentLength, bentAngle, rounding, amount_spacing_check, amount_spacing_value, orientation, structure=None, face-name=None)
- def [editDialog](#) (vobj)
- def [CommandBentShapeRebar](#) ()

Variables

- string [__title__](#) = "BentShapeRebar"
- string [__author__](#) = "Amritpal Singh"
- string [__url__](#) = "https://www.freecadweb.org"

6.1.1 Function Documentation

6.1.1.1 def BentShapeRebar.CommandBentShapeRebar ()

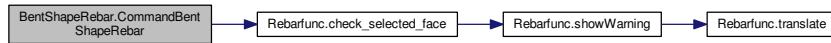
Definition at line 359 of file [BentShapeRebar.py](#).

```

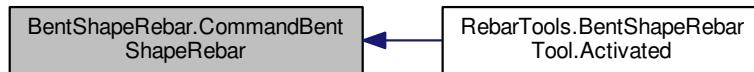
00359 def CommandBentShapeRebar():
00360     selected_obj = check_selected_face()
00361     if selected_obj:
00362         FreeCADGui.Control.showDialog(_BentShapeRebarTaskPanel())
00363

```

Here is the call graph for this function:



Here is the caller graph for this function:



6.1.1.2 def BentShapeRebar.editBentShapeRebar (Rebar, f_cover, b_cover, l_cover, r_cover, diameter, t_cover, bentLength, bentAngle, rounding, amount_spacing_check, amount_spacing_value, orientation, structure = None, facename = None)

Definition at line 270 of file [BentShapeRebar.py](#).

```

00270 def editBentShapeRebar(Rebar, f_cover, b_cover, l_cover, r_cover, diameter, t_cover,
                           bentLength, bentAngle, rounding, amount_spacing_check, amount_spacing_value, orientation, structure = None,
                           facename = None):
00271     sketch = Rebar.Base
00272     if structure and facename:
00273         sketch.Support = [(structure, facename)]
00274     # Check if sketch support is empty.
00275     if not sketch.Support:
00276         showWarning("You have checked remove external geometry of base sketchs when needed.\nTo
unchecked Edit->Preferences->Arch.")
00277         return
00278     # Assigned values
00279     facename = sketch.Support[0][1][0]
00280     structure = sketch.Support[0][0]
00281     face = structure.Shape.Faces[getFaceNumber(facename) - 1]
00282     #StructurePRM = getTrueParametersOfStructure(structure)
00283     # Get parameters of the face where sketch of rebar is drawn
00284     FacePRM = getParametersOfFace(structure, facename)
00285     # Get points of L-Shape rebar
00286     points = getpointsOfBentShapeRebar(FacePRM, l_cover, r_cover, b_cover, t_cover
, bentLength, bentAngle, orientation)
00287     sketch.movePoint(0, 1, points[0], 0)
00288     FreeCAD.ActiveDocument.recompute()
00289     sketch.movePoint(0, 2, points[1], 0)
00290     FreeCAD.ActiveDocument.recompute()
00291     sketch.movePoint(1, 1, points[1], 0)
00292     FreeCAD.ActiveDocument.recompute()
00293     sketch.movePoint(1, 2, points[2], 0)
00294     FreeCAD.ActiveDocument.recompute()
00295     sketch.movePoint(2, 1, points[2], 0)
00296     FreeCAD.ActiveDocument.recompute()
00297     sketch.movePoint(2, 2, points[3], 0)
00298     FreeCAD.ActiveDocument.recompute()
00299

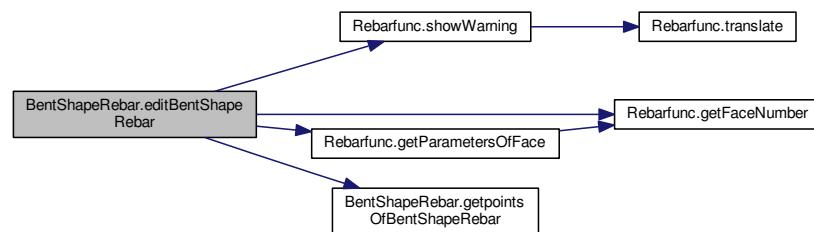
```

```

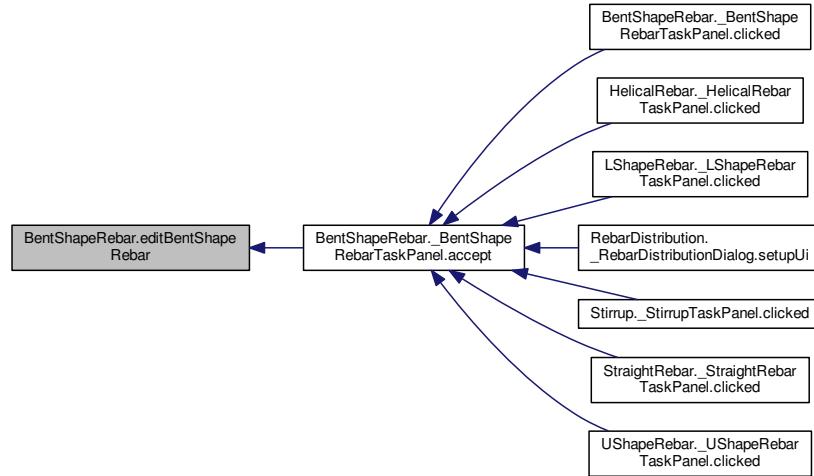
00300     sketch.movePoint(3, 1, points[3], 0)
00301     FreeCAD.ActiveDocument.recompute()
00302     sketch.movePoint(3, 2, points[4], 0)
00303     FreeCAD.ActiveDocument.recompute()
00304
00305     sketch.movePoint(4, 1, points[4], 0)
00306     FreeCAD.ActiveDocument.recompute()
00307     sketch.movePoint(4, 2, points[5], 0)
00308     FreeCAD.ActiveDocument.recompute()
00309
00310     Rebar.OffsetStart = f_cover
00311     Rebar.OffsetEnd = f_cover
00312     if amount_spacing_check:
00313         Rebar.Amount = amount_spacing_value
00314         FreeCAD.ActiveDocument.recompute()
00315         Rebar.AmountCheck = True
00316     else:
00317         size = (ArchCommands.projectToVector(structure.Shape.copy(), face.normalAt(0, 0))).Length
00318         Rebar.Amount = int((size - diameter) / amount_spacing_value)
00319         FreeCAD.ActiveDocument.recompute()
00320     Rebar.AmountCheck = False
00321     Rebar.Diameter = diameter
00322     Rebar.FrontCover = f_cover
00323     Rebar.LeftCover = l_cover
00324     Rebar.RightCover = r_cover
00325     Rebar.BottomCover = b_cover
00326     Rebar.TopCover = t_cover
00327     Rebar.BentLength = bentLength
00328     Rebar.BentAngle = bentAngle
00329     Rebar.Rounding = rounding
00330     Rebar.TrueSpacing = amount_spacing_value
00331     Rebar.Orientation = orientation
00332     FreeCAD.ActiveDocument.recompute()
00333     return Rebar
00334

```

Here is the call graph for this function:



Here is the caller graph for this function:



6.1.1.3 def BentShapeRebar.editDialog (vobj)

Definition at line 335 of file [BentShapeRebar.py](#).

```

00335 def editDialog(vobj):
00336     FreeCADGui.Control.closeDialog()
00337     obj = _BentShapeRebarTaskPanel(vobj.Object)
00338     obj.form.frontCover.setText(str(vobj.Object.FrontCover))
00339     obj.form.l_sideCover.setText(str(vobj.Object.LeftCover))
00340     obj.form.r_sideCover.setText(str(vobj.Object.RightCover))
00341     obj.form.bottomCover.setText(str(vobj.Object.BottomCover))
00342     obj.form.diameter.setText(str(vobj.Object.Diameter))
00343     obj.form.topCover.setText(str(vobj.Object.TopCover))
00344     obj.form.bentLength.setText(str(vobj.Object.BentLength))
00345     obj.form.bentAngle.setValue(vobj.Object.BentAngle)
00346     obj.form.rounding.setValue(vobj.Object.Rounding)
00347     obj.form.orientation.setCurrentIndex(obj.form.orientation.findText(str(vobj.Object.Orientation)))
00348     if vobj.Object.AmountCheck:
00349         obj.form.amount.setValue(vobj.Object.Amount)
00350     else:
00351         obj.form.amount_radio.setChecked(False)
00352         obj.form.spacing_radio.setChecked(True)
00353         obj.form.amount.setDisabled(True)
00354         obj.form.spacing.setEnabled(True)
00355         obj.form.spacing.setText(str(vobj.Object.TrueSpacing))
00356     #obj.form.PickSelectedFace.setVisible(False)
00357     FreeCADGui.Control.showDialog(obj)
00358

```

6.1.1.4 def BentShapeRebar.getpointsOfBentShapeRebar (FacePRM, l_cover, r_cover, b_cover, t_cover, bentLength, bentAngle, orientation)

getpointsOfBentShapeRebar(FacePRM, LeftCover, RightCover, BottomCover, TopCover, BentLength, BentAngle, Orientation)
 Return points of the LShape rebar in the form of array for sketch.
 It takes four different orientations input i.e. 'Bottom', 'Top', 'Left', 'Right'.

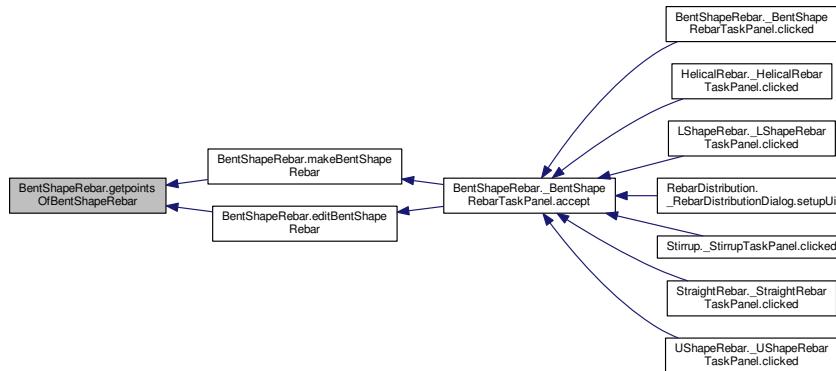
Definition at line 40 of file [BentShapeRebar.py](#).

```

00040 def getpointsOfBentShapeRebar(FacePRM, l_cover, r_cover, b_cover, t_cover,
00041     bentLength, bentAngle, orientation):
00042     """ getpointsOfBentShapeRebar(FacePRM, LeftCover, RightCover, BottomCover, TopCover, BentLength,
00043     BentAngle, Orientation):
00044     Return points of the LShape rebar in the form of array for sketch.
00045     It takes four different orientations input i.e. 'Bottom', 'Top', 'Left', 'Right'.
00046     """
00047     if orientation == "Bottom":
00048         x1 = FacePRM[1][0] - FacePRM[0][0] / 2 + l_cover
00049         y1 = FacePRM[1][1] + FacePRM[0][1] / 2 - t_cover
00050         x2 = x1 + bentLength
00051         y2 = y1
00052         dis = (FacePRM[0][1] - t_cover - b_cover) * math.tan(math.radians(bentAngle - 90))
00053         x3 = x2 + dis
00054         y3 = FacePRM[1][1] - FacePRM[0][1] / 2 + b_cover
00055         x4 = FacePRM[1][0] + FacePRM[0][0] / 2 - r_cover - bentLength - dis
00056         y4 = y3
00057         x5 = x4 + dis
00058         y5 = y2
00059         x6 = x5 + bentLength
00060         y6 = y5
00061     elif orientation == "Top":
00062         x1 = FacePRM[1][0] - FacePRM[0][0] / 2 + l_cover
00063         y1 = FacePRM[1][1] - FacePRM[0][1] / 2 + b_cover
00064         x2 = x1 + bentLength
00065         y2 = y1
00066         dis = (FacePRM[0][1] - t_cover - b_cover) * math.tan(math.radians(bentAngle - 90))
00067         x3 = x2 + dis
00068         y3 = FacePRM[1][1] + FacePRM[0][1] / 2 - t_cover
00069         x4 = FacePRM[1][0] + FacePRM[0][0] / 2 - r_cover - bentLength - dis
00070         y4 = y3
00071         x5 = x4 + dis
00072         y5 = y2
00073         x6 = x5 + bentLength
00074     elif orientation == "Left":
00075         x1 = FacePRM[1][0] + FacePRM[0][0] / 2 - r_cover
00076         y1 = FacePRM[1][1] + FacePRM[0][1] / 2 - t_cover
00077         x2 = x1
00078         y2 = y1 - bentLength
00079         dis = (FacePRM[0][0] - r_cover - l_cover) * math.tan(math.radians(bentAngle - 90))
00080         x3 = FacePRM[1][0] - FacePRM[0][0] / 2 + l_cover
00081         y3 = y2 - dis
00082         x4 = x3
00083         y4 = FacePRM[1][1] - FacePRM[0][1] / 2 + b_cover + bentLength + dis
00084         x5 = x2
00085         y5 = y4 - dis
00086         x6 = x5
00087         y6 = y5 - bentLength
00088     elif orientation == "Right":
00089         x1 = FacePRM[1][0] - FacePRM[0][0] / 2 + l_cover
00090         y1 = FacePRM[1][1] + FacePRM[0][1] / 2 - t_cover
00091         x2 = x1
00092         y2 = y1 - bentLength
00093         dis = (FacePRM[0][0] - r_cover - l_cover) * math.tan(math.radians(bentAngle - 90))
00094         x3 = FacePRM[1][0] + FacePRM[0][0] / 2 - r_cover
00095         y3 = y2 - dis
00096         x4 = x3
00097         y4 = FacePRM[1][1] - FacePRM[0][1] / 2 + b_cover + bentLength + dis
00098         x5 = x2
00099         y5 = y4 - dis
00100         x6 = x5
00101         y6 = y5 - bentLength
00102     return [FreeCAD.Vector(x1, y1, 0), FreeCAD.Vector(x2, y2, 0), \
00103             FreeCAD.Vector(x3, y3, 0), FreeCAD.Vector(x4, y4, 0), \
00104             FreeCAD.Vector(x5, y5, 0), FreeCAD.Vector(x6, y6, 0)]

```

Here is the caller graph for this function:



6.1.1.5 def BentShapeRebar.makeBentShapeRebar (*f_cover, b_cover, l_cover, r_cover, diameter, t_cover, bentLength, bentAngle, rounding, amount_spacing_check, amount_spacing_value, orientation = "Bottom Left", structure =None, facename =None*)

makeBentShapeRebar(FrontCover, BottomCover, LeftCover, RightCover, Diameter, TopCover, BentLength, BentAngle, AmountSpacingCheck, AmountSpacingValue, Orientation, Structure, Facename): Adds the Bent-Shape reinforcement bar to the selected structural object.

It takes four different orientations input i.e. 'Bottom', 'Top', 'Left', 'Right'.

Definition at line 200 of file [BentShapeRebar.py](#).

```

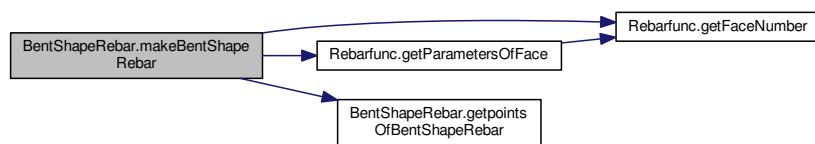
00200 def makeBentShapeRebar(f_cover, b_cover, l_cover, r_cover, diameter, t_cover, bentLength,
                           bentAngle, rounding, amount_spacing_check, amount_spacing_value, orientation = "Bottom Left", structure =
                           None, facename = None):
00201     """ makeBentShapeRebar(FrontCover, BottomCover, LeftCover, RightCover, Diameter, TopCover, BentLength,
                           BentAngle, Rounding,
                           AmountSpacingCheck, AmountSpacingValue, Orientation, Structure, Facename): Adds the Bent-Shape
                           reinforcement bar to the
                           selected structural object.
00204   It takes four different orientations input i.e. 'Bottom', 'Top', 'Left', 'Right'.
00205 """
00206     if not structure and not facename:
00207         selected_obj = FreeCADGui.Selection.getSelectionEx()[0]
00208         structure = selected_obj.Object
00209         facename = selected_obj.SubElementNames[0]
00210     face = structure.Shape.Faces[getFaceNumber(facename) - 1]
00211     #StructurePRM = getTrueParametersOfStructure(structure)
00212     FacePRM = getParametersOfFace(structure, facename)
00213     if not FacePRM:
00214         FreeCAD.Console.PrintError("Cannot identified shape or from which base object sturctural element is
                           derived\n")
00215         return
00216     # Get points of L-Shape rebar
00217     points = getpointsOfBentShapeRebar(FacePRM, l_cover, r_cover, b_cover, t_cover
00218                                         , bentLength, bentAngle, orientation)
00219     import Part
00220     import Arch
00221     sketch = FreeCAD.activeDocument().addObject('Sketcher::SketchObject', 'Sketch')
00222     sketch.MapMode = "FlatFace"
00223     sketch.Support = [(structure, facename)]
00224     FreeCAD.ActiveDocument.recompute()
00225     sketch.addGeometry(Part.LineSegment(points[0], points[1]), False)
00226     sketch.addGeometry(Part.LineSegment(points[1], points[2]), False)
00227     sketch.addGeometry(Part.LineSegment(points[2], points[3]), False)
00228     sketch.addGeometry(Part.LineSegment(points[3], points[4]), False)
00229     sketch.addGeometry(Part.LineSegment(points[4], points[5]), False)
00230     import Sketcher
00231     if amount_spacing_check:
00232         rebar = Arch.makeRebar(structure, sketch, diameter, amount_spacing_value, f_cover)
  
```

```

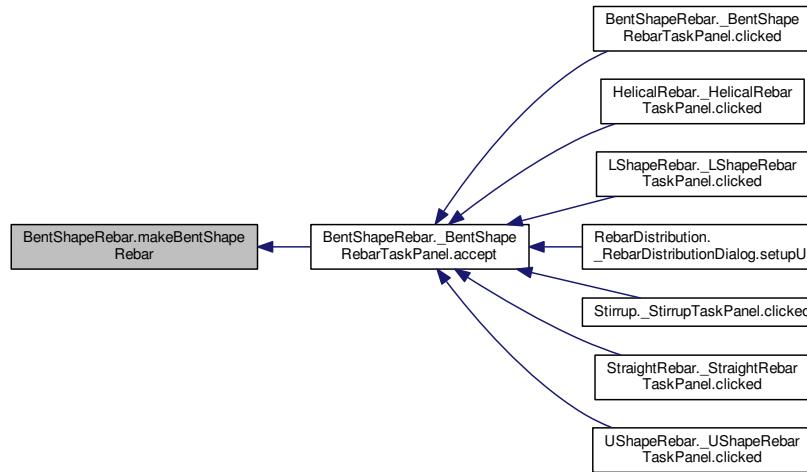
00232     FreeCAD.ActiveDocument.recompute()
00233     else:
00234         size = (ArchCommands.projectToVector(structure.Shape.copy(), face.normalAt(0, 0))).Length
00235         rebar = Arch.makeRebar(structure, sketch, diameter, int((size - diameter) / amount_spacing_value),
00236         f_cover)
00237         rebar.Rounding = rounding
00238         # Adds properties to the rebar object
00239         rebar.ViewObject.addProperty("App::PropertyString", "RebarShape", "RebarDialog", QT_TRANSLATE_NOOP("App::Property", "Shape of rebar")).RebarShape = "BentShapeRebar"
00240         rebar.ViewObject.setEditorMode("RebarShape", 2)
00241         rebar.addProperty("App::PropertyDistance", "FrontCover", "RebarDialog", QT_TRANSLATE_NOOP("App::Property", "Front cover of rebar")).FrontCover = f_cover
00242         rebar.setEditorMode("FrontCover", 2)
00243         rebar.addProperty("App::PropertyDistance", "LeftCover", "RebarDialog", QT_TRANSLATE_NOOP("App::Property", "Left Side cover of rebar")).LeftCover = l_cover
00244         rebar.setEditorMode("LeftCover", 2)
00245         rebar.addProperty("App::PropertyDistance", "RightCover", "RebarDialog", QT_TRANSLATE_NOOP("App::Property", "Right Side cover of rebar")).RightCover = r_cover
00246         rebar.setEditorMode("RightCover", 2)
00247         rebar.addProperty("App::PropertyDistance", "BottomCover", "RebarDialog", QT_TRANSLATE_NOOP("App::Property", "Bottom cover of rebar")).BottomCover = b_cover
00248         rebar.setEditorMode("BottomCover", 2)
00249         rebar.addProperty("App::PropertyBool", "AmountCheck", "RebarDialog", QT_TRANSLATE_NOOP("App::Property", "Amount radio button is checked")).AmountCheck
00250         rebar.setEditorMode("AmountCheck", 2)
00251         rebar.addProperty("App::PropertyDistance", "TopCover", "RebarDialog", QT_TRANSLATE_NOOP("App::Property", "Top cover of rebar")).TopCover = t_cover
00252         rebar.setEditorMode("TopCover", 2)
00253         rebar.addProperty("App::PropertyDistance", "TrueSpacing", "RebarDialog", QT_TRANSLATE_NOOP("App::Property", "Spacing between of rebars")).TrueSpacing = amount_spacing_value
00254         rebar.addProperty("App::PropertyString", "Orientation", "RebarDialog", QT_TRANSLATE_NOOP("App::Property", "Shape of rebar")).Orientation = orientation
00255         rebar.setEditorMode("Orientation", 2)
00256         rebar.addProperty("App::PropertyDistance", "BentLength", "RebarDialog", QT_TRANSLATE_NOOP("App::Property", "BentLength cover of rebar")).BentLength = bentLength
00257         rebar.setEditorMode("BentLength", 2)
00258         rebar.addProperty("App::PropertyDistance", "BentAngle", "RebarDialog", QT_TRANSLATE_NOOP("App::Property", "Bent Angle of rebar")).BentAngle = bentAngle
00259         rebar.setEditorMode("BentAngle", 2)
00260
00261     if amount_spacing_check:
00262         rebar.AmountCheck = True
00263     else:
00264         rebar.AmountCheck = False
00265         rebar.TrueSpacing = amount_spacing_value
00266         rebar.Label = "BentShapeRebar"
00267         FreeCAD.ActiveDocument.recompute()
00268     return rebar
00269

```

Here is the call graph for this function:



Here is the caller graph for this function:



6.1.2 Variable Documentation

6.1.2.1 `string BentShapeRebar.__author__ = "Amritpal Singh" [private]`

Definition at line 25 of file [BentShapeRebar.py](#).

6.1.2.2 `string BentShapeRebar.__title__ = "BentShapeRebar" [private]`

Definition at line 24 of file [BentShapeRebar.py](#).

6.1.2.3 `string BentShapeRebar.__url__ = "https://www.freecadweb.org" [private]`

Definition at line 26 of file [BentShapeRebar.py](#).

6.2 HelicalRebar Namespace Reference

Classes

- class [_HelicalRebarTaskPanel](#)

Functions

- def `getpointsOfHelicalRebar (FacePRM, s_cover, b_cover, t_cover, pitch, edges, diameter, size, direction)`
- def `createHelicalWire (FacePRM, s_cover, b_cover, t_cover, pitch, size, direction, helix=None)`
- def `makeHelicalRebar (s_cover, b_cover, diameter, t_cover, pitch, structure=None, facename=None)`
- def `editHelicalRebar (Rebar, s_cover, b_cover, diameter, t_cover, pitch, structure=None, facename=None)`
- def `editDialog (vobj)`
- def `CommandHelicalRebar ()`

Variables

- string `__title__` = "HelicalRebar"
- string `__author__` = "Amritpal Singh"
- string `__url__` = "https://www.freecadweb.org"

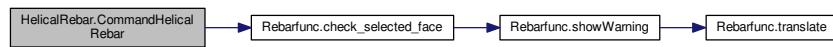
6.2.1 Function Documentation

6.2.1.1 def HelicalRebar.CommandHelicalRebar ()

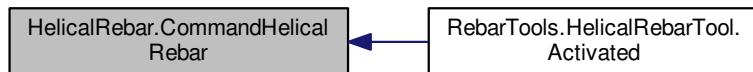
Definition at line 241 of file [HelicalRebar.py](#).

```
00241 def CommandHelicalRebar():
00242     selected_obj = check_selected_face()
00243     if selected_obj:
00244         FreeCADGui.Control.showDialog(_HelicalRebarTaskPanel())
00245
```

Here is the call graph for this function:



Here is the caller graph for this function:



6.2.1.2 def HelicalRebar.createHelicalWire (FacePRM, s_cover, b_cover, t_cover, pitch, size, direction, helix=None)

```
createHelicalWire(FacePRM, SideCover, BottomCover, TopCover, Pitch, Size, Direction, Helix = None):
It creates a helical wire.
```

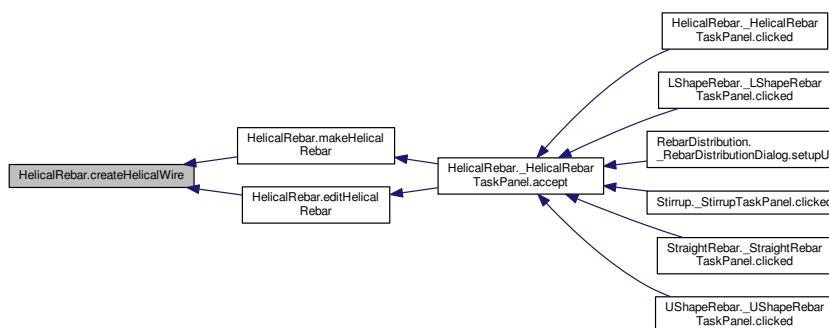
Definition at line 75 of file [HelicalRebar.py](#).

```

00075 def createHelicalWire(FacePRM, s_cover, b_cover, t_cover, pitch, size, direction, helix =
None):
00076     """ createHelicalWire(FacePRM, SideCover, BottomCover, TopCover, Pitch, Size, Direction, Helix = None):
00077     It creates a helical wire."""
00078     import Part
00079     if not helix:
00080         helix = FreeCAD.ActiveDocument.addObject("Part::Helix", "Helix")
00081     helix.Pitch = pitch
00082     helix.Radius = FacePRM[0][0] / 2 - s_cover
00083     helix.Angle = 0
00084     helix.LocalCoord = 0
00085     helix.Height = size - b_cover - t_cover
00086     if round(direction.x) == 1:
00087         helix.Placement.Base = FreeCAD.Vector(FacePRM[1][0] - b_cover, FacePRM[1][1], FacePRM[1][2])
00088         helix.Placement.Rotation = FreeCAD.Rotation(FreeCAD.Vector(0, -1, 0), 90)
00089     elif round(direction.x) == -1:
00090         helix.Placement.Base = FreeCAD.Vector(FacePRM[1][0] + t_cover, FacePRM[1][1], FacePRM[1][2])
00091         helix.Placement.Rotation = FreeCAD.Rotation(FreeCAD.Vector(0, -1, 0), -90)
00092     elif round(direction.y) == 1:
00093         helix.Placement.Base = FreeCAD.Vector(FacePRM[1][0], FacePRM[1][1] - b_cover, FacePRM[1][2])
00094         helix.Placement.Rotation = FreeCAD.Rotation(FreeCAD.Vector(1, 0, 0), 90)
00095     elif round(direction.y) == -1:
00096         helix.Placement.Base = FreeCAD.Vector(FacePRM[1][0], FacePRM[1][1] + t_cover, FacePRM[1][2])
00097         helix.Placement.Rotation = FreeCAD.Rotation(FreeCAD.Vector(-1, 0, 0), 90)
00098     elif round(direction.z) == 1:
00099         helix.Placement.Base = FreeCAD.Vector(FacePRM[1][0], FacePRM[1][1], FacePRM[1][2] - size + b_cover)
00100         helix.Placement.Rotation = FreeCAD.Rotation(FreeCAD.Vector(0, 0, 1), 0)
00101     elif round(direction.z) == -1:
00102         helix.Placement.Base = FreeCAD.Vector(FacePRM[1][0], FacePRM[1][1], FacePRM[1][2] + b_cover)
00103         helix.Placement.Rotation = FreeCAD.Rotation(FreeCAD.Vector(0, 0, -1), 0)
00104     FreeCAD.ActiveDocument.recompute()
00105     return helix
00106

```

Here is the caller graph for this function:



6.2.1.3 def HelicalRebar.editDialog (vobj)

Definition at line 231 of file [HelicalRebar.py](#).

```

00231 def editDialog(vobj):
00232     FreeCADGui.Control.closeDialog()
00233     obj = _HelicalRebarTaskPanel(vobj.Object)
00234     obj.form.sideCover.setText(str(vobj.Object.SideCover))
00235     obj.form.bottomCover.setText(str(vobj.Object.BottomCover))
00236     obj.form.diameter.setText(str(vobj.Object.Diameter))
00237     obj.form.topCover.setText(str(vobj.Object.TopCover))
00238     obj.form.pitch.setText(str(vobj.Object.Pitch))
00239     FreeCADGui.Control.showDialog(obj)
00240

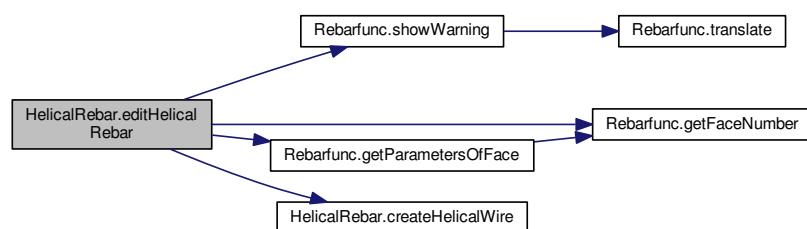
```

```
6.2.1.4 def HelicalRebar.editHelicalRebar( Rebar, s_cover, b_cover, diameter, t_cover, pitch, structure =None,
                                           facename =None )
```

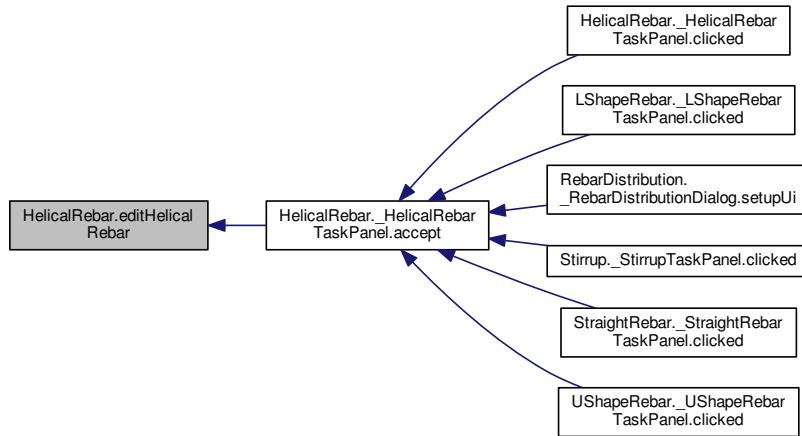
Definition at line 203 of file [HelicalRebar.py](#).

```
00203 def editHelicalRebar(Rebar, s_cover, b_cover, diameter, t_cover, pitch, structure = None,
                           facename = None):
00204     sketch = Rebar.Base
00205     if structure and facename:
00206         sketch.Support = [(structure, facename)]
00207     # Check if sketch support is empty.
00208     if not sketch.Support:
00209         showWarning("You have checked remove external geometry of base sketchs when needed.\nTo
unchecked Edit->Preferences->Arch.")
00210     return
00211     # Assigned values
00212     facename = sketch.Support[0][1][0]
00213     structure = sketch.Support[0][0]
00214     face = structure.Shape.Faces[getFaceNumber(facename) - 1]
00215     #StructurePRM = getTrueParametersOfStructure(structure)
00216     # Get parameters of the face where sketch of rebar is drawn
00217     FacePRM = getParametersOfFace(structure, facename, False)
00218     size = (ArchCommands.projectToVector(structure.Shape.copy(), face.normalAt(0, 0))).Length
00219     normal = face.normalAt(0, 0)
00220     #normal = face.Placement.Rotation.inverted().multVec(normal)
00221     helix = createHelicalWire(FacePRM, s_cover, b_cover, t_cover, pitch, size, normal,
00222                               Rebar.Base)
00223     FreeCAD.ActiveDocument.recompute()
00224     Rebar.Diameter = diameter
00225     Rebar.SideCover = s_cover
00226     Rebar.BottomCover = b_cover
00227     Rebar.TopCover = t_cover
00228     Rebar.Pitch = pitch
00229     FreeCAD.ActiveDocument.recompute()
00230     return Rebar
```

Here is the call graph for this function:



Here is the caller graph for this function:



6.2.1.5 def HelicalRebar.getpointsOfHelicalRebar (FacePRM, s_cover, b_cover, t_cover, pitch, edges, diameter, size, direction)

getpointsOfHelicalRebar(FacePRM, s_cover, b_cover, t_cover):
Return points of the LShape rebar in the form of array for sketch.

Definition at line 39 of file [HelicalRebar.py](#).

```

00039 def getpointsOfHelicalRebar(FacePRM, s_cover, b_cover, t_cover, pitch, edges,
00040     diameter, size, direction):
00041     """ getpointsOfHelicalRebar(FacePRM, s_cover, b_cover, t_cover):
00042     Return points of the LShape rebar in the form of array for sketch."""
00043     dx = s_cover + diameter / 2
00044     dz = float(pitch) / edges
00045     R = diameter / 2 - dx
00046     R = FacePRM[0][0] / 2 - s_cover
00047     points = []
00048     if direction[2] in {-1,1}:
00049         z = 0
00050         l = 0
00051         if direction[2] == 1:
00052             zz = FacePRM[1][2] - t_cover
00053         elif direction[2] == -1:
00054             zz = FacePRM[1][2] + b_cover
00055         count = 0
00056         flag = False
00057         while (round(z) < abs(size - b_cover - t_cover)):
00058             for i in range(0, int(edges) + 1):
00059                 if not i and flag:
00060                     continue
00061                 if not flag:
00062                     z -= dz
00063                     flag = True
00064                     iAngle = i * 360 / edges
00065                     x = FacePRM[1][0] + R * math.cos(math.radians(iAngle))
00066                     y = FacePRM[1][1] + R * math.sin(math.radians(iAngle))
00067                     points.append(FreeCAD.Vector(x, y, zz))
00068                     count += 1
00069                     if direction[2] == 1:
00070                         zz -= dz
00071                     elif direction[2] == -1:
00072                         zz += dz
00073                     z += dz
00074     return points
00074
  
```

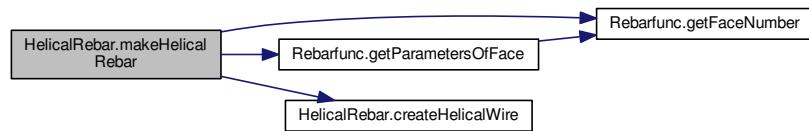
```
6.2.1.6 def HelicalRebar.makeHelicalRebar( s_cover, b_cover, diameter, t_cover, pitch, structure =None, facename =
    None )
```

`makeHelicalRebar(SideCover, BottomCover, Diameter, TopCover, Pitch, Structure, Facename):`
Adds the Helical reinforcement bar to the selected structural object.

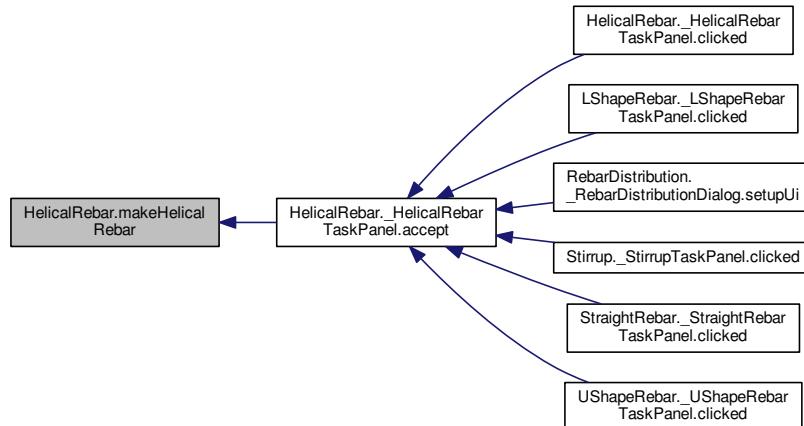
Definition at line 165 of file [HelicalRebar.py](#).

```
00165 def makeHelicalRebar(s_cover, b_cover, diameter, t_cover, pitch, structure = None, facename
    = None):
00166     """ makeHelicalRebar(SideCover, BottomCover, Diameter, TopCover, Pitch, Structure, Facename):
00167         Adds the Helical reinforcement bar to the selected structural object."""
00168     if not structure and not facename:
00169         selected_obj = FreeCADGui.Selection.getSelectionEx()[0]
00170         structure = selected_obj.Object
00171         facename = selected_obj.SubElementNames[0]
00172         face = structure.Shape.Faces[getFaceNumber(facename) - 1]
00173 #StructurePRM = getTrueParametersOfStructure(structure)
00174 FacePRM = getParametersOfFace(structure, facename, False)
00175 if not FacePRM:
00176     FreeCAD.Console.PrintError("Cannot identified shape or from which base object sturctural element is
derived\n")
00177     return
00178 size = (ArchCommands.projectToVector(structure.Shape.copy(), face.normalAt(0, 0))).Length
00179 normal = face.normalAt(0,0)
00180 #normal = face.Placement.Rotation.inverted().multVec(normal)
00181 import Arch
00182 helix = createHelicalWire(FacePRM, s_cover, b_cover, t_cover, pitch, size, normal)
00183 helix.Support = [(structure, facename)]
00184 rebar = Arch.makeRebar(structure, helix, diameter, 1, 0)
00185 rebar.OffsetStart = 0
00186 rebar.OffsetEnd = 0
00187 FreeCAD.ActiveDocument.recompute()
00188 # Adds properties to the rebar object
00189 rebar.ViewObject.addProperty("App::PropertyString", "RebarShape", "RebarDialog", QT_TRANSLATE_NOOP("App::Property",
"Shape of rebar")).RebarShape = "HelicalRebar"
00190 rebar.ViewObject.setEditorMode("RebarShape", 2)
00191 rebar.addProperty("App::PropertyDistance", "SideCover", "RebarDialog", QT_TRANSLATE_NOOP("App::Property",
", "Front cover of rebar")).SideCover = s_cover
00192 rebar.setEditorMode("SideCover", 2)
00193 rebar.addProperty("App::PropertyDistance", "Pitch", "RebarDialog", QT_TRANSLATE_NOOP("App::Property",
"Left Side cover of rebar")).Pitch = pitch
00194 rebar.setEditorMode("Pitch", 2)
00195 rebar.addProperty("App::PropertyDistance", "BottomCover", "RebarDialog", QT_TRANSLATE_NOOP("App::Property",
"App::Property", "Bottom cover of rebar")).BottomCover = b_cover
00196 rebar.setEditorMode("BottomCover", 2)
00197 rebar.addProperty("App::PropertyDistance", "TopCover", "RebarDialog", QT_TRANSLATE_NOOP("App::Property",
", "Top cover of rebar")).TopCover = t_cover
00198 rebar.setEditorMode("TopCover", 2)
00199 rebar.Label = "HelicalRebar"
00200 FreeCAD.ActiveDocument.recompute()
00201 return rebar
00202
```

Here is the call graph for this function:



Here is the caller graph for this function:



6.2.2 Variable Documentation

6.2.2.1 string HelicalRebar.__author__ = "Amritpal Singh" [private]

Definition at line 25 of file [HelicalRebar.py](#).

6.2.2.2 string HelicalRebar.__title__ = "HelicalRebar" [private]

Definition at line 24 of file [HelicalRebar.py](#).

6.2.2.3 string HelicalRebar.__url__ = "https://www.freecadweb.org" [private]

Definition at line 26 of file [HelicalRebar.py](#).

6.3 LShapeRebar Namespace Reference

Classes

- class [_LShapeRebarTaskPanel](#)

Functions

- def [getpointsOfLShapeRebar](#) (FacePRM, l_cover, r_cover, b_cover, t_cover, orientation)
- def [makeLShapeRebar](#) (f_cover, b_cover, l_cover, r_cover, diameter, t_cover, rounding, amount_spacing_check, amount_spacing_value, orientation="Bottom Left", structure=None, facename=None)
- def [editLShapeRebar](#) (Rebar, f_cover, b_cover, l_cover, r_cover, diameter, t_cover, rounding, amount_spacing_check, amount_spacing_value, orientation, structure=None, facename=None)
- def [editDialog](#) (vobj)
- def [CommandLShapeRebar](#) ()

Variables

- string `__title__` = "LShapeRebar"
- string `__author__` = "Amritpal Singh"
- string `__url__` = "https://www.freecadweb.org"

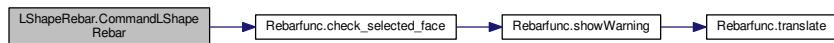
6.3.1 Function Documentation

6.3.1.1 def LShapeRebar.CommandLShapeRebar ()

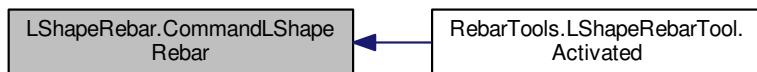
Definition at line 300 of file [LShapeRebar.py](#).

```
00300 def CommandLShapeRebar():
00301     selected_obj = check_selected_face()
00302     if selected_obj:
00303         FreeCADGui.Control.showDialog(_LShapeRebarTaskPanel())
00304
```

Here is the call graph for this function:



Here is the caller graph for this function:



6.3.1.2 def LShapeRebar.editDialog (vobj)

Definition at line 278 of file [LShapeRebar.py](#).

```
00278 def editDialog(vobj):
00279     FreeCADGui.Control.closeDialog()
00280     obj = _LShapeRebarTaskPanel(vobj.Object)
00281     obj.form.frontCover.setText(str(vobj.Object.FrontCover))
00282     obj.form.l_sideCover.setText(str(vobj.Object.LeftCover))
00283     obj.form.r_sideCover.setText(str(vobj.Object.RightCover))
00284     obj.form.bottomCover.setText(str(vobj.Object.BottomCover))
00285     obj.form.diameter.setText(str(vobj.Object.Diameter))
00286     obj.form.topCover.setText(str(vobj.Object.TopCover))
00287     obj.form.rounding.setValue(vobj.Object.Rounding)
00288     obj.form.orientation.setCurrentIndex(obj.form.orientation.findText(str(vobj.Object.Orientation)))
00289     if vobj.Object.AmountCheck:
00290         obj.form.amount.setValue(vobj.Object.Amount)
00291     else:
00292         obj.form.amount_radio.setChecked(False)
00293         obj.form.spacing_radio.setChecked(True)
00294         obj.form.amount.setDisabled(True)
00295         obj.form.spacing.setEnabled(True)
00296         obj.form.spacing.setText(str(vobj.Object.TrueSpacing))
00297     #obj.form.PickSelectedFace.setVisible(False)
00298     FreeCADGui.Control.showDialog(obj)
00299
```

6.3.1.3 `def LShapeRebar.editLShapeRebar(Rebar, f_cover, b_cover, l_cover, r_cover, diameter, t_cover, rounding, amount_spacing_check, amount_spacing_value, orientation, structure =None, facename =None)`

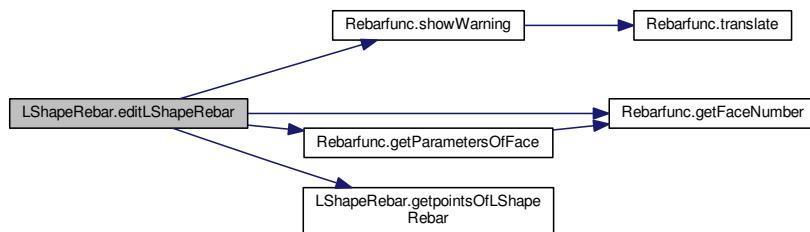
Definition at line 230 of file [LShapeRebar.py](#).

```

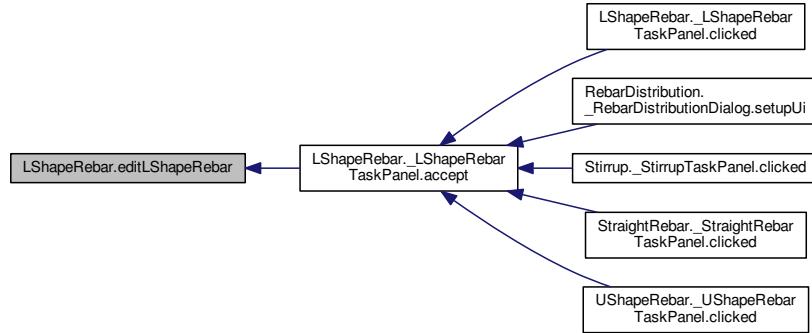
00230 def editLShapeRebar(Rebar, f_cover, b_cover, l_cover, r_cover, diameter, t_cover, rounding,
00231     amount_spacing_check, amount_spacing_value, orientation, structure = None, facename = None):
00232     sketch = Rebar.Base
00233     if structure and facename:
00234         sketch.Support = [(structure, facename)]
00235     # Check if sketch support is empty.
00236     if not sketch.Support:
00237         showWarning("You have checked remove external geometry of base sketchs when needed.\nTo
00238         uncheck Edit->Preferences->Arch.")
00239     # Assigned values
00240     facename = sketch.Support[0][1][0]
00241     structure = sketch.Support[0][0]
00242     face = structure.Shape.Faces[getFaceNumber(facename) - 1]
00243     #StructurePRM = getTrueParametersOfStructure(structure)
00244     # Get parameters of the face where sketch of rebar is drawn
00245     FacePRM = getParametersOfFace(structure, facename)
00246     points = getpointsOfLShapeRebar(FacePRM, l_cover, r_cover, b_cover, t_cover,
00247         orientation)
00248     sketch.movePoint(0, 1, points[0], 0)
00249     FreeCAD.ActiveDocument.recompute()
00250     sketch.movePoint(0, 2, points[1], 0)
00251     FreeCAD.ActiveDocument.recompute()
00252     sketch.movePoint(1, 1, points[1], 0)
00253     FreeCAD.ActiveDocument.recompute()
00254     sketch.movePoint(1, 2, points[2], 0)
00255     FreeCAD.ActiveDocument.recompute()
00256     Rebar.OffsetStart = f_cover
00257     Rebar.OffsetEnd = f_cover
00258     if amount_spacing_check:
00259         Rebar.Amount = amount_spacing_value
00260         FreeCAD.ActiveDocument.recompute()
00261     else:
00262         size = (ArchCommands.projectToVector(structure.Shape.copy(), face.normalAt(0, 0))).Length
00263         Rebar.Amount = int((size - diameter) / amount_spacing_value)
00264         FreeCAD.ActiveDocument.recompute()
00265     Rebar.AmountCheck = False
00266     Rebar.Diameter = diameter
00267     Rebar.FrontCover = f_cover
00268     Rebar.LeftCover = l_cover
00269     Rebar.RightCover = r_cover
00270     Rebar.BottomCover = b_cover
00271     Rebar.TopCover = t_cover
00272     Rebar.Rounding = rounding
00273     Rebar.TrueSpacing = amount_spacing_value
00274     Rebar.Orientation = orientation
00275     FreeCAD.ActiveDocument.recompute()
00276     return Rebar
00277

```

Here is the call graph for this function:



Here is the caller graph for this function:



6.3.1.4 def LShapeRebar.getpointsOfLShapeRebar (FacePRM, l_cover, r_cover, b_cover, t_cover, orientation)

getpointsOfLShapeRebar(FacePRM, LeftCover, RightCover, BottomCover, TopCover, Orientation):
 Return points of the LShape rebar in the form of array for sketch.
 It takes four different orientations input i.e. 'Bottom Left', 'Bottom Right ', 'Top Left', 'Top Right'.

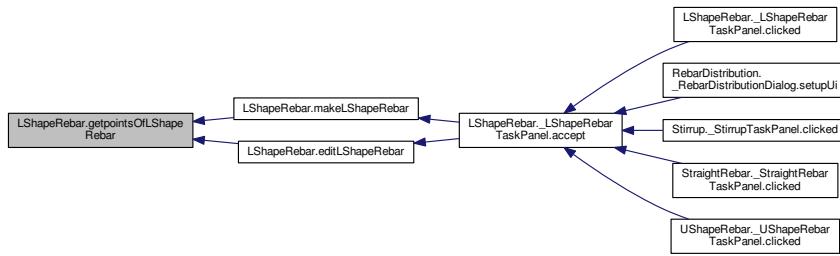
Definition at line 40 of file [LShapeRebar.py](#).

```

00040 def getpointsOfLShapeRebar(FacePRM, l_cover, r_cover, b_cover, t_cover, orientation):
00041     """ getpointsOfLShapeRebar(FacePRM, LeftCover, RightCover, BottomCover, TopCover, Orientation):
00042     Return points of the LShape rebar in the form of array for sketch.
00043     It takes four different orientations input i.e. 'Bottom Left', 'Bottom Right ', 'Top Left', 'Top Right'
00044
00045     """
00046     if orientation == "Bottom Left":
00047         x1 = FacePRM[1][0] - FacePRM[0][0] / 2 + l_cover
00048         y1 = FacePRM[1][1] + FacePRM[0][1] / 2 - t_cover
00049         x2 = FacePRM[1][0] - FacePRM[0][0] / 2 + l_cover
00050         y2 = FacePRM[1][1] - FacePRM[0][1] / 2 + b_cover
00051         x3 = FacePRM[1][0] - FacePRM[0][0] / 2 + FacePRM[0][0] - r_cover
00052         y3 = FacePRM[1][1] - FacePRM[0][1] / 2 + b_cover
00053     elif orientation == "Bottom Right":
00054         x1 = FacePRM[1][0] - FacePRM[0][0] / 2 + FacePRM[0][0] - r_cover
00055         y1 = FacePRM[1][1] + FacePRM[0][1] / 2 - t_cover
00056         x2 = FacePRM[1][0] - FacePRM[0][0] / 2 + FacePRM[0][0] - r_cover
00057         y2 = FacePRM[1][1] - FacePRM[0][1] / 2 + b_cover
00058         x3 = FacePRM[1][0] - FacePRM[0][0] / 2 + l_cover
00059         y3 = FacePRM[1][1] - FacePRM[0][1] / 2 + b_cover
00060     elif orientation == "Top Left":
00061         x1 = FacePRM[1][0] - FacePRM[0][0] / 2 + l_cover
00062         y1 = FacePRM[1][1] - FacePRM[0][1] / 2 + b_cover
00063         x2 = FacePRM[1][0] - FacePRM[0][0] / 2 + l_cover
00064         y2 = FacePRM[1][1] + FacePRM[0][1] / 2 - t_cover
00065         x3 = FacePRM[1][0] - FacePRM[0][0] / 2 + FacePRM[0][0] - r_cover
00066         y3 = FacePRM[1][1] + FacePRM[0][1] / 2 - t_cover
00067     elif orientation == "Top Right":
00068         x1 = FacePRM[1][0] - FacePRM[0][0] / 2 + FacePRM[0][0] - r_cover
00069         y1 = FacePRM[1][1] - FacePRM[0][1] / 2 + b_cover
00070         x2 = FacePRM[1][0] - FacePRM[0][0] / 2 + FacePRM[0][0] - r_cover
00071         y2 = FacePRM[1][1] + FacePRM[0][1] / 2 - t_cover
00072         x3 = FacePRM[1][0] - FacePRM[0][0] / 2 + l_cover
00073         y3 = FacePRM[1][1] + FacePRM[0][1] / 2 - t_cover
00074     return [FreeCAD.Vector(x1, y1, 0), FreeCAD.Vector(x2, y2, 0), \
00075             FreeCAD.Vector(x3, y3, 0)]
00076

```

Here is the caller graph for this function:



6.3.1.5 def LShapeRebar.makeLShapeRebar (*f_cover*, *b_cover*, *l_cover*, *r_cover*, *diameter*, *t_cover*, *rounding*, *amount_spacing_check*, *amount_spacing_value*, *orientation* = "Bottom Left", *structure* = None, *facename* = None)

makeLShapeRebar(FrontCover, BottomCover, LeftCover, RightCover, Diameter, TopCover, Rounding, AmountSpacingCheck, Orientation, Structure, Facename): Adds the L-Shape reinforcement bar to the selected structural object. It takes four different orientations input i.e. 'Bottom Left', 'Bottom Right', 'Top Left', 'Top Right'.

Definition at line 169 of file [LShapeRebar.py](#).

```

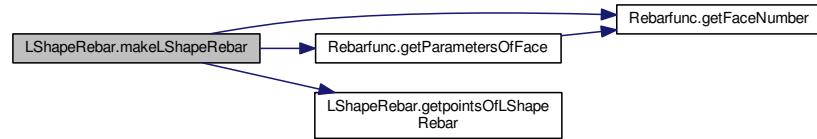
00169 def makeLShapeRebar(f_cover, b_cover, l_cover, r_cover, diameter, t_cover, rounding,
00170     amount_spacing_check, amount_spacing_value, orientation = "Bottom Left", structure = None, facename = None):
00171     """ makeLShapeRebar(FrontCover, BottomCover, LeftCover, RightCover, Diameter, TopCover, Rounding,
00172     AmountSpacingCheck, AmountSpacingValue,
00173     Orientation, Structure, Facename): Adds the L-Shape reinforcement bar to the selected structural
00174     object.
00175     It takes four different orientations input i.e. 'Bottom Left', 'Bottom Right', 'Top Left', 'Top Right'
00176
00177     """
00178     if not structure and not facename:
00179         selected_obj = FreeCADGui.Selection.getSelectionEx()[0]
00180         structure = selected_obj.Object
00181         facename = selected_obj.SubElementNames[0]
00182         face = structure.Shape.Faces[getFaceNumber(facename) - 1]
00183         #StructurePRM = getTrueParametersOfStructure(structure)
00184         FacePRM = getParametersOfFace(structure, facename)
00185         if not FacePRM:
00186             FreeCAD.Console.PrintError("Cannot identified shape or from which base object sturctural element is
00187 derived\n")
00188             return
00189         # Get points of L-Shape rebar
00190         points = getpointsOfLShapeRebar(FacePRM, l_cover, r_cover, b_cover, t_cover,
00191         orientation)
00192         import Part
00193         import Arch
00194         sketch = FreeCAD.activeDocument().addObject('Sketcher::SketchObject', 'Sketch')
00195         sketch.MapMode = "FlatFace"
00196         sketch.Support = [(structure, facename)]
00197         FreeCAD.ActiveDocument.recompute()
00198         sketch.addGeometry(Part.LineSegment(points[0], points[1]), False)
00199         sketch.addGeometry(Part.LineSegment(points[1], points[2]), False)
00200         import Sketcher
00201         if amount_spacing_check:
00202             rebar = Arch.makeRebar(structure, sketch, diameter, amount_spacing_value, f_cover)
00203             FreeCAD.ActiveDocument.recompute()
00204         else:
00205             size = (ArchCommands.projectToVector(structure.Shape.copy(), face.normalAt(0, 0))).Length
00206             rebar = Arch.makeRebar(structure, sketch, diameter, int((size - diameter) / amount_spacing_value),
00207             f_cover)
00208             rebar.Rounding = rounding
00209             # Adds properties to the rebar object
00210             rebar.ViewObject.addProperty("App::PropertyString", "RebarShape", "RebarDialog", QT_TRANSLATE_NOOP("App::Property",
00211             "Shape of rebar")).RebarShape = "LShapeRebar"
00212             rebar.ViewObject.setEditorMode("RebarShape", 2)
00213             rebar.addProperty("App::PropertyDistance", "FrontCover", "RebarDialog", QT_TRANSLATE_NOOP("App::PropertyDistance",
00214             "Front Cover"))
  
```

```

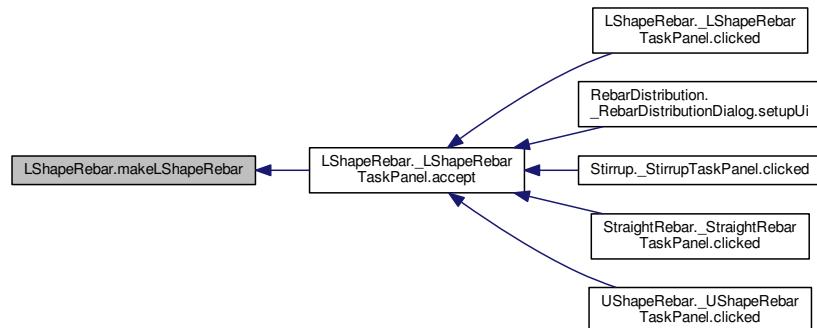
00206     App::Property", "Front cover of rebar").FrontCover = f_cover
00207     rebar.setEditorMode("FrontCover", 2)
00208     rebar.addProperty("App::PropertyDistance", "LeftCover", "RebarDialog", QT_TRANSLATE_NOOP("App::Property"
00209     ", "Left Side cover of rebar")).LeftCover = l_cover
00210     rebar.setEditorMode("LeftCover", 2)
00211     rebar.addProperty("App::PropertyDistance", "RightCover", "RebarDialog", QT_TRANSLATE_NOOP("
00212     App::Property", "Right Side cover of rebar")).RightCover = r_cover
00213     rebar.setEditorMode("RightCover", 2)
00214     rebar.addProperty("App::PropertyDistance", "BottomCover", "RebarDialog", QT_TRANSLATE_NOOP("
00215     App::Property", "Bottom cover of rebar")).BottomCover = b_cover
00216     rebar.setEditorMode("BottomCover", 2)
00217     rebar.addProperty("App::PropertyBool", "AmountCheck", "RebarDialog", QT_TRANSLATE_NOOP("App::Property",
00218     "Amount radio button is checked")).AmountCheck
00219     rebar.setEditorMode("AmountCheck", 2)
00220     rebar.addProperty("App::PropertyDistance", "TrueSpacing", "RebarDialog", QT_TRANSLATE_NOOP("App::Property"
00221     ", "Spacing between of rebars")).TrueSpacing = amount_spacing_value
00222     rebar.addProperty("App::PropertyString", "Orientation", "RebarDialog", QT_TRANSLATE_NOOP("App::Property
00223     ", "Shape of rebar")).Orientation = orientation
00224     rebar.setEditorMode("Orientation", 2)
00225     rebar.setEditorMode("TrueSpacing", 2)
00226     if amount_check:
00227         rebar.AmountCheck = True
00228     else:
00229         rebar.AmountCheck = False
00230     rebar.TrueSpacing = amount_spacing_value
00231     rebar.Label = "LShapeRebar"
00232     FreeCAD.ActiveDocument.recompute()
00233     return rebar
00234

```

Here is the call graph for this function:



Here is the caller graph for this function:



6.3.2 Variable Documentation

6.3.2.1 string LShapeRebar.__author__ = "Amritpal Singh" [private]

Definition at line 25 of file [LShapeRebar.py](#).

6.3.2.2 string LShapeRebar.__title__ = "LShapeRebar" [private]

Definition at line 24 of file [LShapeRebar.py](#).

6.3.2.3 string LShapeRebar.__url__ = "https://www.freecadweb.org" [private]

Definition at line 26 of file [LShapeRebar.py](#).

6.4 PopUpImage Namespace Reference

Classes

- class [PopUpImage](#)

Functions

- def [showPopUpImageDialog \(img\)](#)

Variables

- string __title__ = "PopUpImage"
- string __author__ = "Amritpal Singh"
- string __url__ = "https://www.freecadweb.org"

6.4.1 Function Documentation

6.4.1.1 def [PopUpImage.showPopUpImageDialog \(img \)](#)

`showPopUpImageDialog(image)`: This function will show a given image in a pop-up dialog box.

Definition at line 43 of file [PopUpImage.py](#).

```
00043 def showPopUpImageDialog(img):
00044     """ showPopUpImageDialog(image) : This function will show a given image in a pop-up
00045     dialog box."""
00046     dialog = PopUpImage(img)
00047     dialog.exec_()
00048
```

6.4.2 Variable Documentation

6.4.2.1 string [PopUpImage.__author__ = "Amritpal Singh" \[private\]](#)

Definition at line 25 of file [PopUpImage.py](#).

6.4.2.2 string PopUpImage.__title__ = "PopUpImage" [private]

Definition at line 24 of file [PopUpImage.py](#).

6.4.2.3 string PopUpImage.__url__ = "https://www.freecadweb.org" [private]

Definition at line 26 of file [PopUpImage.py](#).

6.5 RebarDistribution Namespace Reference

Classes

- class [_RebarDistributionDialog](#)

Functions

- def [getCustomSpacingString](#) (amount1, spacing1, amount2, spacing2, amount3, spacing3, frontCover, size)
- def [gettupleOfCustomSpacing](#) (span_string)
- def [runRebarDistribution](#) (self)
- def [removeRebarDistribution](#) (self)

Variables

- string [__title__](#) = "DialogDistribution"
- string [__author__](#) = "Amritpal Singh"
- string [__url__](#) = "https://www.freecadweb.org"
- [CustomSpacing](#)

6.5.1 Function Documentation

6.5.1.1 def RebarDistribution.getCustomSpacingString (*amount1, spacing1, amount2, spacing2, amount3, spacing3, frontCover, size*)

Definition at line 63 of file [RebarDistribution.py](#).

```
00063 def getCustomSpacingString(amount1, spacing1, amount2, spacing2, amount3, spacing3,
    frontCover, size):
00064     seg1_area = amount1 * spacing1 - spacing1 / 2
00065     seg3_area = amount3 * spacing3 - spacing3 / 2
00066     seg2_area = size - seg1_area - seg3_area - 2 * frontCover
00067     if seg2_area < 0:
00068         FreeCAD.Console.PrintError("Sum of length of segment 1 and segment 2 is greater than length of
        rebar expands.\n")
00069     return
00070     if spacing1 and spacing2 and spacing3 and amount1 and amount2 and amount3:
00071         pass
00072     else:
00073         if spacing1 and spacing2 and spacing3:
00074             amount2 = math.ceil(seg2_area / spacing2)
00075             spacing2 = seg2_area / amount2
00076         elif amount1 and amount2 and amount3:
00077             spacing2 = math.floor(seg2_area / amount2)
00078         CustomSpacing = str(amount1) + "@" + str(spacing1) + "+" + str(int(amount2)) + "@" + str(spacing2) + "+"
        " + str(amount3) + "@" + str(spacing3)
00079     return CustomSpacing
00080
```

6.5.1.2 def RebarDistribution.gettupleOfCustomSpacing (span_string)

gettupleOfCustomSpacing(span_string): This function take input in specific syntax and return output in the form of list. For eg.
Input: "3@100+2@200+3@100"
Output: [(3, 100), (2, 200), (3, 100)]

Definition at line 81 of file [RebarDistribution.py](#).

```
00081 def gettupleOfCustomSpacing(span_string):
00082     """ gettupleOfCustomSpacing(span_string): This function take input
00083     in specific syntax and return output in the form of list. For eg.
00084     Input: "3@100+2@200+3@100"
00085     Output: [(3, 100), (2, 200), (3, 100)]"""
00086     import string
00087     span_st = string.strip(span_string)
00088     span_sp = string.split(span_st, '+')
00089     index = 0
00090     spacinglist = []
00091     while index < len(span_sp):
00092         # Find "@" recursively in span_sp array.
00093         in_sp = string.split(span_sp[index], '@')
00094         spacinglist.append((int(in_sp[0]), float(in_sp[1])))
00095         index += 1
00096     return spacinglist
00097
```

6.5.1.3 def RebarDistribution.removeRebarDistribution (self)

Definition at line 108 of file [RebarDistribution.py](#).

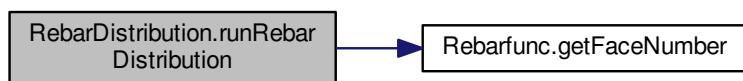
```
00108 def removeRebarDistribution(self):
00109     self.CustomSpacing = ""
00110     self.Rebar.CustomSpacing = ""
00111     FreeCAD.ActiveDocument.recompute()
00112
00113 #runRebarDistribution(App.ActiveDocument.Rebar)
00114
```

6.5.1.4 def RebarDistribution.runRebarDistribution (self)

Definition at line 98 of file [RebarDistribution.py](#).

```
00098 def runRebarDistribution(self):
00099     frontCover = self.form.frontCover.text()
00100     frontCover = FreeCAD.Units.Quantity(frontCover).Value
00101     face = self.SelectedObj.Shape.Faces[getFaceNumber(self.FaceName) - 1]
00102     size = (ArchCommands.projectToVector(self.SelectedObj.Shape.copy(), face.normalAt(0, 0))).Length
00103     dialog = _RebarDistributionDialog(frontCover, size)
00104     dialog.setupUi()
00105     dialog.form.exec_()
00106     self.CustomSpacing = dialog.CustomSpacing
00107
```

Here is the call graph for this function:



6.5.2 Variable Documentation

6.5.2.1 string `RebarDistribution.__author__ = "Amritpal Singh"` [private]

Definition at line 25 of file [RebarDistribution.py](#).

6.5.2.2 string `RebarDistribution.__title__ = "DialogDistribution"` [private]

Definition at line 24 of file [RebarDistribution.py](#).

6.5.2.3 string `RebarDistribution.__url__ = "https://www.freecadweb.org"` [private]

Definition at line 26 of file [RebarDistribution.py](#).

6.5.2.4 `RebarDistribution.CustomSpacing`

Definition at line 106 of file [RebarDistribution.py](#).

6.6 Rebarfunc Namespace Reference

Functions

- def `getEdgesAngle` (`edge1, edge2`)
- def `checkRectangle` (`edges`)
- def `getBaseStructuralObject` (`obj`)
- def `getBaseObject` (`obj`)
- def `getFaceNumber` (`s`)
- def `facenormalDirection` (`structure=None, facename=None`)
- def `getTrueParametersOfStructure` (`obj`)
- def `getParametersOfFace` (`structure, facename, sketch=True`)
- def `extendedTangentPartLength` (`rounding, diameter, angle`)
- def `extendedTangentLength` (`rounding, diameter, angle`)
- def `check_selected_face` ()
- def `getSelectedFace` (`self`)
- def `showWarning` (`message`)
- def `translate` (`context, text, disambig=None`)

Variables

- string `__title__ = "GenericRebarFuctions"`
- string `__author__ = "Amritpal Singh"`
- string `__url__ = "https://www.freecadweb.org"`
- `SelectedObj`
- `FaceName`

6.6.1 Function Documentation

6.6.1.1 def Rebarfunc.check_selected_face()

check_selected_face(): This function checks whether user have selected any face or not.

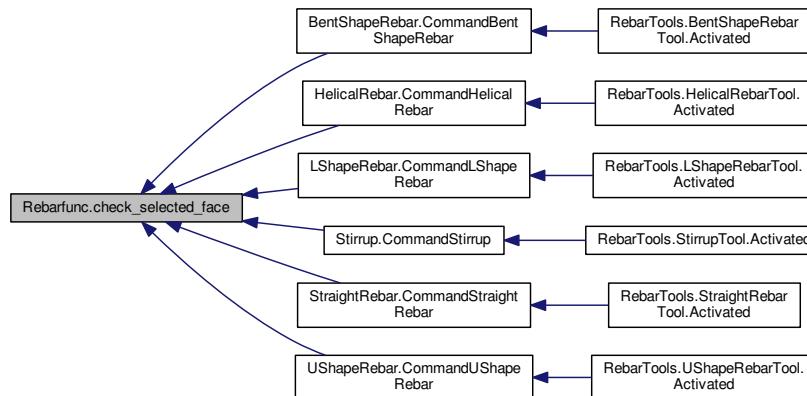
Definition at line 255 of file [Rebarfunc.py](#).

```
00255 def check_selected_face():
00256     """ check_selected_face(): This function checks whether user have selected
00257         any face or not."""
00258     selected_objs = FreeCADGui.Selection.getSelectionEx()
00259     if not selected_objs:
00260         showWarning("Select any face of the structural element.")
00261         selected_obj = None
00262     else:
00263         selected_face_names = selected_objs[0].SubElementNames
00264         if not selected_face_names:
00265             selected_obj = None
00266             showWarning("Select any face of the structural element.")
00267         elif "Face" in selected_face_names[0]:
00268             if len(selected_face_names) > 1:
00269                 showWarning("You have selected more than one face of the structural element.")
00270                 selected_obj = None
00271             elif len(selected_face_names) == 1:
00272                 selected_obj = selected_objs[0]
00273         else:
00274             showWarning("Select any face of the selected the face.")
00275             selected_obj = None
00276     return selected_obj
00277
```

Here is the call graph for this function:



Here is the caller graph for this function:



6.6.1.2 def Rebarfunc.checkRectangle(edges)

checkRectangle(edges[]): This function checks whether the given form rectangle or not. It will return True when edges form rectangular shape or return False when edges not form a rectangular.

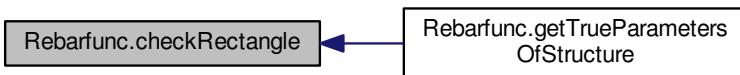
Definition at line 46 of file [Rebarfunc.py](#).

```
00046 def checkRectangle(edges):
00047     """ checkRectangle(edges[]): This function checks whether the given form rectangle
00048         or not. It will return True when edges form rectangular shape or return False
00049         when edges not form a rectangular."""
00050     angles = [round(getEdgesAngle(edges[0], edges[1])), round(
00051             getEdgesAngle(edges[0], edges[2])),
00052             round(getEdgesAngle(edges[0], edges[3]))]
00053     if angles.count(90) == 2 and (angles.count(180) == 1 or angles.count(0) == 1):
00054         return True
00055     else:
00056         return False
00056
```

Here is the call graph for this function:



Here is the caller graph for this function:



6.6.1.3 def Rebarfunc.extendedTangentLength(rounding, diameter, angle)

extendedTangentLength(rounding, diameter, angle): Get a extended length of rounding at the end of Stirrup for bent.

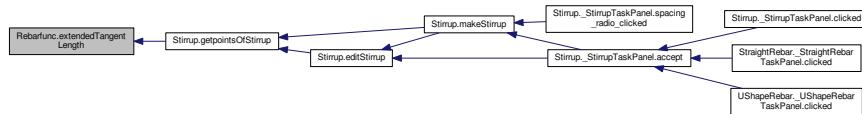
Definition at line 243 of file [Rebarfunc.py](#).

```

00243 def extendedTangentLength(rounding, diameter, angle):
00244     """ extendedTangentLength(rounding, diameter, angle): Get a extended
00245     length of rounding at the end of Stirrup for bent."""
00246     radius = rounding * diameter
00247     x1 = radius / math.sin(math.radians(angle))
00248     x2 = radius * math.tan(math.radians(90 - angle))
00249     return x1 + x2
00250
00251 # -----
00252 # Warning / Alert functions when user do something wrong.
00253 #-----
00254

```

Here is the caller graph for this function:



6.6.1.4 def Rebarfunc.extendedTangentPartLength (rounding, diameter, angle)

extendedTangentPartLength(rounding, diameter, angle): Get a extended length of rounding on corners.

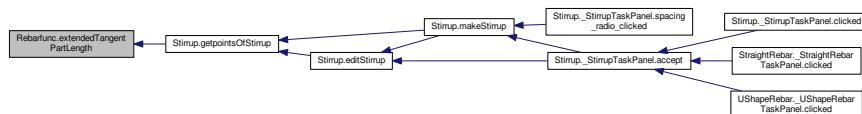
Definition at line 235 of file [Rebarfunc.py](#).

```

00235 def extendedTangentPartLength(rounding, diameter, angle):
00236     """ extendedTangentPartLength(rounding, diameter, angle): Get a extended
00237     length of rounding on corners."""
00238     radius = rounding * diameter
00239     x1 = radius / math.tan(math.radians(angle))
00240     x2 = radius / math.cos(math.radians(90 - angle)) - radius
00241     return x1 + x2
00242

```

Here is the caller graph for this function:



6.6.1.5 def Rebarfunc.facenormalDirection (structure=None, facename=None)

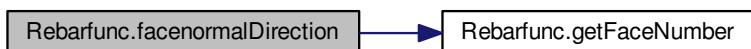
Definition at line 81 of file [Rebarfunc.py](#).

```

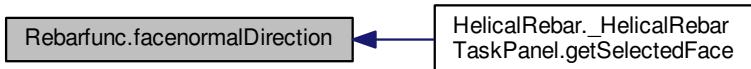
00081 def facenormalDirection(structure = None, facename = None):
00082     if not structure and not facename:
00083         selected_obj = FreeCADGui.Selection.getSelectionEx()[0]
00084         structure = selected_obj.Object
00085         facename = selected_obj.SubElementNames[0]
00086     face = structure.Shape.Faces[getFaceNumber(facename) - 1]
00087     normal = face.normalAt(0,0)
00088     normal = face.Placement.Rotation.inverted().multVec(normal)
00089     return normal
00090
00091 # -----
00092 # Main functions which is use while creating any rebar.
00093 # -----
00094

```

Here is the call graph for this function:



Here is the caller graph for this function:



6.6.1.6 def Rebarfunc.getBaseObject (obj)

getBaseObject (obj): This function will return last base object.

Definition at line 65 of file [Rebarfunc.py](#).

```

00065 def getBaseObject(obj):
00066     """ getBaseObject(obj): This function will return last base object."""
00067     if hasattr(obj, "Base"):
00068         return getBaseObject(obj.Base)
00069     else:
00070         return obj
00071

```

Here is the caller graph for this function:



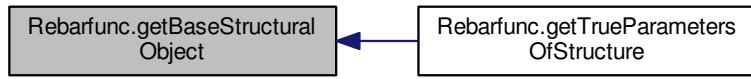
6.6.1.7 def Rebarfunc.getBaseStructuralObject(obj)

```
getBaseStructuralObject(obj): This function will return last base
structural object.
```

Definition at line 57 of file [Rebarfunc.py](#).

```
00057 def getBaseStructuralObject(obj):
00058     """ getBaseStructuralObject(obj): This function will return last base
00059         structural object."""
00060     if not obj.Base:
00061         return obj
00062     else:
00063         return getBaseStructuralObject(obj.Base)
00064
```

Here is the caller graph for this function:



6.6.1.8 def Rebarfunc.getEdgesAngle(edge1, edge2)

```
getEdgesAngle(edge1, edge2): returns a angle between two edges.
```

Definition at line 38 of file [Rebarfunc.py](#).

```
00038 def getEdgesAngle(edge1, edge2):
00039     """ getEdgesAngle(edge1, edge2): returns a angle between two edges."""
00040     vec1 = vec(edge1)
00041     vec2 = vec(edge2)
00042     angle = vec1.getAngle(vec2)
00043     angle = math.degrees(angle)
00044     return angle
00045
```

Here is the caller graph for this function:



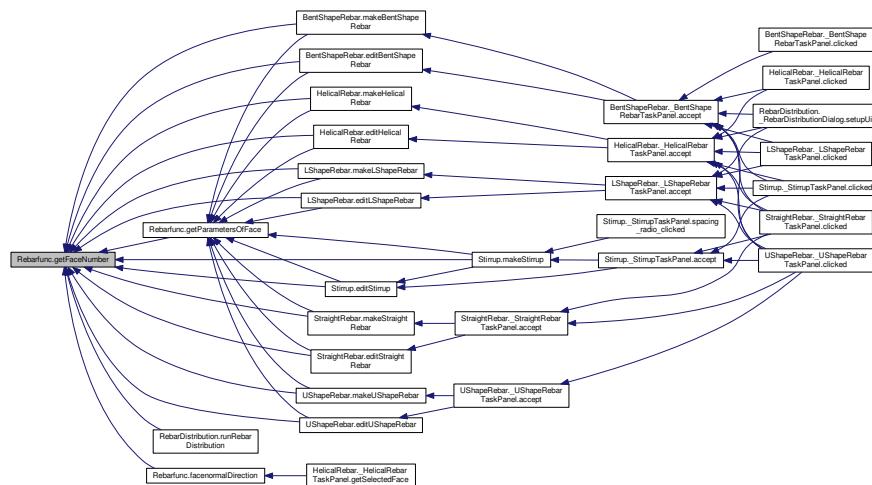
6.6.1.9 def Rebarfunc.getFaceNumber(s)

```
getFaceNumber(facename): This will return a face number from face name.
For eg.:
Input: "Face12"
Output: 12
```

Definition at line 72 of file [Rebarfunc.py](#).

```
00072 def getFaceNumber(s):
00073     """ getFaceNumber(facename): This will return a face number from face name.
00074     For eg.:
00075         Input: "Face12"
00076         Output: 12"""
00077     head = s.rstrip('0123456789')
00078     tail = s[len(head):]
00079     return int(tail)
00080
```

Here is the caller graph for this function:



6.6.1.10 def Rebarfunc.getParametersOfFace(structure, facename, sketch = True)

```
getParametersOfFace(structure, facename, sketch = True): This function will return
length, width and points of center of mass of a given face according to the sketch
value in the form of list.
```

For eg.:

Case 1: When sketch is True: We use True when we want to create rebars from sketch (planar rebars) and the sketch is strictly based on 2D so we neglected the normal axis of the face.
Output: [(FaceLength, FaceWidth), (CenterOfMassX, CenterOfMassY)]

Case 2: When sketch is False: When we want to create non-planar rebars (like stirrup) or we want to create rebar from a wire. Also for creating rebar from wire we will require three coordinates (x, y, z).
Output: [(FaceLength, FaceWidth), (CenterOfMassX, CenterOfMassY, CenterOfMassZ)]

Definition at line 126 of file [Rebarfunc.py](#).

```

00126 def getParametersOfFace(structure, facename, sketch = True):
00127     """ getParametersOfFace(structure, facename, sketch = True): This function will return
00128     length, width and points of center of mass of a given face according to the sketch
00129     value in the form of list.
00130
00131     For eg.:
00132     Case 1: When sketch is True: We use True when we want to create rebars from sketch
00133         (planar rebars) and the sketch is strictly based on 2D so we neglected the normal
00134         axis of the face.
00135         Output: [(FaceLength, FaceWidth), (CenterOfMassX, CenterOfMassY)]
00136
00137     Case 2: When sketch is False: When we want to create non-planar rebars(like stirrup)
00138         or we want to create rebar from a wire. Also for creating rebar from wire
00139         we will require three coordinates (x, y, z).
00140         Output: [(FaceLength, FaceWidth), (CenterOfMassX, CenterOfMassY, CenterOfMassZ)]"""
00141     face = structure.Shape.Faces[getFaceNumber(facename) - 1]
00142     center_of_mass = face.CenterOfMass
00143     #center_of_mass = center_of_mass.sub(getBaseStructuralObject(structure).Placement.Base)
00144     center_of_mass = center_of_mass.sub(structure.Placement.Base)
00145     Edges = []
00146     facePRM = []
00147     # When structure is cubic. It support all structure is derived from
00148     # any other object (like a sketch, wire etc).
00149     if isCubic(structure.Shape):
00150         print 423
00151         for edge in face.Edges:
00152             if not Edges:
00153                 Edges.append(edge)
00154             else:
00155                 # Checks whether similar edges is already present in Edges list
00156                 # or not.
00157                 if round((vec(edge)).Length) not in [round((vec(x)).Length) for x in Edges]:
00158                     Edges.append(edge)
00159             if len(Edges) == 1:
00160                 Edges.append(edge)
00161             # facePRM holds length of a edges.
00162             facePRM = [(vec(edge)).Length for edge in Edges]
00163             # Find the orientation of the face. Also eliminating normal axes
00164             # to the edge/face.
00165             # When edge is parallel to x-axis
00166             if round(Edges[0].tangentAt(0)[0]) in {1,-1}:
00167                 x = center_of_mass[0]
00168                 if round(Edges[1].tangentAt(0)[1]) in {1, -1}:
00169                     y = center_of_mass[1]
00170                 else:
00171                     y = center_of_mass[2]
00172             # When edge is parallel to y-axis
00173             elif round(Edges[0].tangentAt(0)[1]) in {1,-1}:
00174                 x = center_of_mass[1]
00175                 if round(Edges[1].tangentAt(0)[0]) in {1, -1}:
00176                     # Change order when edge along x-axis is at second place.
00177                     facePRM.reverse()
00178                     y = center_of_mass[1]
00179                 else:
00180                     y = center_of_mass[2]
00181             elif round(Edges[0].tangentAt(0)[2]) in {1,-1}:
00182                 y = center_of_mass[2]
00183                 if round(Edges[1].tangentAt(0)[0]) in {1, -1}:
00184                     x = center_of_mass[0]
00185                 else:
00186                     x = center_of_mass[1]
00187                 facePRM.reverse()
00188             facelength = facePRM[0]
00189             facewidth = facePRM[1]
00190
00191     # When structure is not cubic. For founding parameters of given face
00192     # I have used bounding box.
00193     else:
00194         boundbox = face.BoundingBox
00195         # Check that one length of bounding box is zero. Here bounding box
00196         # looks like a plane.
00197         if 0 in {round(boundbox.XLength), round(boundbox.YLength), round(boundbox.ZLength)}:
00198             normal = face.normalAt(0,0)
00199             normal = face.Placement.Rotation.inverted().multVec(normal)
00200             #print "x: ", boundbox.XLength
00201             #print "y: ", boundbox.YLength
00202             #print "z: ", boundbox.ZLength
00203             # Set length and width of user selected face of structural element
00204             flag = True
00205             #FIXME: Improve below logic.
00206             for i in range(len(normal)):
00207                 if round(normal[i]) == 0:
00208                     if flag and i == 0:
00209                         x = center_of_mass[i]
00210                         facelength = boundbox.XLength
00211                         flag = False
00212                     elif flag and i == 1:
00213                         x = center_of_mass[i]

```

```

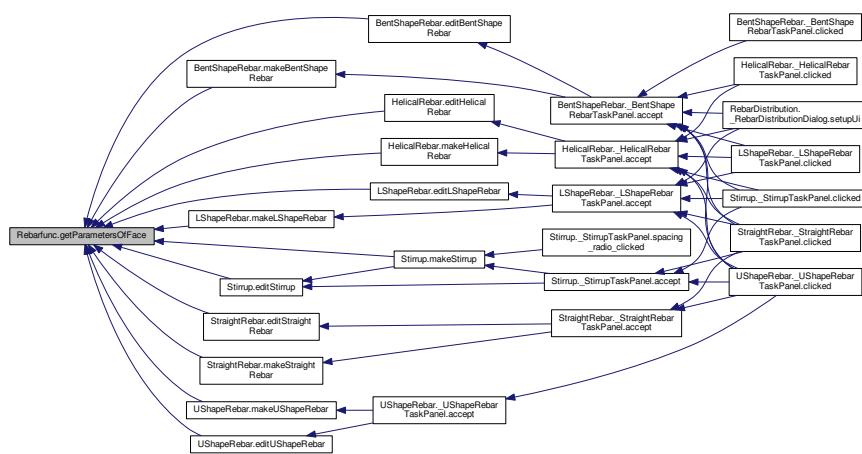
00213             facelength = boundbox.YLength
00214             flag = False
00215             if i == 1:
00216                 y = center_of_mass[i]
00217                 facewidth = boundbox.YLength
00218             elif i == 2:
00219                 y = center_of_mass[i]
00220                 facewidth = boundbox.ZLength
00221             #print [(facelength, facewidth), (x, y)]
00222             # Return parameter of the face when rebar is not created from the sketch.
00223             # For eg. non-planar rebars like stirrup etc.
00224             if not sketch:
00225                 center_of_mass = face.CenterOfMass
00226                 return [(facelength, facewidth), center_of_mass]
00227             #TODO: Add support when bounding box have depth. Here bounding box looks
00228             # like cuboid. If we given curved face.
00229             return [(facelength, facewidth), (x, y)]
00230
00231 # -----
00232 # Functions which is mainly used while creating stirrup.
00233 # -----
00234

```

Here is the call graph for this function:



Here is the caller graph for this function:



6.6.1.11 def Rebarfunc.getSelectedFace (self)

Definition at line 278 of file [Rebarfunc.py](#).

```

00278 def getSelectedFace(self):
00279     selected_objs = FreeCADGui.Selection.getSelectionEx()
00280     if selected_objs:

```

```

00281     if len(selected_objs[0].SubObjects) == 1:
00282         if "Face" in selected_objs[0].SubElementNames[0]:
00283             self.SelectedObj = selected_objs[0].Object
00284             self.FaceName = selected_objs[0].SubElementNames[0]
00285             self.form.PickSelectedFaceLabel.setText("Selected face is " + self.FaceName)
00286         else:
00287             showWarning("Select any face of the structural element.")
00288         else:
00289             showWarning("Select only one face of the structural element.")
00290     else:
00291         showWarning("Select any face of the structural element.")
00292

```

6.6.1.12 def Rebarfunc.isTrueParametersOfStructure(obj)

`getTrueParametersOfStructure(obj)`: This function return actual length, width and height of the structural element in the form of array like [Length, Width, Height]

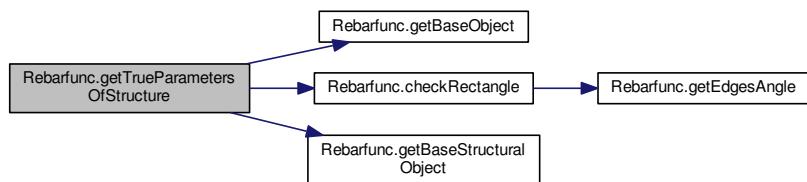
Definition at line 95 of file [Rebarfunc.py](#).

```

00095 def getTrueParametersOfStructure(obj):
00096     """ getTrueParametersOfStructure(obj): This function return actual length,
00097     width and height of the structural element in the form of array like
00098     [Length, Width, Height]"""
00099     baseObject = getBaseObject(obj)
00100     # If selected_obj is not derived from any base object
00101     if baseObject:
00102         # If selected_obj is derived from SketchObject
00103         if baseObject.isDerivedFrom("Sketcher::SketchObject"):
00104             edges = baseObject.Shape.Edges
00105             if checkRectangle(edges):
00106                 for edge in edges:
00107                     # Representation vector of edge
00108                     rep_vector = edge.Vertexes[1].Point.sub(edge.Vertexes[0].Point)
00109                     rep_vector_angle = round(math.degrees(rep_vector.getAngle(FreeCAD.Vector(1,0,0))))
00110                     if rep_vector_angle in {0, 180}:
00111                         length = edge.Length
00112                     else:
00113                         width = edge.Length
00114                 else:
00115                     return None
00116             else:
00117                 return None
00118             height = obj.Height.Value
00119         else:
00120             structuralBaseObject = getBaseStructuralObject(obj)
00121             length = structuralBaseObject.Length.Value
00122             width = structuralBaseObject.Width.Value
00123             height = structuralBaseObject.Height.Value
00124     return [length, width, height]
00125

```

Here is the call graph for this function:



6.6.1.13 def Rebarfunc.showWarning(message)

`showWarning(message)`: This function is used to produce warning message for the user.

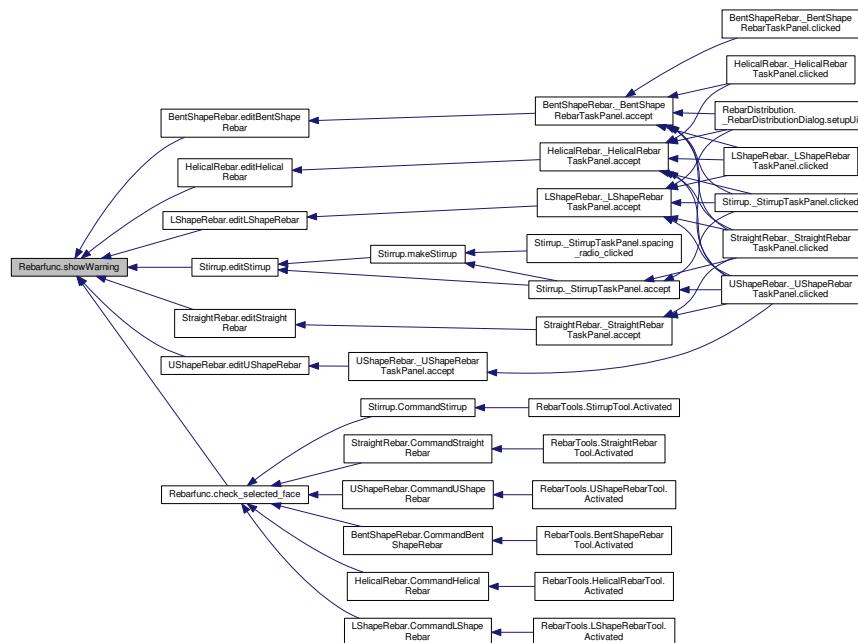
Definition at line 293 of file [Rebarfunc.py](#).

```
00293 def showWarning(message):
00294     """ showWarning(message): This function is used to produce warning
00295     message for the user."""
00296     msg = QtGui.QMessageBox()
00297     msg.setIcon(QtGui.QMessageBox.Warning)
00298     msg.setText(translate\("RebarAddon", message\)\)
00299     msg.setStandardButtons(QtGui.QMessageBox.Ok)
00300     msg.exec_()
00301
00302 # Qt translation handling
```

Here is the call graph for this function:



Here is the caller graph for this function:

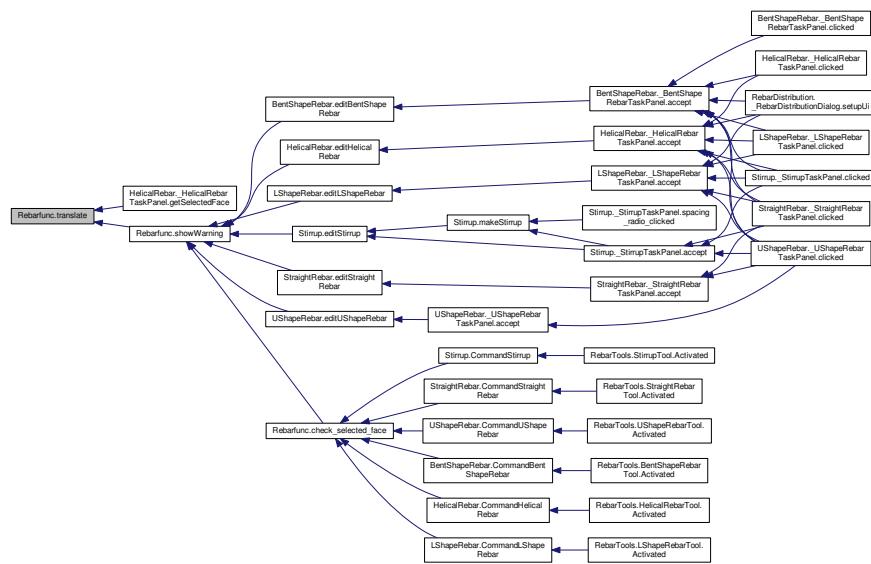


6.6.1.14 def Rebarfunc.translate (context, text, disambig=None)

Definition at line 303 of file [Rebarfunc.py](#).

```
00303 def translate(context, text, disambig=None):
00304     return QtCore.QCoreApplication.translate(context, text, disambig)
00305
```

Here is the caller graph for this function:



6.6.2.1 string Rebarfunc.__author__ = "Amritpal Singh" [private]

Definition at line 25 of file [Rebarfunc.py](#).

6.6.2.2 string Rebarfunc.__title__ = "GenericRebarFuctions" [private]

Definition at line 24 of file [Rebarfunc.py](#).

6.6.2.3 string Rebarfunc.__url__ = "https://www.freecadweb.org" [private]

Definition at line 26 of file [Rebarfunc.py](#).

6.6.2.4 Rebarfunc.FaceName

Definition at line 284 of file [Rebarfunc.py](#).

6.6.2.5 Rebarfunc.SelectedObj

Definition at line 283 of file [Rebarfunc.py](#).

6.7 RebarTools Namespace Reference

Classes

- class [BentShapeRebarTool](#)
- class [HelicalRebarTool](#)
- class [LShapeRebarTool](#)
- class [StirrupTool](#)
- class [StraightRebarTool](#)
- class [UShapeRebarTool](#)

Variables

- string [__title__](#) = "RebarCommands"
- string [__author__](#) = "Amritpal Singh"
- string [__url__](#) = "https://www.freecadweb.org"
- list [RebarCommands](#) = ["Arch_Rebar_Straight", "Arch_Rebar_UShape", "Arch_Rebar_LShape", "Arch_Rebar_Stirrup", "Arch_Rebar_BentShape", "Arch_Rebar_Helical"]

6.7.1 Variable Documentation

6.7.1.1 string RebarTools.__author__ = "Amritpal Singh" [private]

Definition at line 25 of file [RebarTools.py](#).

6.7.1.2 string RebarTools.__title__ = "RebarCommands" [private]

Definition at line 24 of file [RebarTools.py](#).

6.7.1.3 string RebarTools.__url__ = "https://www.freecadweb.org" [private]

Definition at line 26 of file [RebarTools.py](#).

6.7.1.4 list RebarTools.RebarCommands = ["Arch_Rebar_Straight", "Arch_Rebar_UShape", "Arch_Rebar_LShape", "Arch_Rebar_Stirrup", "Arch_Rebar_BentShape", "Arch_Rebar_Helical"]

Definition at line 148 of file [RebarTools.py](#).

6.8 Stirrup Namespace Reference

Classes

- class [_StirrupTaskPanel](#)

Functions

- def [getpointsOfStirrup](#) (FacePRM, l_cover, r_cover, t_cover, b_cover, bentAngle, bentFactor, diameter, rounding, facenormal)
- def [makeStirrup](#) (l_cover, r_cover, t_cover, b_cover, f_cover, bentAngle, bentFactor, diameter, rounding, amount_spacing_check, amount_spacing_value, structure=None, facename=None)
- def [editStirrup](#) (Rebar, l_cover, r_cover, t_cover, b_cover, f_cover, bentAngle, bentFactor, diameter, rounding, amount_spacing_check, amount_spacing_value, structure=None, facename=None)
- def [editDialog](#) (vobj)
- def [CommandStirrup](#) ()

Variables

- string [__title__](#) = "StirrupRebar"
- string [__author__](#) = "Amritpal Singh"
- string [__url__](#) = "https://www.freecadweb.org"

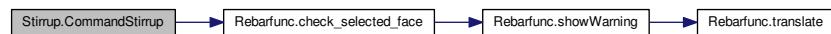
6.8.1 Function Documentation

6.8.1.1 def Stirrup.CommandStirrup ()

Definition at line 350 of file [Stirrup.py](#).

```
00350 def CommandStirrup():
00351     selected_obj = check_selected_face()
00352     if selected_obj:
00353         FreeCADGui.Control.showDialog(_StirrupTaskPanel())
00354
```

Here is the call graph for this function:



Here is the caller graph for this function:



6.8.1.2 def Stirrup.editDialog (vobj)

Definition at line 327 of file [Stirrup.py](#).

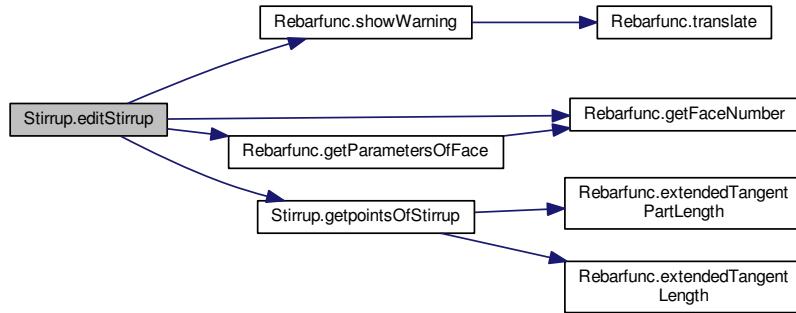
```
00327 def editDialog(vobj):
00328     FreeCADGui.Control.closeDialog()
00329     obj = _StirrupTaskPanel(vobj.Object)
00330     obj.form.frontCover.setText(str(vobj.Object.FrontCover))
00331     obj.form.l_sideCover.setText(str(vobj.Object.LeftCover))
00332     obj.form.r_sideCover.setText(str(vobj.Object.RightCover))
00333     obj.form.t_sideCover.setText(str(vobj.Object.TopCover))
00334     obj.form.b_sideCover.setText(str(vobj.Object.BottomCover))
00335     obj.form.diameter.setText(str(vobj.Object.Diameter))
00336     obj.form.bentAngle.setCurrentIndex(obj.form.bentAngle.findText(str(vobj.Object.BentAngle)))
00337     obj.form.bentFactor.setValue(vobj.Object.BentFactor)
00338     obj.form.rounding.setValue(vobj.Object.Rounding)
00339     if vobj.Object.AmountCheck:
00340         obj.form.amount.setValue(vobj.Object.Amount)
00341     else:
00342         obj.form.amount_radio.setChecked(False)
00343         obj.form.spacing_radio.setChecked(True)
00344         obj.form.amount.setDisabled(True)
00345         obj.form.spacing.setEnabled(True)
00346         obj.form.spacing.setText(str(vobj.Object.TrueSpacing))
00347 #obj.form.PickSelectedFace.setVisible(False)
00348 FreeCADGui.Control.showDialog(obj)
00349
```

6.8.1.3 def Stirrup.editStirrup (Rebar, l_cover, r_cover, t_cover, b_cover, f_cover, bentAngle, bentFactor, diameter, rounding, amount_spacing_check, amount_spacing_value, structure=None, facename=None)

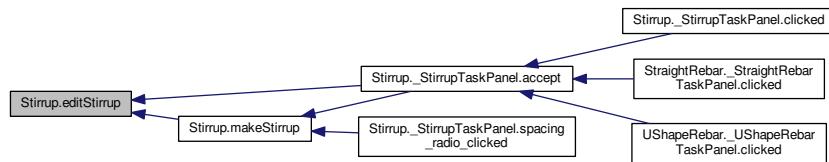
Definition at line 281 of file [Stirrup.py](#).

```
00281     amount_spacing_check, amount_spacing_value, structure = None, facename = None):
00282     sketch = Rebar.Base
00283     if structure and facename:
00284         sketch.Support = [(structure, facename)]
00285     # Check if sketch support is empty.
00286     if not sketch.Support:
00287         showWarning("You have checked remove external geometry of base sketchs when needed.\nTo
unchecked Edit->Preferences->Arch.")
00288         return
00289     # Assigned values
00290     facename = sketch.Support[0][1][0]
00291     structure = sketch.Support[0][0]
00292     face = structure.Shape.Faces[getFaceNumber(facename) - 1]
00293     #StructurePRM = getTrueParametersOfStructure(structure)
00294     # Get parameters of the face where sketch of rebar is drawn
00295     FacePRM = getParametersOfFace(structure, facename, False)
00296     FaceNormal = face.normalAt(0, 0)
00297     #FaceNormal = face.Placement.Rotation.inverted().multVec(FaceNormal)
00298     # Calculate the coordinates value of Stirrup rebar
00299     points = getPointsOfStirrup(FacePRM, l_cover, r_cover, t_cover, b_cover, bentAngle,
00300     bentFactor, diameter, rounding, FaceNormal)
00301     Rebar.Base.Points = points
00302     FreeCAD.ActiveDocument.recompute()
00303     Rebar.Direction = FaceNormal.negative()
00304     Rebar.OffsetStart = f_cover
00305     Rebar.OffsetEnd = f_cover
00306     Rebar.BentAngle = bentAngle
00307     Rebar.BentFactor = bentFactor
00308     Rebar.Rounding = rounding
00309     Rebar.Diameter = diameter
00310     if amount_spacing_check:
00311         Rebar.Amount = amount_spacing_value
00312         FreeCAD.ActiveDocument.recompute()
00313         Rebar.AmountCheck = True
00314     else:
00315         size = (ArchCommands.projectToVector(structure.Shape.copy(), face.normalAt(0, 0))).Length
00316         Rebar.Amount = int((size - diameter) / amount_spacing_value)
00317         FreeCAD.ActiveDocument.recompute()
00318         Rebar.AmountCheck = False
00319     Rebar.FrontCover = f_cover
00320     Rebar.LeftCover = l_cover
00321     Rebar.RightCover = r_cover
00322     Rebar.TopCover = t_cover
00323     Rebar.BottomCover = b_cover
00324     Rebar.TrueSpacing = amount_spacing_value
00325     FreeCAD.ActiveDocument.recompute()
00326     return Rebar
```

Here is the call graph for this function:



Here is the caller graph for this function:



6.8.1.4 def Stirrup.getpointsOfStirrup (FacePRM, l_cover, r_cover, t_cover, b_cover, bentAngle, bentFactor, diameter, rounding, facenormal)

getpointsOfStirrup(FacePRM, LeftCover, RightCover, TopCover, BottomCover, BentAngle, BentFactor, Diameter, Rounding, Facenormal)
Return the coordinates points of the Stirrup in the form of array.

Definition at line 40 of file [Stirrup.py](#).

```

00040 def getpointsOfStirrup(FacePRM, l_cover, r_cover, t_cover, b_cover, bentAngle,
                           bentFactor, diameter, rounding, facenormal):
00041     """ getpointsOfStirrup(FacePRM, LeftCover, RightCover, TopCover, BottomCover, BentAngle, BentFactor,
                           Diameter, Rounding, FaceNormal):
00042     Return the coordinates points of the Stirrup in the form of array."""
00043     angle = 180 - bentAngle
00044     tangent_part_length = extendedTangentPartLength(rounding, diameter, angle)
00045     tangent_length = extendedTangentLength(rounding, diameter, angle)
00046     if round(facenormal[0]) in {1,-1}:
00047         x1 = FacePRM[1][0]
00048         y1 = FacePRM[1][1] - FacePRM[0][0] / 2 + l_cover
00049         z1 = FacePRM[1][2] + FacePRM[0][1] / 2 - t_cover + tangent_part_length
00050         y2 = FacePRM[1][1] - FacePRM[0][0] / 2 + l_cover
00051         z2 = FacePRM[1][2] - FacePRM[0][1] / 2 + b_cover
00052         y3 = FacePRM[1][1] + FacePRM[0][0] / 2 - r_cover
00053         z3 = FacePRM[1][2] - FacePRM[0][1] / 2 + b_cover
00054         y4 = FacePRM[1][1] + FacePRM[0][0] / 2 - r_cover
00055         z4 = FacePRM[1][2] + FacePRM[0][1] / 2 - t_cover
00056         y5 = FacePRM[1][1] - FacePRM[0][0] / 2 + l_cover - tangent_part_length
00057         z5 = FacePRM[1][2] + FacePRM[0][1] / 2 - t_cover
00058         side_length = abs(y5 - y4) - tangent_part_length
00059         normal_dis = (diameter * (side_length + tangent_part_length)) / side_length
00060         x2 = x1 - normal_dis / 4

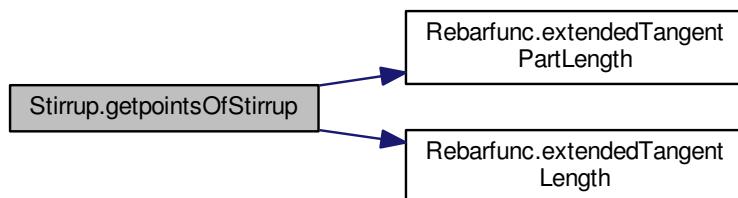
```

```

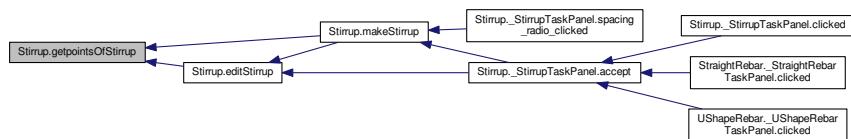
00061      x3 = x2 - normal_dis / 4
00062      x4 = x3 - normal_dis / 4
00063      x5 = x4 - normal_dis / 4
00064      x0 = x1 + normal_dis / 4
00065      y0 = y1 + (tangent_length + bentFactor * diameter) * math.sin(math.radians(angle))
00066      z0 = z1 - (tangent_length + bentFactor * diameter) * math.cos(math.radians(angle))
00067      x6 = x5 - normal_dis / 4
00068      y6 = y5 + (tangent_length + bentFactor * diameter) * math.sin(math.radians(90 - angle))
00069      z6 = z5 - (tangent_length + bentFactor * diameter) * math.cos(math.radians(90 - angle))
00070  elif round(facenormal[1]) in {1,-1}:
00071      x1 = FacePRM[1][0] - FacePRM[0][0] / 2 + l_cover
00072      y1 = FacePRM[1][1]
00073      z1 = FacePRM[1][2] + FacePRM[0][1] / 2 - t_cover + tangent_part_length
00074      x2 = FacePRM[1][0] - FacePRM[0][0] / 2 + l_cover
00075      z2 = FacePRM[1][2] - FacePRM[0][1] / 2 + b_cover
00076      x3 = FacePRM[1][0] + FacePRM[0][0] / 2 - r_cover
00077      z3 = FacePRM[1][2] - FacePRM[0][1] / 2 + b_cover
00078      x4 = FacePRM[1][0] + FacePRM[0][0] / 2 - r_cover
00079      z4 = FacePRM[1][2] + FacePRM[0][1] / 2 - t_cover
00080      x5 = FacePRM[1][0] - FacePRM[0][0] / 2 + l_cover - tangent_part_length
00081      z5 = FacePRM[1][2] + FacePRM[0][1] / 2 - t_cover
00082      side_length = abs(x5 - x4) - tangent_part_length
00083      normal_dis = (diameter * (side_length + tangent_part_length)) / side_length
00084      y2 = y1 - normal_dis / 4
00085      y3 = y2 - normal_dis / 4
00086      y4 = y3 - normal_dis / 4
00087      y5 = y4 - normal_dis / 4
00088      y0 = y1 + normal_dis / 4
00089      x0 = x1 + (tangent_length + bentFactor * diameter) * math.sin(math.radians(angle))
00090      z0 = z1 - (tangent_length + bentFactor * diameter) * math.cos(math.radians(angle))
00091      x6 = x5 + (tangent_length + bentFactor * diameter) * math.sin(math.radians(90 - angle))
00092      y6 = y5 - normal_dis / 4
00093      z6 = z5 - (tangent_length + bentFactor * diameter) * math.cos(math.radians(90 - angle))
00094  elif round(facenormal[2]) in {1,-1}:
00095      x1 = FacePRM[1][0] - FacePRM[0][0] / 2 + l_cover
00096      y1 = FacePRM[1][1] + FacePRM[0][1] / 2 - t_cover + tangent_part_length
00097      z1 = FacePRM[1][2]
00098      x2 = FacePRM[1][0] - FacePRM[0][0] / 2 + l_cover
00099      y2 = FacePRM[1][1] - FacePRM[0][1] / 2 + b_cover
00100     x3 = FacePRM[1][0] + FacePRM[0][0] / 2 - r_cover
00101     y3 = FacePRM[1][1] - FacePRM[0][1] / 2 + b_cover
00102     x4 = FacePRM[1][0] + FacePRM[0][0] / 2 - r_cover
00103     y4 = FacePRM[1][1] + FacePRM[0][1] / 2 - t_cover
00104     x5 = FacePRM[1][0] - FacePRM[0][0] / 2 + l_cover - tangent_part_length
00105     y5 = FacePRM[1][1] + FacePRM[0][1] / 2 - t_cover
00106     side_length = abs(x5 - x4) - tangent_part_length
00107     normal_dis = (diameter * (side_length + tangent_part_length)) / side_length
00108     z2 = z1 - normal_dis / 4
00109     z3 = z2 - normal_dis / 4
00110     z4 = z3 - normal_dis / 4
00111     z5 = z4 - normal_dis / 4
00112     z0 = z1 + normal_dis / 4
00113     x0 = x1 + (tangent_length + bentFactor * diameter) * math.sin(math.radians(angle))
00114     y0 = y1 - (tangent_length + bentFactor * diameter) * math.cos(math.radians(angle))
00115     x6 = x5 + (tangent_length + bentFactor * diameter) * math.sin(math.radians(90 - angle))
00116     y6 = y5 - (tangent_length + bentFactor * diameter) * math.cos(math.radians(90 - angle))
00117     z6 = z5 - normal_dis / 4
00118  return [FreeCAD.Vector(x0, y0, z0), FreeCAD.Vector(x1, y1, z1), \
00119          FreeCAD.Vector(x2, y2, z2), FreeCAD.Vector(x3, y3, z3), \
00120          FreeCAD.Vector(x4, y4, z4), FreeCAD.Vector(x5, y5, z5), \
00121          FreeCAD.Vector(x6, y6, z6)]
00122

```

Here is the call graph for this function:



Here is the caller graph for this function:



6.8.1.5 def Stirrup.makeStirrup(l_cover, r_cover, t_cover, b_cover, f_cover, bentAngle, bentFactor, diameter, rounding, amount_spacing_check, amount_spacing_value, structure =None, facename =None)

makeStirrup(LeftCover, RightCover, TopCover, BottomCover, FrontCover, BentAngle, BentFactor, Diameter, Rounding, AmountSpacingCheck, AmountSpacingValue, Structure, Facename):
Adds the Stirrup reinforcement bar to the selected structural object.

Definition at line 210 of file [Stirrup.py](#).

```

00210     amount_spacing_check, amount_spacing_value, structure = None, facename = None):
00211     """ makeStirrup(LeftCover, RightCover, TopCover, BottomCover, FrontCover, BentAngle,
00212     BentFactor, Diameter, Rounding, AmountSpacingCheck, AmountSpacingValue, Structure, Facename):
00213     Adds the Stirrup reinforcement bar to the selected structural object."""
00214     if not structure and not facename:
00215         selected_obj = FreeCADGui.Selection.getSelectionEx()[0]
00216         structure = selected_obj.Object
00217         facename = selected_obj.SubElementNames[0]
00218         face = structure.Shape.Faces[getFaceNumber(facename) - 1]
00219         #StructurePRM = getTrueParametersOfStructure(structure)
00220         FacePRM = getParametersOfFace(structure, facename, False)
00221         FaceNormal = face.normalAt(0,0)
00222         #FaceNormal = face.Placement.Rotation.inverted().multVec(FaceNormal)
00223         if not FacePRM:
00224             FreeCAD.Console.PrintError("Cannot identified shape or from which base object sturctural element is
derived\n")
00225         return
00226     # Calculate the coordinate values of Stirrup
00227     points = getpointsOfStirrup(FacePRM, l_cover, r_cover, t_cover, b_cover, bentAngle,
00228     bentFactor, diameter, rounding, FaceNormal)
00229     import Draft
00230     line = Draft.makeWire(points, closed = False, face = True, support = None)
00231     import Arch
00232     line.Support = [(structure, facename)]
00233     if amount_spacing_check:
00234         rebar = Arch.makeRebar(structure, line, diameter, amount_spacing_value, f_cover)
00235     else:
00236         size = (ArchCommands.projectToVector(structure.Shape.copy(), face.normalAt(0, 0))).Length
00237         rebar = Arch.makeRebar(structure, line, diameter,\n
00238             int((size - diameter) / amount_spacing_value), f_cover)
00239         rebar.Direction = FaceNormal.negative()
00240     rebar.Rounding = rounding
00241     # Adds properties to the rebar object
00242     rebar.ViewObject.addProperty("App::PropertyString", "RebarShape", "RebarDialog",\
00243         QT_TRANSLATE_NOOP("App::Property", "Shape of rebar")).RebarShape = "Stirrup"
00244     rebar.ViewObject.setEditorMode("RebarShape", 2)
00245     rebar.addProperty("App::PropertyDistance", "LeftCover", "RebarDialog",\
00246         QT_TRANSLATE_NOOP("App::Property", "Left Side cover of rebar")).LeftCover = l_cover
00247     rebar.setEditorMode("LeftCover", 2)
00248     rebar.addProperty("App::PropertyDistance", "RightCover", "RebarDialog",\
00249         QT_TRANSLATE_NOOP("App::Property", "Right Side cover of rebar")).RightCover = r_cover
00250     rebar.setEditorMode("RightCover", 2)
00251     rebar.addProperty("App::PropertyDistance", "TopCover", "RebarDialog",\
00252         QT_TRANSLATE_NOOP("App::Property", "Top Side cover of rebar")).TopCover = t_cover
00253     rebar.setEditorMode("TopCover", 2)
00254     rebar.addProperty("App::PropertyDistance", "BottomCover", "RebarDialog",\
00255         QT_TRANSLATE_NOOP("App::Property", "Bottom Side cover of rebar")).BottomCover = b_cover
00256     rebar.setEditorMode("BottomCover", 2)
00257     rebar.addProperty("App::PropertyDistance", "FrontCover", "RebarDialog",\
00258         QT_TRANSLATE_NOOP("App::Property", "Top cover of rebar")).FrontCover = f_cover
00259     rebar.setEditorMode("FrontCover", 2)
00260     rebar.addProperty("App::PropertyInteger", "BentAngle", "RebarDialog",\

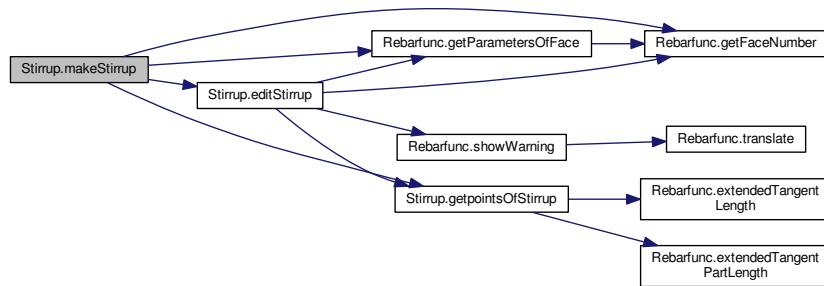
```

```

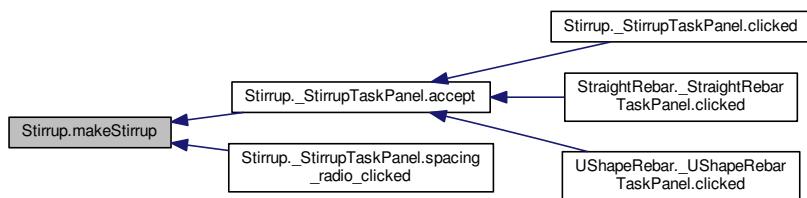
00260     QT_TRANSLATE_NOOP("App::Property", "Bent angle between at the end of rebar").BentAngle = bentAngle
00261     rebar.setEditorMode("BentAngle", 2)
00262     rebar.addProperty("App::PropertyInteger", "BentFactor", "RebarDialog", \
00263         QT_TRANSLATE_NOOP("App::Property", "Bent Length is the equal to BentFactor * Diameter")).BentFactor
00264     = bentFactor
00265     rebar.setEditorMode("BentFactor", 2)
00266     rebar.addProperty("App::PropertyBool", "AmountCheck", "RebarDialog", \
00267         QT_TRANSLATE_NOOP("App::Property", "Amount radio button is checked")).AmountCheck
00268     rebar.setEditorMode("AmountCheck", 2)
00269     rebar.addProperty("App::PropertyDistance", "TrueSpacing", "RebarDialog", \
00270         QT_TRANSLATE_NOOP("App::Property", "Spacing between of rebars")).TrueSpacing = amount_spacing_value
00271     rebar.setEditorMode("TrueSpacing", 2)
00272     if amount_spacing_check:
00273         rebar.AmountCheck = True
00274     else:
00275         rebar.AmountCheck = False
00276     rebar.Label = "Stirrup"
00277     FreeCAD.ActiveDocument.recompute()
00278     return rebar
00279

```

Here is the call graph for this function:



Here is the caller graph for this function:



6.8.2 Variable Documentation

6.8.2.1 string Stirrup.__author__ = "Amritpal Singh" [private]

Definition at line 25 of file [Stirrup.py](#).

6.8.2.2 string Stirrup.__title__ = "StirrupRebar" [private]

Definition at line 24 of file [Stirrup.py](#).

6.8.2.3 string Stirrup.__url__ = "https://www.freecadweb.org" [private]

Definition at line 26 of file [Stirrup.py](#).

6.9 StraightRebar Namespace Reference

Classes

- class [_StraightRebarTaskPanel](#)

Functions

- def [getpointsOfStraightRebar](#) (FacePRM, rt_cover, lb_cover, coverAlong, orientation)
- def [makeStraightRebar](#) (f_cover, coverAlong, rt_cover, lb_cover, diameter, amount_spacing_check, amount_spacing_value, orientation="Horizontal", structure=None, facename=None)
- def [editStraightRebar](#) (Rebar, f_cover, coverAlong, rt_cover, lb_cover, diameter, amount_spacing_check, amount_spacing_value, orientation, structure=None, facename=None)
- def [editDialog](#) (vobj)
- def [CommandStraightRebar](#) ()

Variables

- string [__title__](#) = "StraightRebar"
- string [__author__](#) = "Amritpal Singh"
- string [__url__](#) = "https://www.freecadweb.org"

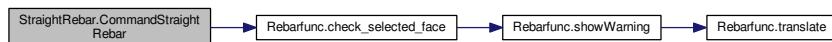
6.9.1 Function Documentation

6.9.1.1 def StraightRebar.CommandStraightRebar ()

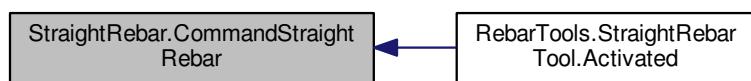
Definition at line 320 of file [StraightRebar.py](#).

```
00320 def CommandStraightRebar():
00321     selected_obj = check_selected_face()
00322     if selected_obj:
00323         FreeCADGui.Control.showDialog(_StraightRebarTaskPanel())
00324
```

Here is the call graph for this function:



Here is the caller graph for this function:



6.9.1.2 def StraightRebar.editDialog (*vobj*)

Definition at line 299 of file [StraightRebar.py](#).

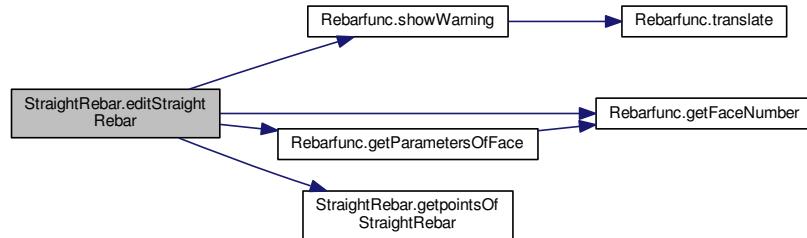
```
00299 def editDialog(vobj):
00300     FreeCADGui.Control.closeDialog()
00301     obj = _StraightRebarTaskPanel(vobj.Object)
00302     obj.form.frontCover.setText(str(vobj.Object.FrontCover))
00303     obj.form.r_sideCover.setText(str(vobj.Object.RightTopCover))
00304     obj.form.l_sideCover.setText(str(vobj.Object.LeftBottomCover))
00305     obj.form.bottomCover.setText(str(vobj.Object.Cover))
00306     obj.form.diameter.setText(str(vobj.Object.Diameter))
00307     obj.form.orientation.setCurrentIndex(obj.form.orientation.findText(str(vobj.Object.Orientation)))
00308     obj.form.coverAlong.setCurrentIndex(obj.form.coverAlong.findText(str(vobj.Object.CoverAlong)))
00309     if vobj.Object.AmountCheck:
00310         obj.form.amount.setValue(vobj.Object.Amount)
00311     else:
00312         obj.form.amount_radio.setChecked(False)
00313         obj.form.spacing_radio.setChecked(True)
00314         obj.form.amount.setDisabled(True)
00315         obj.form.spacing.setEnabled(True)
00316         obj.form.spacing.setText(str(vobj.Object.TrueSpacing))
00317     #obj.form.PickSelectedFace.setVisible(False)
00318     FreeCADGui.Control.showDialog(obj)
00319
```

6.9.1.3 def StraightRebar.editStraightRebar (*Rebar*, *f_cover*, *coverAlong*, *rt_cover*, *lb_cover*, *diameter*,
amount_spacing_check, *amount_spacing_value*, *orientation*, *structure* =None, *facename* =None)

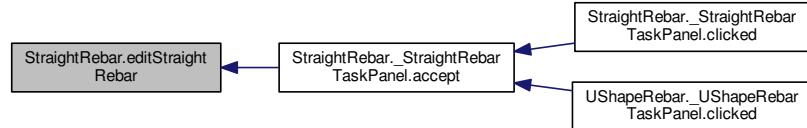
Definition at line 255 of file [StraightRebar.py](#).

```
00255 def editStraightRebar(Rebar, f_cover, coverAlong, rt_cover, lb_cover, diameter,
                           amount_spacing_check, amount_spacing_value, orientation, structure = None, facename = None):
00256     sketch = Rebar.Base
00257     if structure and facename:
00258         sketch.Support = [(structure, facename)]
00259         FreeCAD.ActiveDocument.recompute()
00260     # Check if sketch support is empty.
00261     if not sketch.Support:
00262         showWarning("You have checked remove external geometry of base sketchs when needed.\nTo
unchecked Edit->Preferences->Arch.")
00263         return
00264     # Assigned values
00265     facename = sketch.Support[0][1][0]
00266     structure = sketch.Support[0][0]
00267     face = structure.Shape.Faces[getFaceNumber(facename) - 1]
00268     #StructurePRM = getTrueParametersOfStructure(structure)
00269     # Get parameters of the face where sketch of rebar is drawn
00270     FacePRM = getParametersOfFace(structure, facename)
00271     # Get points of Striaght rebar
00272     points = getpointsOfStraightRebar(FacePRM, rt_cover, lb_cover, coverAlong,
00273                                         orientation)
00273     sketch.movePoint(0, 1, points[0], 0)
00274     FreeCAD.ActiveDocument.recompute()
00275     sketch.movePoint(0, 2, points[1], 0)
00276     FreeCAD.ActiveDocument.recompute()
00277     Rebar.OffsetStart = f_cover
00278     Rebar.OffsetEnd = f_cover
00279     if amount_spacing_check:
00280         Rebar.Amount = amount_spacing_value
00281         FreeCAD.ActiveDocument.recompute()
00282         Rebar.AmountCheck = True
00283     else:
00284         size = (ArchCommands.projectToVector(structure.Shape.copy(), face.normalAt(0, 0))).Length
00285         Rebar.Amount = int((size - diameter) / amount_spacing_value)
00286         FreeCAD.ActiveDocument.recompute()
00287         Rebar.AmountCheck = False
00288     Rebar.FrontCover = f_cover
00289     Rebar.RightTopCover = rt_cover
00290     Rebar.LeftBottomCover = lb_cover
00291     Rebar.CoverAlong = coverAlong[0]
00292     Rebar.Cover = coverAlong[1]
00293     Rebar.TrueSpacing = amount_spacing_value
00294     Rebar.Diameter = diameter
00295     Rebar.Orientation = orientation
00296     FreeCAD.ActiveDocument.recompute()
00297     return Rebar
00298
```

Here is the call graph for this function:



Here is the caller graph for this function:



6.9.1.4 def StraightRebar.getpointsOfStraightRebar (FacePRM, rt_cover, lb_cover, coverAlong, orientation)

getpointsOfStraightRebar(FacePRM, RightTopcover, LeftBottomcover, CoverAlong, Orientation):
Return points of the Straight rebar in the form of array for sketch.

Case I: When Orientation is 'Horizontal':

We have two option in CoverAlong i.e. 'Bottom Side' or 'Top Side'

Case II: When Orientation is 'Vertical':

We have two option in CoverAlong i.e. 'Left Side' or 'Right Side'

Definition at line 40 of file [StraightRebar.py](#).

```

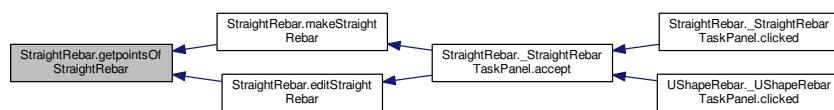
00040 def getpointsOfStraightRebar(FacePRM, rt_cover, lb_cover, coverAlong, orientation):
00041     """ getpointsOfStraightRebar(FacePRM, RightTopcover, LeftBottomcover, CoverAlong, Orientation):
00042         Return points of the Straight rebar in the form of array for sketch.
00043
00044     Case I: When Orientation is 'Horizontal':
00045         We have two option in CoverAlong i.e. 'Bottom Side' or 'Top Side'
00046     Case II: When Orientation is 'Vertical':
00047         We have two option in CoverAlong i.e. 'Left Side' or 'Right Side'
00048
00049     """
00050     if orientation == "Horizontal":
00051         if coverAlong[0] == "Bottom Side":
00052             x1 = FacePRM[1][0] - FacePRM[0][0] / 2 + lb_cover
00053             y1 = FacePRM[1][1] - FacePRM[0][1] / 2 + coverAlong[1]
00054             x2 = FacePRM[1][0] - FacePRM[0][0] / 2 + FacePRM[0][0] - rt_cover
00055             y2 = FacePRM[1][1] - FacePRM[0][1] / 2 + coverAlong[1]
00056         elif coverAlong[0] == "Top Side":
00057             cover = FacePRM[0][1] - coverAlong[1]
00058             x1 = FacePRM[1][0] - FacePRM[0][0] / 2 + lb_cover
00059             y1 = FacePRM[1][1] - FacePRM[0][1] / 2 + cover
00060             x2 = FacePRM[1][0] - FacePRM[0][0] / 2 + FacePRM[0][0] - rt_cover
00061             y2 = FacePRM[1][1] - FacePRM[0][1] / 2 + cover
00062     elif orientation == "Vertical":
00063         if coverAlong[0] == "Left Side":
  
```

```

00063     x1 = FacePRM[1][0] - FacePRM[0][0] / 2 + coverAlong[1]
00064     y1 = FacePRM[1][1] - FacePRM[0][1] / 2 + lb_cover
00065     x2 = FacePRM[1][0] - FacePRM[0][0] / 2 + coverAlong[1]
00066     y2 = FacePRM[1][1] - FacePRM[0][1] / 2 + FacePRM[0][1] - rt_cover
00067     elif coverAlong[0] == "Right Side":
00068         cover = FacePRM[0][0] - coverAlong[1]
00069         x1 = FacePRM[1][0] - FacePRM[0][0] / 2 + cover
00070         y1 = FacePRM[1][1] - FacePRM[0][1] / 2 + lb_cover
00071         x2 = FacePRM[1][0] - FacePRM[0][0] / 2 + cover
00072         y2 = FacePRM[1][1] - FacePRM[0][1] / 2 + FacePRM[0][1] - rt_cover
00073     return [FreeCAD.Vector(x1, y1, 0), FreeCAD.Vector(x2, y2, 0)]
00074

```

Here is the caller graph for this function:



6.9.1.5 def StraightRebar.makeStraightRebar(*f_cover, coverAlong, rt_cover, lb_cover, diameter, amount_spacing_check, amount_spacing_value, orientation = "Horizontal", structure =None, facename =None*)

Adds the straight reinforcement bar to the selected structural object.

Case I: When orientation of straight rebar is 'Horizontal':

```
makeStraightRebar(FrontCover, CoverAlong, RightCover, LeftCover, Diameter, AmountSpacingCheck, AmountSpacingValue, Orientation = "Horizontal", Structure, Facename)
```

Note: Type of CoverAlong argument is a tuple. Syntax: (<Along>, <Value>). Here we have horizontal orientation and Bottom Side to <Along> arguments.

For eg. ("Top Side", 20) and ("Bottom Side", 20)

Case II: When orientation of straight rebar is 'Vertical':

```
makeStraightRebar(FrontCover, CoverAlong, TopCover, BottomCover, Diameter, AmountSpacingCheck, AmountSpacingValue, Orientation = "Vertical", Structure, Facename)
```

Note: Type of CoverAlong argument is a tuple. Syntax: (<Along>, <Value>). Here we have vertical orientation and Right Side to <Along> arguments.

For eg. ("Left Side", 20) and ("Right Side", 20)

Definition at line 185 of file [StraightRebar.py](#).

```

00185 def makeStraightRebar(f_cover, coverAlong, rt_cover, lb_cover, diameter,
00186     amount_spacing_check, amount_spacing_value, orientation = "Horizontal", structure = None, facename = None):
00187     """ Adds the straight reinforcement bar to the selected structural object.
00188
00189     Case I: When orientation of straight rebar is 'Horizontal':
00190         makeStraightRebar(FrontCover, CoverAlong, RightCover, LeftCover, Diameter, AmountSpacingCheck,
00191             AmountSpacingValue, Orientation = "Horizontal",
00192             Structure, Facename)
00193         Note: Type of CoverAlong argument is a tuple. Syntax: (<Along>, <Value>). Here we have horizontal
00194         orientation so we can pass Top Side
00195         and Bottom Side to <Along> arguments.
00196         For eg. ("Top Side", 20) and ("Bottom Side", 20)
00197
00198     Case II: When orientation of straight rebar is 'Vertical':
00199         makeStraightRebar(FrontCover, CoverAlong, TopCover, BottomCover, Diameter, AmountSpacingCheck,
00200             AmountSpacingValue, Orientation = "Vertical",
00201             Structure, Facename)
00202         Note: Type of CoverAlong argument is a tuple. Syntax: (<Along>, <Value>). Here we have vertical
00203         orientation so we can pass Left Side
00204         and Right Side to <Along> arguments.
00205         For eg. ("Left Side", 20) and ("Right Side", 20)
00206
00207     if not structure and not facename:
00208         selected_obj = FreeCADGui.Selection.getSelectionEx()[0]

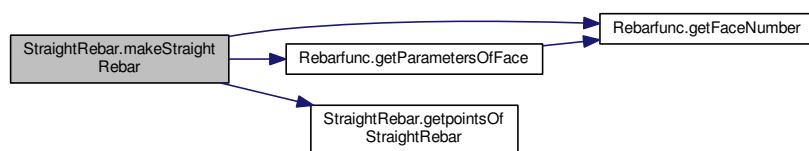
```

```

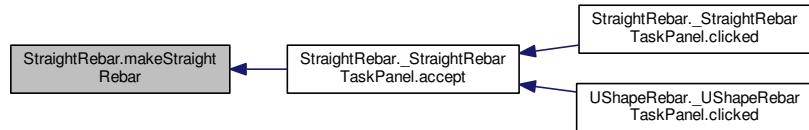
00204     structure = selected_obj.Object
00205     facename = selected_obj.SubElementNames[0]
00206     face = structure.Shape.Faces[getFaceNumber(facename) - 1]
00207     #StructurePRM = getTrueParametersOfStructure(structure)
00208     FacePRM = getParametersOfFace(structure, facename)
00209     if not FacePRM:
00210         FreeCAD.Console.PrintError('Cannot identified shape or from which base object sturctural element is
00211         derived\n')
00212         return
00213     # Get points of Striaght rebar
00214     points = getpointsOfStraightRebar(FacePRM, rt_cover, lb_cover, coverAlong,
00215     orientation)
00216     import Part
00217     import Arch
00218     sketch = FreeCAD.activeDocument().addObject('Sketcher::SketchObject', 'Sketch')
00219     sketch.MapMode = "FlatFace"
00220     sketch.Support = [(structure, facename)]
00221     FreeCAD.ActiveDocument.recompute()
00222     sketch.addGeometry(Part.LineSegment(points[0], points[1]), False)
00223     if amount_spacing_check:
00224         rebar = Arch.makeRebar(structure, sketch, diameter, amount_spacing_value, f_cover)
00225         FreeCAD.ActiveDocument.recompute()
00226     else:
00227         size = (ArchCommands.projectToVector(structure.Shape.copy(), face.normalAt(0, 0))).Length
00228         rebar = Arch.makeRebar(structure, sketch, diameter, int((size - diameter) / amount_spacing_value),
00229         f_cover)
00230         # Adds properties to the rebar object
00231         rebar.ViewObject.addProperty("App::PropertyString", "RebarShape", "RebarDialog", QT_TRANSLATE_NOOP("App::Property",
00232         "Shape of rebar")).RebarShape = "StraightRebar"
00233         rebar.ViewObject.setEditorMode("RebarShape", 2)
00234         rebar.addProperty("App::PropertyDistance", "FrontCover", "RebarDialog", QT_TRANSLATE_NOOP("App::Property",
00235         "Front cover of rebar")).FrontCover = f_cover
00236         rebar.setEditorMode("FrontCover", 2)
00237         rebar.addProperty("App::PropertyDistance", "RightTopCover", "RebarDialog", QT_TRANSLATE_NOOP("App::Property",
00238         "Right/Top Side cover of rebar")).RightTopCover = rt_cover
00239         rebar.setEditorMode("RightTopCover", 2)
00240         rebar.addProperty("App::PropertyDistance", "LeftBottomCover", "RebarDialog", QT_TRANSLATE_NOOP("App::Property",
00241         "Left/Bottom Side cover of rebar")).LeftBottomCover = lb_cover
00242         rebar.setEditorMode("LeftBottomCover", 2)
00243         rebar.addProperty("App::PropertyString", "CoverAlong", "RebarDialog", QT_TRANSLATE_NOOP("App::Property",
00244         "Cover along")).CoverAlong = coverAlong[0]
00245         rebar.setEditorMode("CoverAlong", 2)
00246         rebar.addProperty("App::PropertyDistance", "Cover", "RebarDialog", QT_TRANSLATE_NOOP("App::Property",
00247         "Cover of rebar along user selected side")).Cover = coverAlong[1]
00248         rebar.setEditorMode("Cover", 2)
00249         rebar.addProperty("App::PropertyBool", "AmountCheck", "RebarDialog", QT_TRANSLATE_NOOP("App::Property",
00250         "Amount radio button is checked")).AmountCheck
00251         rebar.setEditorMode("AmountCheck", 2)
00252         rebar.addProperty("App::PropertyDistance", "TrueSpacing", "RebarDialog", QT_TRANSLATE_NOOP("App::Property",
00253         "Spacing between of rebars")).TrueSpacing = amount_spacing_value
00254         rebar.setEditorMode("TrueSpacing", 2)
00255         rebar.addProperty("App::PropertyString", "Orientation", "RebarDialog", QT_TRANSLATE_NOOP("App::Property",
00256         "Shape of rebar")).Orientation = orientation
00257         rebar.setEditorMode("Orientation", 2)
00258         if amount_spacing_check:
00259             rebar.AmountCheck = True
00260         else:
00261             rebar.AmountCheck = False
00262             rebar.TrueSpacing = amount_spacing_value
00263             rebar.Label = "StraightRebar"
00264             FreeCAD.ActiveDocument.recompute()
00265             return rebar
00266

```

Here is the call graph for this function:



Here is the caller graph for this function:



6.9.2 Variable Documentation

6.9.2.1 string StraightRebar.__author__ = "Amritpal Singh" [private]

Definition at line 25 of file [StraightRebar.py](#).

6.9.2.2 string StraightRebar.__title__ = "StraightRebar" [private]

Definition at line 24 of file [StraightRebar.py](#).

6.9.2.3 string StraightRebar.__url__ = "https://www.freecadweb.org" [private]

Definition at line 26 of file [StraightRebar.py](#).

6.10 UShapeRebar Namespace Reference

Classes

- class [_UShapeRebarTaskPanel](#)

Functions

- def [getpointsOfUShapeRebar](#) (FacePRM, r_cover, l_cover, b_cover, t_cover, orientation)
- def [makeUShapeRebar](#) (f_cover, b_cover, r_cover, l_cover, diameter, t_cover, rounding, amount_spacing_check, amount_spacing_value, orientation="Bottom", structure=None, facename=None)
- def [editUShapeRebar](#) (Rebar, f_cover, b_cover, r_cover, l_cover, diameter, t_cover, rounding, amount_spacing_check, amount_spacing_value, orientation, structure=None, facename=None)
- def [editDialog](#) (vobj)
- def [CommandUShapeRebar](#) ()

Variables

- string [__title__](#) = "UShapeRebar"
- string [__author__](#) = "Amritpal Singh"
- string [__url__](#) = "https://www.freecadweb.org"

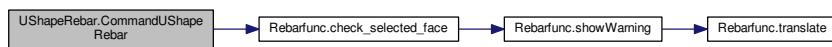
6.10.1 Function Documentation

6.10.1.1 def UShapeRebar.CommandUShapeRebar()

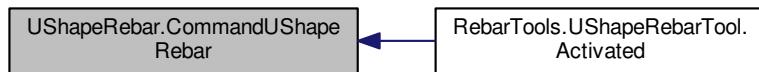
Definition at line 313 of file [UShapeRebar.py](#).

```
00313 def CommandUShapeRebar():
00314     selected_obj = check_selected_face()
00315     if selected_obj:
00316         FreeCADGui.Control.showDialog(_UShapeRebarTaskPanel())
00317
```

Here is the call graph for this function:



Here is the caller graph for this function:



6.10.1.2 def UShapeRebar.editDialog(vobj)

Definition at line 291 of file [UShapeRebar.py](#).

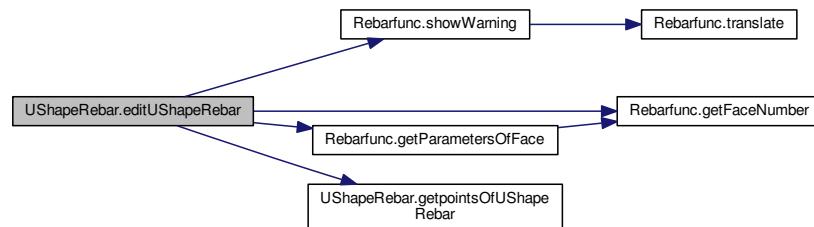
```
00291 def editDialog(vobj):
00292     FreeCADGui.Control.closeDialog()
00293     obj = _UShapeRebarTaskPanel(vobj.Object)
00294     obj.form.frontCover.setText(str(vobj.Object.FrontCover))
00295     obj.form.r_sideCover.setText(str(vobj.Object.RightCover))
00296     obj.form.l_sideCover.setText(str(vobj.Object.LeftCover))
00297     obj.form.bottomCover.setText(str(vobj.Object.BottomCover))
00298     obj.form.diameter.setText(str(vobj.Object.Diameter))
00299     obj.form.topCover.setText(str(vobj.Object.TopCover))
00300     obj.form.rounding.setValue(vobj.Object.Rounding)
00301     obj.form.orientation.setCurrentIndex(obj.form.orientation.findText(str(vobj.Object.Orientation)))
00302     if vobj.Object.AmountCheck:
00303         obj.form.amount.setValue(vobj.Object.Amount)
00304     else:
00305         obj.form.amount_radio.setChecked(False)
00306         obj.form.spacing_radio.setChecked(True)
00307         obj.form.amount.setDisabled(True)
00308         obj.form.spacing.setEnabled(True)
00309         obj.form.spacing.setText(str(vobj.Object.TrueSpacing))
00310     #obj.form.PickSelectedFace.setVisible(False)
00311     FreeCADGui.Control.showDialog(obj)
00312
```

```
6.10.1.3 def UShapeRebar.editUShapeRebar ( Rebar, f_cover, b_cover, r_cover, l_cover, diameter, t_cover, rounding,
amount_spacing_check, amount_spacing_value, orientation, structure =None, facename =None )
```

Definition at line 239 of file [UShapeRebar.py](#).

```
00239 def editUShapeRebar(Rebar, f_cover, b_cover, r_cover, l_cover, diameter, t_cover, rounding,
amount_spacing_check, amount_spacing_value, orientation, structure = None, facename = None):
00240     sketch = Rebar.Base
00241     if structure and facename:
00242         sketch.Support = [(structure, facename)]
00243     # Check if sketch support is empty.
00244     if not sketch.Support:
00245         showWarning("You have checked remove external geometry of base sketchs when needed.\nTo
unchecked Edit->Preferences->Arch.")
00246         return
00247     # Assigned values
00248     facename = sketch.Support[0][1][0]
00249     structure = sketch.Support[0][0]
00250     face = structure.Shape.Faces[getFaceNumber(facename) - 1]
00251     #StructurePRM = getTrueParametersOfStructure(structure)
00252     # Get parameters of the face where sketch of rebar is drawn
00253     FacePRM = getParametersOfFace(structure, facename)
00254     # Get points of U-Shape rebar
00255     points = getpointsOfUShapeRebar(FacePRM, r_cover, l_cover, b_cover, t_cover,
orientation)
00256     sketch.movePoint(0, 1, points[0], 0)
00257     FreeCAD.ActiveDocument.recompute()
00258     sketch.movePoint(0, 2, points[1], 0)
00259     FreeCAD.ActiveDocument.recompute()
00260     sketch.movePoint(1, 1, points[1], 0)
00261     FreeCAD.ActiveDocument.recompute()
00262     sketch.movePoint(1, 2, points[2], 0)
00263     FreeCAD.ActiveDocument.recompute()
00264     sketch.movePoint(2, 1, points[2], 0)
00265     FreeCAD.ActiveDocument.recompute()
00266     sketch.movePoint(2, 2, points[3], 0)
00267     FreeCAD.ActiveDocument.recompute()
00268     Rebar.OffsetStart = f_cover
00269     Rebar.OffsetEnd = f_cover
00270     if amount_spacing_check:
00271         Rebar.Amount = amount_spacing_value
00272         FreeCAD.ActiveDocument.recompute()
00273         Rebar.AmountCheck = True
00274     else:
00275         size = (ArchCommands.projectToVector(structure.Shape.copy(), face.normalAt(0, 0))).Length
00276         Rebar.Amount = int((size - diameter) / amount_spacing_value)
00277         FreeCAD.ActiveDocument.recompute()
00278         Rebar.AmountCheck = False
00279     Rebar.Diameter = diameter
00280     Rebar.FrontCover = f_cover
00281     Rebar.RightCover = r_cover
00282     Rebar.LeftCover = l_cover
00283     Rebar.BottomCover = b_cover
00284     Rebar.TopCover = t_cover
00285     Rebar.Rounding = rounding
00286     Rebar.TrueSpacing = amount_spacing_value
00287     Rebar.Orientation = orientation
00288     FreeCAD.ActiveDocument.recompute()
00289     return Rebar
00290
```

Here is the call graph for this function:



Here is the caller graph for this function:



6.10.1.4 def UShapeRebar.getpointsOfUShapeRebar(FacePRM, r_cover, l_cover, b_cover, t_cover, orientation)

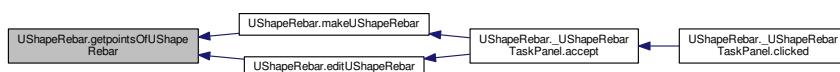
getpointsOfUShapeRebar(FacePRM, RightCover, LeftCover, BottomCover, TopCover, Orientation):
Return points of the UShape rebar in the form of array for sketch.
It takes four different orientations input i.e. 'Bottom', 'Top', 'Left', 'Right'.

Definition at line 40 of file [UShapeRebar.py](#).

```

00040 def getpointsOfUShapeRebar(FacePRM, r_cover, l_cover, b_cover, t_cover, orientation):
00041     """ getpointsOfUShapeRebar(FacePRM, RightCover, LeftCover, BottomCover, TopCover, Orientation):
00042     Return points of the UShape rebar in the form of array for sketch.
00043     It takes four different orientations input i.e. 'Bottom', 'Top', 'Left', 'Right'.
00044     """
00045     if orientation == "Bottom":
00046         x1 = FacePRM[1][0] - FacePRM[0][0] / 2 + l_cover
00047         y1 = FacePRM[1][1] + FacePRM[0][1] / 2 - t_cover
00048         x2 = FacePRM[1][0] - FacePRM[0][0] / 2 + l_cover
00049         y2 = FacePRM[1][1] - FacePRM[0][1] / 2 + b_cover
00050         x3 = FacePRM[1][0] - FacePRM[0][0] / 2 + FacePRM[0][0] - r_cover
00051         y3 = FacePRM[1][1] - FacePRM[0][1] / 2 + b_cover
00052         x4 = FacePRM[1][0] - FacePRM[0][0] / 2 + FacePRM[0][0] - r_cover
00053         y4 = FacePRM[1][1] + FacePRM[0][1] / 2 - t_cover
00054     elif orientation == "Top":
00055         x1 = FacePRM[1][0] - FacePRM[0][0] / 2 + l_cover
00056         y1 = FacePRM[1][1] - FacePRM[0][1] / 2 + b_cover
00057         x2 = FacePRM[1][0] - FacePRM[0][0] / 2 + l_cover
00058         y2 = FacePRM[1][1] + FacePRM[0][1] / 2 - t_cover
00059         x3 = FacePRM[1][0] - FacePRM[0][0] / 2 + FacePRM[0][0] - r_cover
00060         y3 = FacePRM[1][1] + FacePRM[0][1] / 2 - t_cover
00061         x4 = FacePRM[1][0] - FacePRM[0][0] / 2 + FacePRM[0][0] - r_cover
00062         y4 = FacePRM[1][1] - FacePRM[0][1] / 2 + b_cover
00063     elif orientation == "Left":
00064         x1 = FacePRM[1][0] - FacePRM[0][0] / 2 + FacePRM[0][0] - r_cover
00065         y1 = FacePRM[1][1] + FacePRM[0][1] / 2 - t_cover
00066         x2 = FacePRM[1][0] - FacePRM[0][0] / 2 + l_cover
00067         y2 = FacePRM[1][1] + FacePRM[0][1] / 2 - t_cover
00068         x3 = FacePRM[1][0] - FacePRM[0][0] / 2 + l_cover
00069         y3 = FacePRM[1][1] - FacePRM[0][1] / 2 + b_cover
00070         x4 = FacePRM[1][0] - FacePRM[0][0] / 2 + FacePRM[0][0] - r_cover
00071         y4 = FacePRM[1][1] - FacePRM[0][1] / 2 + b_cover
00072     elif orientation == "Right":
00073         x1 = FacePRM[1][0] - FacePRM[0][0] / 2 + l_cover
00074         y1 = FacePRM[1][1] + FacePRM[0][1] / 2 - t_cover
00075         x2 = FacePRM[1][0] - FacePRM[0][0] / 2 + FacePRM[0][0] - r_cover
00076         y2 = FacePRM[1][1] + FacePRM[0][1] / 2 - t_cover
00077         x3 = FacePRM[1][0] - FacePRM[0][0] / 2 + FacePRM[0][0] - r_cover
00078         y3 = FacePRM[1][1] - FacePRM[0][1] / 2 + b_cover
00079         x4 = FacePRM[1][0] - FacePRM[0][0] / 2 + l_cover
00080         y4 = FacePRM[1][1] - FacePRM[0][1] / 2 + b_cover
00081     return [FreeCAD.Vector(x1, y1, 0), FreeCAD.Vector(x2, y2, 0), \
00082             FreeCAD.Vector(x3, y3, 0), FreeCAD.Vector(x4, y4, 0)]
00083
  
```

Here is the caller graph for this function:



```
6.10.1.5 def UShapeRebar.makeUShapeRebar( f_cover, b_cover, r_cover, l_cover, diameter, t_cover, rounding,
                                         amount_spacing_check, amount_spacing_value, orientation = "Bottom", structure = None, facename =
                                         None )
```

`makeUShapeRebar(FrontCover, BottomCover, RightCover, LeftCover, Diameter, Topcover, Rounding, AmountSpacingCheck, Orientation, Structure, Facename):` Adds the U-Shape reinforcement bar to the selected structural object. It takes four different types of orientations as input i.e 'Bottom', 'Top', 'Right', 'Left'.

Definition at line 177 of file [UShapeRebar.py](#).

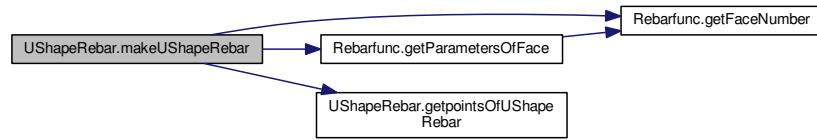
```
00177 def makeUShapeRebar(f_cover, b_cover, r_cover, l_cover, diameter, t_cover, rounding,
00178     amount_spacing_check, amount_spacing_value, orientation = "Bottom", structure = None, facename = None):
00179     """ makeUShapeRebar(FrontCover, BottomCover, RightCover, LeftCover, Diameter, Topcover, Rounding,
00180     AmountSpacingCheck, AmountSpacingValue,
00181     Orientation, Structure, Facename): Adds the U-Shape reinforcement bar to the selected structural
00182     object.
00183     It takes four different types of orientations as input i.e 'Bottom', 'Top', 'Right', 'Left'.
00184     """
00185     if not structure and not facename:
00186         selected_obj = FreeCADGui.Selection.getSelectionEx()[0]
00187         structure = selected_obj.Object
00188         facename = selected_obj.SubElementNames[0]
00189         face = structure.Shape.Faces[getFaceNumber(facename) - 1]
00190         #StructurePRM = getTrueParametersOfStructure(structure)
00191         FacePRM = getParametersOfFace(structure, facename)
00192         if not FacePRM:
00193             FreeCAD.Console.PrintError("Cannot identified shape or from which base object sturctural element is
00194             derived\n")
00195             return
00196         # Get points of U-Shape rebar
00197         points = getpointsOfUShapeRebar(FacePRM, r_cover, l_cover, b_cover, t_cover,
00198             orientation)
00199         import Part
00200         import Arch
00201         sketch = FreeCAD.activeDocument().addObject('Sketcher::SketchObject', 'Sketch')
00202         sketch.MapMode = "FlatFace"
00203         sketch.Support = [(structure, facename)]
00204         FreeCAD.ActiveDocument.recompute()
00205         sketch.addGeometry(Part.LineSegment(points[0], points[1]), False)
00206         sketch.addGeometry(Part.LineSegment(points[1], points[2]), False)
00207         import Sketcher
00208         sketch.addGeometry(Part.LineSegment(points[2], points[3]), False)
00209         if amount_spacing_check:
00210             rebar = Arch.makeRebar(structure, sketch, diameter, amount_spacing_value, f_cover)
00211             FreeCAD.ActiveDocument.recompute()
00212         else:
00213             size = (ArchCommands.projectToVector(structure.Shape.copy(), face.normalAt(0, 0))).Length
00214             rebar = Arch.makeRebar(structure, sketch, diameter, int((size - diameter) / amount_spacing_value),
00215             f_cover)
00216             rebar.Rounding = rounding
00217             # Adds properties to the rebar object
00218             rebar.ViewObject.addProperty("App::PropertyString", "RebarShape", "RebarDialog", QT_TRANSLATE_NOOP("App::Property", "Shape of rebar")).RebarShape = "UShapeRebar"
00219             rebar.ViewObject.setEditorMode("RebarShape", 2)
00220             rebar.addProperty("App::PropertyDistance", "FrontCover", "RebarDialog", QT_TRANSLATE_NOOP("App::Property", "Front cover of rebar")).FrontCover = f_cover
00221             rebar.setEditorMode("FrontCover", 2)
00222             rebar.addProperty("App::PropertyDistance", "RightCover", "RebarDialog", QT_TRANSLATE_NOOP("App::Property", "Right Side cover of rebar")).RightCover = r_cover
00223             rebar.setEditorMode("RightCover", 2)
00224             rebar.addProperty("App::PropertyDistance", "LeftCover", "RebarDialog", QT_TRANSLATE_NOOP("App::Property", "Left Side cover of rebar")).LeftCover = l_cover
00225             rebar.setEditorMode("LeftCover", 2)
00226             rebar.addProperty("App::PropertyDistance", "BottomCover", "RebarDialog", QT_TRANSLATE_NOOP("App::Property", "Bottom cover of rebar")).BottomCover = b_cover
00227             rebar.setEditorMode("BottomCover", 2)
00228             rebar.addProperty("App::PropertyBool", "AmountCheck", "RebarDialog", QT_TRANSLATE_NOOP("App::Property", "Amount radio button is checked")).AmountCheck
00229             rebar.setEditorMode("AmountCheck", 2)
00230             rebar.addProperty("App::PropertyDistance", "TopCover", "RebarDialog", QT_TRANSLATE_NOOP("App::Property", "Top cover of rebar")).TopCover = t_cover
00231             rebar.setEditorMode("TopCover", 2)
00232             rebar.addProperty("App::PropertyDistance", "TrueSpacing", "RebarDialog", QT_TRANSLATE_NOOP("App::Property", "Spacing between of rebars")).TrueSpacing = amount_spacing_value
00233             rebar.setEditorMode("TrueSpacing", 2)
00234             rebar.addProperty("App::PropertyString", "Orientation", "RebarDialog", QT_TRANSLATE_NOOP("App::Property", "Shape of rebar")).Orientation = orientation
00235             rebar.setEditorMode("Orientation", 2)
00236             if amount_spacing_check:
00237                 rebar.AmountCheck = True
00238             else:
```

```

00233     rebar.AmountCheck = False
00234     rebar.TrueSpacing = amount_spacing_value
00235     rebar.Label = "UShapeRebar"
00236     FreeCAD.ActiveDocument.recompute()
00237     return rebar
00238

```

Here is the call graph for this function:



Here is the caller graph for this function:



6.10.2 Variable Documentation

6.10.2.1 string `UShapeRebar.__author__` = "Amritpal Singh" [private]

Definition at line 25 of file [UShapeRebar.py](#).

6.10.2.2 string `UShapeRebar.__title__` = "UShapeRebar" [private]

Definition at line 24 of file [UShapeRebar.py](#).

6.10.2.3 string `UShapeRebar.__url__` = "https://www.freecadweb.org" [private]

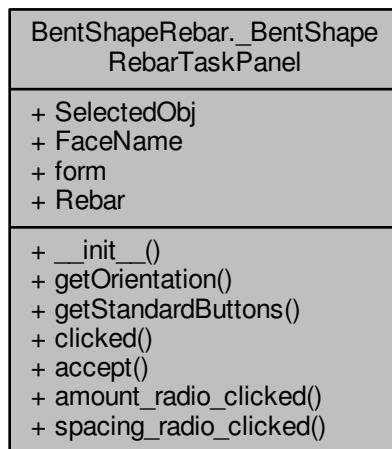
Definition at line 26 of file [UShapeRebar.py](#).

Chapter 7

Class Documentation

7.1 BentShapeRebar._BentShapeRebarTaskPanel Class Reference

Collaboration diagram for BentShapeRebar._BentShapeRebarTaskPanel:



Public Member Functions

- def `__init__` (self, `Rebar`=None)
- def `getOrientation` (self)
- def `getStandardButtons` (self)
- def `clicked` (self, button)
- def `accept` (self, signal=None)
- def `amount_radio_clicked` (self)
- def `spacing_radio_clicked` (self)

Public Attributes

- SelectedObj
- FaceName
- form
- Rebar

7.1.1 Detailed Description

Definition at line 105 of file [BentShapeRebar.py](#).

7.1.2 Constructor & Destructor Documentation

7.1.2.1 def BentShapeRebar._BentShapeRebarTaskPanel.__init__(self, Rebar = None)

Definition at line 106 of file [BentShapeRebar.py](#).

```
00106     def __init__(self, Rebar = None):
00107         if not Rebar:
00108             selected_obj = FreeCADGui.Selection.getSelectionEx()[0]
00109             self.SelectedObj = selected_obj.Object
00110             self.FaceName = selected_obj.SubElementNames[0]
00111         else:
00112             self.FaceName = Rebar.Base.Support[0][1][0]
00113             self.SelectedObj = Rebar.Base.Support[0][0]
00114         self.form = FreeCADGui.PySideUic.loadUi(os.path.splitext(__file__)[0] + ".ui")
00115         self.form.setWindowTitle(QtGui.QApplication.translate("RebarAddon", "Bent Shape Rebar", None))
00116         self.form.orientation.addItems(["Bottom", "Top", "Right", "Left"])
00117         self.form.amount_radio.clicked.connect(self.amount_radio_clicked)
00118         self.form.spacing_radio.clicked.connect(self.spacing_radio_clicked)
00119         self.form.customSpacing.clicked.connect(lambda: runRebarDistribution(self))
00120         self.form.removeCustomSpacing.clicked.connect(lambda:
00121             removeRebarDistribution(self))
00122         self.form.PickSelectedFace.clicked.connect(lambda: getSelectedFace(self))
00123         self.form.orientation.currentIndexChanged.connect(self.getOrientation)
00124         self.form.image.setPixmap(QtGui.QPixmap(os.path.split(os.path.abspath(__file__))[0] + "icons/BentShapeRebar.svg"))
00125         self.form.toolButton.setIcon(self.form.toolButton.style().standardIcon(
00126             QtGui.QStyle.SP_DialogHelpButton))
00127         self.form.toolButton.clicked.connect(lambda: showPopUpImageDialog(os.path.split(
00128             os.path.abspath(__file__))[0] + "/icons/BentShapeRebarDetailed.svg"))
00129         self.Rebar = Rebar
```

7.1.3 Member Function Documentation

7.1.3.1 def BentShapeRebar._BentShapeRebarTaskPanel.accept(self, signal = None)

Definition at line 146 of file [BentShapeRebar.py](#).

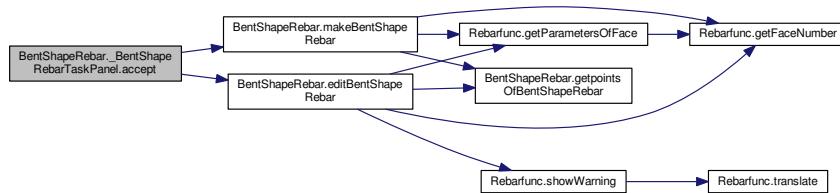
```
00146     def accept(self, signal = None):
00147         f_cover = self.form.frontCover.text()
00148         f_cover = FreeCAD.Units.Quantity(f_cover).Value
00149         b_cover = self.form.bottomCover.text()
00150         b_cover = FreeCAD.Units.Quantity(b_cover).Value
00151         l_cover = self.form.l_sideCover.text()
00152         l_cover = FreeCAD.Units.Quantity(l_cover).Value
00153         r_cover = self.form.r_sideCover.text()
00154         r_cover = FreeCAD.Units.Quantity(r_cover).Value
00155         t_cover = self.form.topCover.text()
00156         t_cover = FreeCAD.Units.Quantity(t_cover).Value
00157         bentLength = self.form.bentLength.text()
00158         bentLength = FreeCAD.Units.Quantity(bentLength).Value
```

```

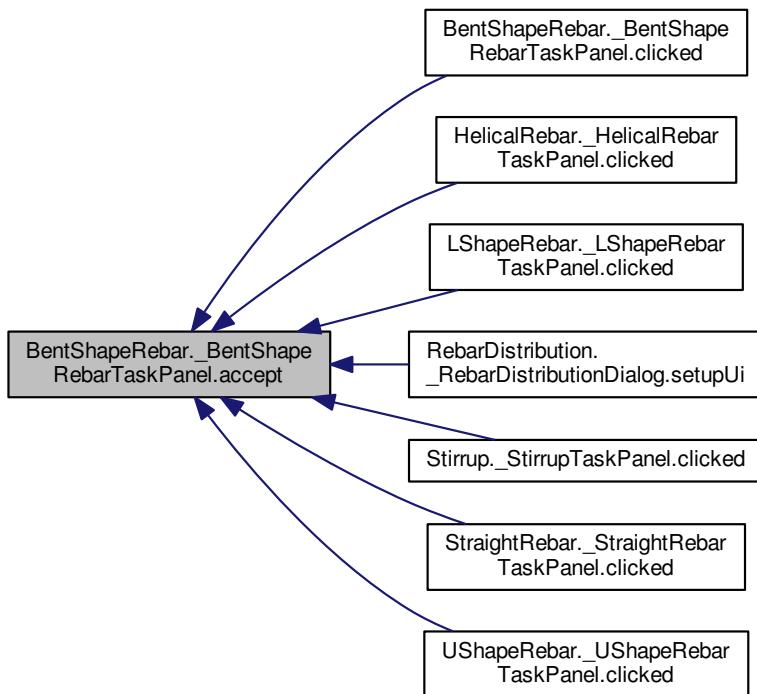
00159     bentAngle = self.form.bentAngle.value()
00160     diameter = self.form.diameter.text()
00161     diameter = FreeCAD.Units.Quantity(diameter).Value
00162     rounding = self.form.rounding.value()
00163     orientation = self.form.orientation.currentText()
00164     amount_check = self.form.amount_radio.isChecked()
00165     spacing_check = self.form.spacing_radio.isChecked()
00166     if not self.Rebar:
00167         if amount_check:
00168             amount = self.form.amount.value()
00169             rebar = makeBentShapeRebar(f_cover, b_cover, l_cover, r_cover, diameter,
00170             t_cover, bentLength, bentAngle, rounding, True, amount, orientation, self.
00171             SelectedObj, self.FaceName)
00172             elif spacing_check:
00173                 spacing = self.form.spacing.text()
00174                 spacing = FreeCAD.Units.Quantity(spacing).Value
00175                 rebar = makeBentShapeRebar(f_cover, b_cover, l_cover, r_cover, diameter,
00176                 t_cover, bentLength, bentAngle, rounding, False, spacing, orientation, self.
00177                 SelectedObj, self.FaceName)
00178             else:
00179                 if amount_check:
00180                     amount = self.form.amount.value()
00181                     rebar = editBentShapeRebar(self.Rebar, f_cover, b_cover, l_cover,
00182                     r_cover, diameter, t_cover, bentLength, bentAngle, rounding, True, amount, orientation, self.
00183                     SelectedObj, self.FaceName)
00184                 elif spacing_check:
00185                     spacing = self.form.spacing.text()
00186                     spacing = FreeCAD.Units.Quantity(spacing).Value
00187                     rebar = editBentShapeRebar(self.Rebar, f_cover, b_cover, l_cover,
00188                     r_cover, diameter, t_cover, bentLength, bentAngle, rounding, False, spacing, orientation, self.
00189                     SelectedObj, self.FaceName)
00190             if self.CustomSpacing:
00191                 rebar.CustomSpacing = self.CustomSpacing
00192                 FreeCAD.ActiveDocument.recompute()
00193             self.Rebar = rebar
00194             if signal == int(QtGui.QDialogButtonBox.Apply):
00195                 pass
00196             else:
00197                 FreeCADGui.Control.closeDialog(self)

```

Here is the call graph for this function:



Here is the caller graph for this function:



7.1.3.2 def BentShapeRebar._BentShapeRebarTaskPanel.amount_radio_clicked (self)

Definition at line 191 of file [BentShapeRebar.py](#).

```

00191     def amount_radio_clicked(self):
00192         self.form.spacing.setEnabled(False)
00193         self.form.amount.setEnabled(True)
00194
  
```

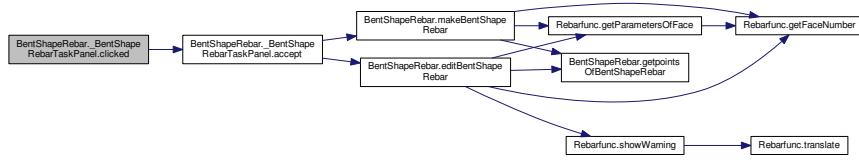
7.1.3.3 def BentShapeRebar._BentShapeRebarTaskPanel.clicked (self, button)

Definition at line 142 of file [BentShapeRebar.py](#).

```

00142     def clicked(self, button):
00143         if button == int(QtGui.QDialogButtonBox.Apply):
00144             self.accept(button)
00145
  
```

Here is the call graph for this function:



7.1.3.4 def BentShapeRebar._BentShapeRebarTaskPanel.getOrientation (self)

Definition at line 128 of file [BentShapeRebar.py](#).

```

00128     def getOrientation(self):
00129         orientation = self.form.orientation.currentText()
00130         #if orientation == "Bottom":
00131             #    self.form.image.setPixmap(QtGui.QPixmap(os.path.abspath(__file__))[0] +
00132             #        "/icons/LShapeRebarBR.svg"))
00133         #elif orientation == "Top":
00134             #    self.form.image.setPixmap(QtGui.QPixmap(os.path.abspath(__file__))[0] +
00135             #        "/icons/LShapeRebarBL.svg"))
00136         #elif orientation == "Right":
00137             #    self.form.image.setPixmap(QtGui.QPixmap(os.path.abspath(__file__))[0] +
00138             #        "/icons/LShapeRebarTR.svg"))
00139         #else:
00140             #    self.form.image.setPixmap(QtGui.QPixmap(os.path.abspath(__file__))[0] +
00141             #        "/icons/LShapeRebarTL.svg"))
00142
  
```

7.1.3.5 def BentShapeRebar._BentShapeRebarTaskPanel.getStandardButtons (self)

Definition at line 139 of file [BentShapeRebar.py](#).

```

00139     def getStandardButtons(self):
00140         return int(QtGui.QDialogButtonBox.Ok) | int(QtGui.QDialogButtonBox.Apply) | int(
00141             QtGui.QDialogButtonBox.Cancel)
  
```

7.1.3.6 def BentShapeRebar._BentShapeRebarTaskPanel.spacing_radio_clicked (self)

Definition at line 195 of file [BentShapeRebar.py](#).

```

00195     def spacing_radio_clicked(self):
00196         self.form.amount.setEnabled(False)
00197         self.form.spacing.setEnabled(True)
00198
00199
  
```

7.1.4 Member Data Documentation

7.1.4.1 BentShapeRebar._BentShapeRebarTaskPanel.FaceName

Definition at line 110 of file [BentShapeRebar.py](#).

7.1.4.2 BentShapeRebar._BentShapeRebarTaskPanel.form

Definition at line 114 of file [BentShapeRebar.py](#).

7.1.4.3 BentShapeRebar._BentShapeRebarTaskPanel.Rebar

Definition at line 126 of file [BentShapeRebar.py](#).

7.1.4.4 BentShapeRebar._BentShapeRebarTaskPanel.SelectedObj

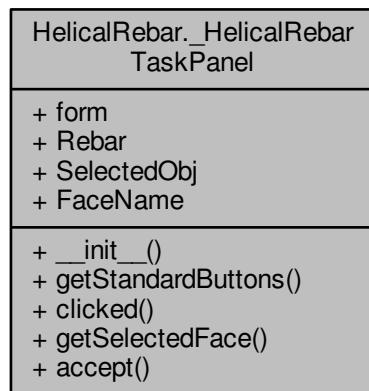
Definition at line 109 of file [BentShapeRebar.py](#).

The documentation for this class was generated from the following file:

- [BentShapeRebar.py](#)

7.2 HelicalRebar._HelicalRebarTaskPanel Class Reference

Collaboration diagram for HelicalRebar._HelicalRebarTaskPanel:



Public Member Functions

- def [__init__](#) (self, Rebar=None)
- def [getStandardButtons](#) (self)
- def [clicked](#) (self, button)
- def [getSelectedFace](#) (self)
- def [accept](#) (self, signal=None)

Public Attributes

- [form](#)
- [Rebar](#)
- [SelectedObj](#)
- [FaceName](#)

7.2.1 Detailed Description

Definition at line 107 of file [HelicalRebar.py](#).

7.2.2 Constructor & Destructor Documentation

7.2.2.1 def HelicalRebar_HelicalRebarTaskPanel.__init__(self, Rebar=None)

Definition at line 108 of file [HelicalRebar.py](#).

```
00108     def __init__(self, Rebar = None):
00109         self.form = FreeCADGui.PySideUic.loadUi(os.path.splitext(__file__)[0] + ".ui")
00110         self.form.setWindowTitle(QtGui.QApplication.translate("Arch", "Helical Rebar", None))
00111         if not Rebar:
00112             normal = facenormalDirection()
00113         else:
00114             normal = facenormalDirection(Rebar.Base.Support[0][0], Rebar.Base.Support[0]
00115                                         [1][0])
00116             if not round(normal.z) in {1, -1}:
00117                 self.form.topCoverLabel.setText(translate("RebarAddon", "Left Cover"))
00118                 self.form.bottomCoverLabel.setText(translate("RebarAddon", "Right Cover"))
00119             self.form.PickSelectedFace.clicked.connect(self.getSelectedFace)
00120             self.form.image.setPixmap(QtGui.QPixmap(os.path.split(os.path.abspath(__file__))[0] + "
00121 /icons/HelicalRebar.svg"))
00122             self.form.toolButton.clicked.connect(lambda: showPopUpImageDialog(os.path.split(
00123 os.path.abspath(__file__))[0] + "/icons/HelicalRebarDetailed.svg"))
00124             self.form.toolButton.setIcon(self.form.toolButton.style().standardIcon(
00125 QtGui.QStyle.SP_DialogHelpButton))
00126             self.Rebar = Rebar
00127             self.SelectedObj = None
00128             self.FaceName = None
00129
00130
00131
00132
00133
00134
00135
00136
00137
00138
00139
00140
00141
00142
00143
00144
00145
00146
00147
00148
00149
00150
00151
00152
00153
00154
00155
00156
00157
00158
00159
00160
00161
00162
00163
00164
```

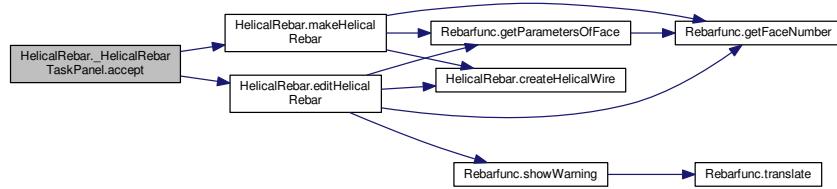
7.2.3 Member Function Documentation

7.2.3.1 def HelicalRebar_HelicalRebarTaskPanel.accept(self, signal=None)

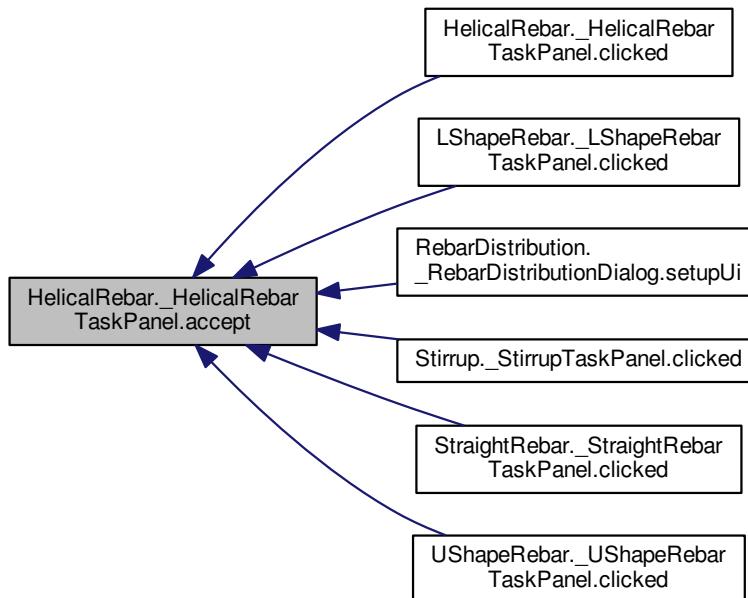
Definition at line 144 of file [HelicalRebar.py](#).

```
00144     def accept(self, signal = None):
00145         b_cover = self.form.bottomCover.text()
00146         b_cover = FreeCAD.Units.Quantity(b_cover).Value
00147         s_cover = self.form.sideCover.text()
00148         s_cover = FreeCAD.Units.Quantity(s_cover).Value
00149         t_cover = self.form.topCover.text()
00150         t_cover = FreeCAD.Units.Quantity(t_cover).Value
00151         pitch = self.form.pitch.text()
00152         pitch = FreeCAD.Units.Quantity(pitch).Value
00153         diameter = self.form.diameter.text()
00154         diameter = FreeCAD.Units.Quantity(diameter).Value
00155         if not self.Rebar:
00156             rebar = makeHelicalRebar(s_cover, b_cover, diameter, t_cover, pitch, self.
00157             SelectedObj, self.FaceName)
00158         else:
00159             rebar = editHelicalRebar(self.Rebar, s_cover, b_cover, diameter, t_cover,
00160             pitch, self.SelectedObj, self.FaceName)
00161             self.Rebar = rebar
00162             if signal == int(QtGui.QDialogButtonBox.Apply):
00163                 pass
00164             else:
00165                 FreeCADGui.Control.closeDialog(self)
```

Here is the call graph for this function:



Here is the caller graph for this function:



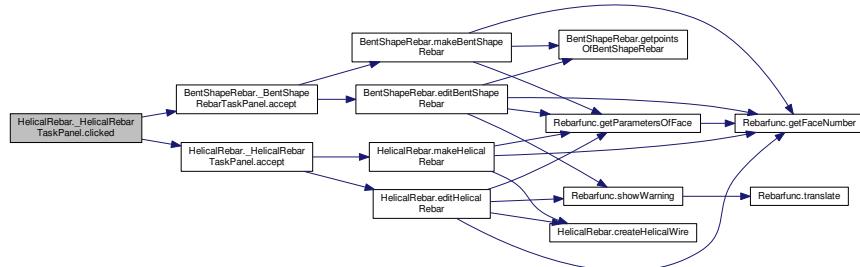
7.2.3.2 def HelicalRebar._HelicalRebarTaskPanel.clicked (self, button)

Definition at line 129 of file [HelicalRebar.py](#).

```

00129     def clicked(self, button):
00130         if button == int(QtGui.QDialogButtonBox.Apply):
00131             self.accept(button)
00132
  
```

Here is the call graph for this function:



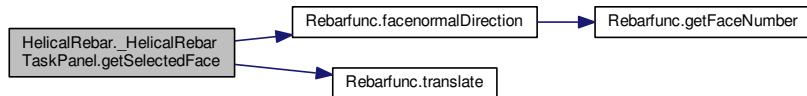
7.2.3.3 def HelicalRebar._HelicalRebarTaskPanel.getSelectedFace (self)

Definition at line 133 of file [HelicalRebar.py](#).

```

00133     def getSelectedFace(self):
00134         getSelectedFace(self)
00135         normal = facenormalDirection()
00136         if not round(normal.z) in {1, -1}:
00137             self.form.topCoverLabel.setText(translate("RebarAddon", "Left Cover"))
00138             self.form.bottomCoverLabel.setText(translate("RebarAddon", "Right Cover"))
00139         else:
00140             self.form.topCoverLabel.setText(translate("RebarAddon", "Top Cover"))
00141             self.form.bottomCoverLabel.setText(translate("RebarAddon", "Bottom Cover"))
00142
00143
  
```

Here is the call graph for this function:



7.2.3.4 def HelicalRebar._HelicalRebarTaskPanel.getStandardButtons (self)

Definition at line 126 of file [HelicalRebar.py](#).

```

00126     def getStandardButtons(self):
00127         return int(QtGui.QDialogButtonBox.Ok) | int(QtGui.QDialogButtonBox.Apply) | int(
00128             QtGui.QDialogButtonBox.Cancel)
  
```

7.2.4 Member Data Documentation

7.2.4.1 HelicalRebar._HelicalRebarTaskPanel.FaceName

Definition at line 124 of file [HelicalRebar.py](#).

7.2.4.2 HelicalRebar._HelicalRebarTaskPanel.form

Definition at line 109 of file [HelicalRebar.py](#).

7.2.4.3 HelicalRebar._HelicalRebarTaskPanel.Rebar

Definition at line 122 of file [HelicalRebar.py](#).

7.2.4.4 HelicalRebar._HelicalRebarTaskPanel.SelectedObj

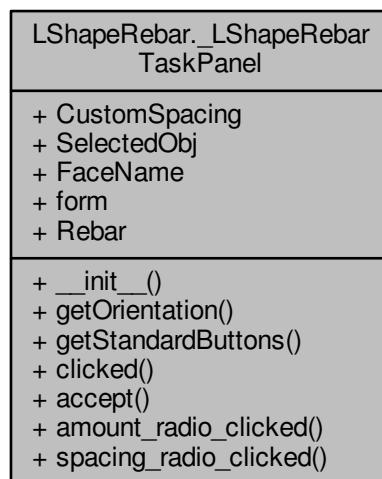
Definition at line 123 of file [HelicalRebar.py](#).

The documentation for this class was generated from the following file:

- [HelicalRebar.py](#)

7.3 LShapeRebar._LShapeRebarTaskPanel Class Reference

Collaboration diagram for LShapeRebar._LShapeRebarTaskPanel:



Public Member Functions

- def `__init__` (self, Rebar=None)
- def `getOrientation` (self)
- def `getStandardButtons` (self)
- def `clicked` (self, button)
- def `accept` (self, signal=None)
- def `amount_radio_clicked` (self)
- def `spacing_radio_clicked` (self)

Public Attributes

- [CustomSpacing](#)
- [SelectedObj](#)
- [FaceName](#)
- [form](#)
- [Rebar](#)

7.3.1 Detailed Description

Definition at line 76 of file [LShapeRebar.py](#).

7.3.2 Constructor & Destructor Documentation

7.3.2.1 def LShapeRebar._LShapeRebarTaskPanel.__init__(self, Rebar=None)

Definition at line 77 of file [LShapeRebar.py](#).

```
00077     def __init__(self, Rebar = None):
00078         self.CustomSpacing = None
00079         if not Rebar:
00080             selected_obj = FreeCADGui.Selection.getSelectionEx()[0]
00081             self.SelectedObj = selected_obj.Object
00082             self.FaceName = selected_obj.SubElementNames[0]
00083         else:
00084             self.FaceName = Rebar.Base.Support[0][1][0]
00085             self.SelectedObj = Rebar.Base.Support[0][0]
00086         self.form = FreeCADGui.PySideUic.loadUi(os.path.splitext(__file__)[0] + ".ui")
00087         self.form.setWindowTitle(QtGui.QApplication.translate("RebarAddon", "L-Shape Rebar", None))
00088         self.form.orientation.addItems(["Bottom Right", "Bottom Left", "Top Right", "Top Left"])
00089         self.form.amount_radio.clicked.connect(self.amount_radio_clicked)
00090         self.form.spacing_radio.clicked.connect(self.spacing_radio_clicked)
00091         self.form.customSpacing.clicked.connect(lambda: runRebarDistribution(self))
00092         self.form.removeCustomSpacing.clicked.connect(lambda:
00093             removeRebarDistribution(self))
00093         self.form.PickSelectedFace.clicked.connect(lambda: getSelectedFace(self))
00094         self.form.orientation.currentIndexChanged.connect(self.getOrientation)
00095         self.form.image.setPixmap(QtGui.QPixmap(os.path.split(os.path.abspath(__file__))[0] + "
00096             /icons/LShapeRebarBR.svg"))
00096         self.form.toolButton.setIcon(self.form.toolButton.style().standardIcon(
00097             QtGui.QStyle.SP_DialogHelpButton))
00097         self.form.toolButton.clicked.connect(lambda: showPopUpImageDialog(os.path.split(
00098             (os.path.abspath(__file__))[0] + "/icons/LShapeRebarDetailed.svg")))
00098         self.Rebar = Rebar
00099
```

7.3.3 Member Function Documentation

7.3.3.1 def LShapeRebar._LShapeRebarTaskPanel.accept(self, signal=None)

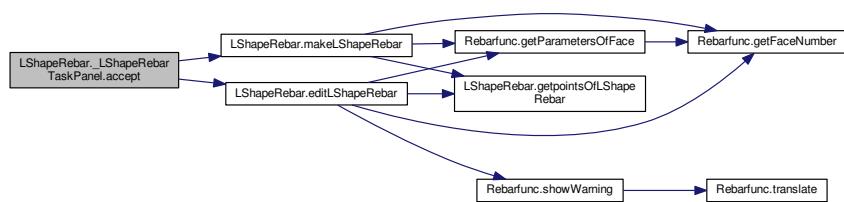
Definition at line 118 of file [LShapeRebar.py](#).

```

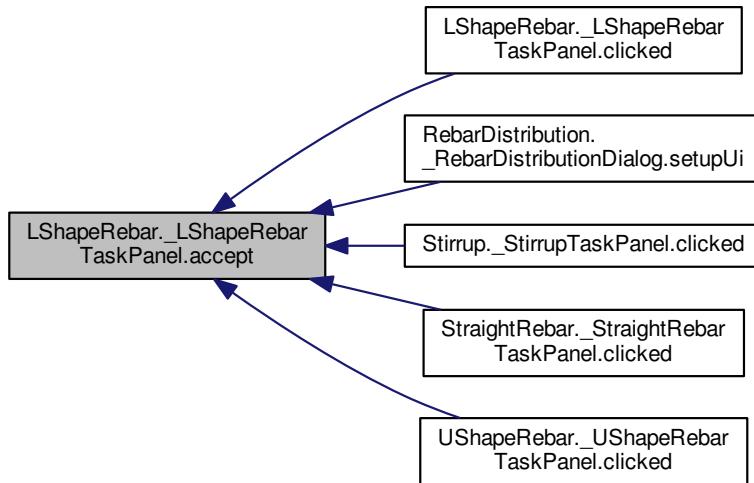
00118     def accept(self, signal = None):
00119         f_cover = self.form.frontCover.text()
00120         f_cover = FreeCAD.Units.Quantity(f_cover).Value
00121         b_cover = self.form.bottomCover.text()
00122         b_cover = FreeCAD.Units.Quantity(b_cover).Value
00123         l_cover = self.form.l_sideCover.text()
00124         l_cover = FreeCAD.Units.Quantity(l_cover).Value
00125         r_cover = self.form.r_sideCover.text()
00126         r_cover = FreeCAD.Units.Quantity(r_cover).Value
00127         t_cover = self.form.topCover.text()
00128         t_cover = FreeCAD.Units.Quantity(t_cover).Value
00129         diameter = self.form.diameter.text()
00130         diameter = FreeCAD.Units.Quantity(diameter).Value
00131         rounding = self.form.rounding.value()
00132         orientation = self.form.orientation.currentText()
00133         amount_check = self.form.amount_radio.isChecked()
00134         spacing_check = self.form.spacing_radio.isChecked()
00135         if not self.Rebar:
00136             if amount_check:
00137                 amount = self.form.amount.value()
00138                 rebar = makeLShapeRebar(f_cover, b_cover, l_cover, r_cover, diameter,
00139                                         t_cover, rounding, True, amount, orientation, self.SelectedObj, self.
00140                                         FaceName)
00141             elif spacing_check:
00142                 spacing = self.form.spacing.text()
00143                 spacing = FreeCAD.Units.Quantity(spacing).Value
00144                 rebar = makeLShapeRebar(f_cover, b_cover, l_cover, r_cover, diameter,
00145                                         t_cover, rounding, False, spacing, orientation, self.SelectedObj, self.
00146                                         FaceName)
00147             else:
00148                 if amount_check:
00149                     amount = self.form.amount.value()
00150                     rebar = editLShapeRebar(self.Rebar, f_cover, b_cover, l_cover, r_cover,
00151                                         diameter, t_cover, rounding, True, amount, orientation, self.SelectedObj, self.
00152                                         FaceName)
00153             elif spacing_check:
00154                 spacing = self.form.spacing.text()
00155                 spacing = FreeCAD.Units.Quantity(spacing).Value
00156                 rebar = editLShapeRebar(self.Rebar, f_cover, b_cover, l_cover, r_cover,
00157                                         diameter, t_cover, rounding, False, spacing, orientation, self.SelectedObj, self.
00158                                         FaceName)
00159         if self.CustomSpacing:
00160             rebar.CustomSpacing = self.CustomSpacing
00161             FreeCAD.ActiveDocument.recompute()
00162         self.Rebar = rebar
00163         if signal == int(QtGui.QDialogButtonBox.Apply):
00164             pass
00165         else:
00166             FreeCADGui.Control.closeDialog(self)
00167

```

Here is the call graph for this function:



Here is the caller graph for this function:



7.3.3.2 def LShapeRebar._LShapeRebarTaskPanel.amount_radio_clicked (self)

Definition at line 160 of file [LShapeRebar.py](#).

```

00160     def amount_radio_clicked(self):
00161         self.form.spacing.setEnabled(False)
00162         self.form.amount.setEnabled(True)
00163
  
```

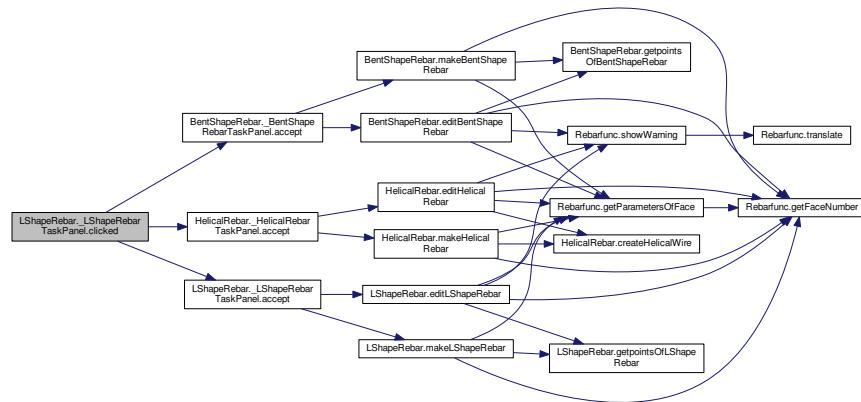
7.3.3.3 def LShapeRebar._LShapeRebarTaskPanel.clicked (self, button)

Definition at line 114 of file [LShapeRebar.py](#).

```

00114     def clicked(self, button):
00115         if button == int(QtGui.QDialogButtonBox.Apply):
00116             self.accept(button)
00117
  
```

Here is the call graph for this function:



7.3.3.4 def LShapeRebar._LShapeRebarTaskPanel.getOrientation(self)

Definition at line 100 of file [LShapeRebar.py](#).

```

00100     def getOrientation(self):
00101         orientation = self.form.orientation.currentText()
00102         if orientation == "Bottom Right":
00103             self.form.image.setPixmap(QtGui.QPixmap(os.path.split(os.path.abspath(__file__))[0] + "
00104 /icons/LShapeRebarBR.svg"))
00104         elif orientation == "Bottom Left":
00105             self.form.image.setPixmap(QtGui.QPixmap(os.path.split(os.path.abspath(__file__))[0] + "
00106 /icons/LShapeRebarBL.svg"))
00107         elif orientation == "Top Right":
00108             self.form.image.setPixmap(QtGui.QPixmap(os.path.split(os.path.abspath(__file__))[0] + "
00109 /icons/LShapeRebarTR.svg"))
00110         else:
00111             self.form.image.setPixmap(QtGui.QPixmap(os.path.split(os.path.abspath(__file__))[0] + "
00112 /icons/LShapeRebarTL.svg"))
00113

```

7.3.3.5 def LShapeRebar._LShapeRebarTaskPanel.getStandardButtons(self)

Definition at line 111 of file [LShapeRebar.py](#).

```

00111     def getStandardButtons(self):
00112         return int(QtGui.QDialogButtonBox.Ok) | int(QtGui.QDialogButtonBox.Apply) | int(
00113             QtGui.QDialogButtonBox.Cancel)
00114

```

7.3.3.6 def LShapeRebar._LShapeRebarTaskPanel.spacing_radio_clicked(self)

Definition at line 164 of file [LShapeRebar.py](#).

```

00164     def spacing_radio_clicked(self):
00165         self.form.amount.setEnabled(False)
00166         self.form.spacing.setEnabled(True)
00167
00168

```

7.3.4 Member Data Documentation

7.3.4.1 LShapeRebar._LShapeRebarTaskPanel.CustomSpacing

Definition at line 78 of file [LShapeRebar.py](#).

7.3.4.2 LShapeRebar._LShapeRebarTaskPanel.FaceName

Definition at line 82 of file [LShapeRebar.py](#).

7.3.4.3 LShapeRebar._LShapeRebarTaskPanel.form

Definition at line 86 of file [LShapeRebar.py](#).

7.3.4.4 LShapeRebar._LShapeRebarTaskPanel.Rebar

Definition at line 98 of file [LShapeRebar.py](#).

7.3.4.5 LShapeRebar._LShapeRebarTaskPanel.SelectedObj

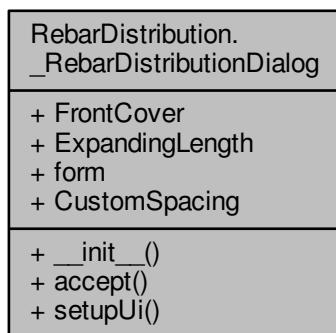
Definition at line 81 of file [LShapeRebar.py](#).

The documentation for this class was generated from the following file:

- [LShapeRebar.py](#)

7.4 RebarDistribution._RebarDistributionDialog Class Reference

Collaboration diagram for RebarDistribution._RebarDistributionDialog:



Public Member Functions

- def `__init__` (self, frontCover, size)
- def `accept` (self)
- def `setupUi` (self)

Public Attributes

- `FrontCover`
- `ExpandingLength`
- `form`
- `CustomSpacing`

7.4.1 Detailed Description

Definition at line 38 of file [RebarDistribution.py](#).

7.4.2 Constructor & Destructor Documentation

7.4.2.1 def RebarDistribution._RebarDistributionDialog.__init__ (self, frontCover, size)

Definition at line 39 of file [RebarDistribution.py](#).

```
00039     def __init__(self, frontCover, size):
00040         self.FrontCover = frontCover
00041         self.ExpandingLength = size
00042         self.form = FreeCADGui.PySideUic.loadUi(os.path.splitext(__file__)[0] + ".ui")
00043         self.form.setWindowTitle(QtGui.QApplication.translate("Arch", "Rebar Distribution", None))
00044         self.form.image.setPixmap(QtGui.QPixmap(os.path.split(os.path.abspath(__file__))[0] + "
00045             /icons/RebarDistribution.svg"))
```

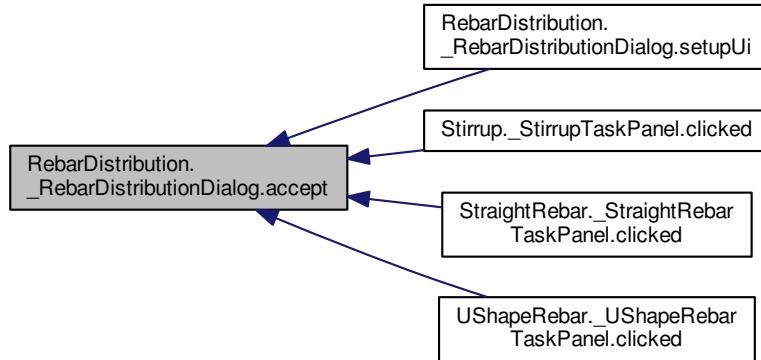
7.4.3 Member Function Documentation

7.4.3.1 def RebarDistribution._RebarDistributionDialog.accept (self)

Definition at line 46 of file [RebarDistribution.py](#).

```
00046     def accept(self):
00047         amount1 = self.form.amount1.value()
00048         spacing1 = self.form.spacing1.text()
00049         spacing1 = FreeCAD.Units.Quantity(spacing1).Value
00050         amount2 = self.form.amount2.value()
00051         spacing2 = self.form.spacing2.text()
00052         spacing2 = FreeCAD.Units.Quantity(spacing2).Value
00053         amount3 = self.form.amount3.value()
00054         spacing3 = self.form.spacing3.text()
00055         spacing3 = FreeCAD.Units.Quantity(spacing3).Value
00056         self.CustomSpacing = getCustomSpacingString(amount1, spacing1,
00057             amount2, spacing2, amount3, spacing3, self.FrontCover, self.
00058             ExpandingLength)
00059
```

Here is the caller graph for this function:



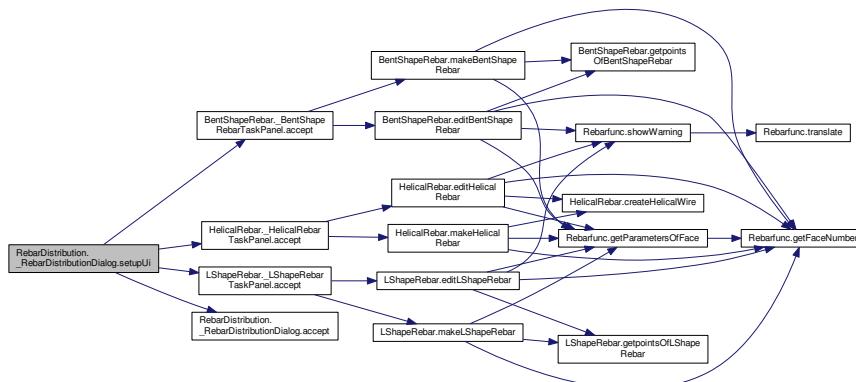
7.4.3.2 def RebarDistribution._RebarDistributionDialog.setupUi (self)

Definition at line 58 of file [RebarDistribution.py](#).

```

00058     def setupUi(self):
00059         # Connect Signals and Slots
00060         self.form.buttonBox.accepted.connect(self.accept)
00061         pass
00062
  
```

Here is the call graph for this function:



7.4.4 Member Data Documentation

7.4.4.1 RebarDistribution._RebarDistributionDialog.CustomSpacing

Definition at line 56 of file [RebarDistribution.py](#).

7.4.4.2 RebarDistribution._RebarDistributionDialog.ExpandingLength

Definition at line 41 of file [RebarDistribution.py](#).

7.4.4.3 RebarDistribution._RebarDistributionDialog.form

Definition at line 42 of file [RebarDistribution.py](#).

7.4.4.4 RebarDistribution._RebarDistributionDialog.FrontCover

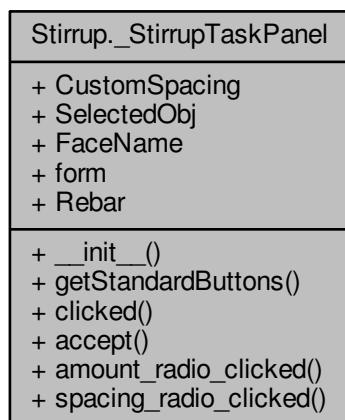
Definition at line 40 of file [RebarDistribution.py](#).

The documentation for this class was generated from the following file:

- [RebarDistribution.py](#)

7.5 Stirrup._StirrupTaskPanel Class Reference

Collaboration diagram for Stirrup._StirrupTaskPanel:



Public Member Functions

- def [__init__](#) (self, Rebar=None)
- def [getStandardButtons](#) (self)
- def [clicked](#) (self, button)
- def [accept](#) (self, signal=None)
- def [amount_radio_clicked](#) (self)
- def [spacing_radio_clicked](#) (self)

Public Attributes

- [CustomSpacing](#)
- [SelectedObj](#)
- [FaceName](#)
- [form](#)
- [Rebar](#)

7.5.1 Detailed Description

Definition at line 123 of file [Stirrup.py](#).

7.5.2 Constructor & Destructor Documentation

7.5.2.1 def Stirrup._StirrupTaskPanel.__init__(self, Rebar=None)

Definition at line 124 of file [Stirrup.py](#).

```
00124     def __init__(self, Rebar = None):
00125         self.CustomSpacing = None
00126         if not Rebar:
00127             selected_obj = FreeCADGui.Selection.getSelectionEx()[0]
00128             self.SelectedObj = selected_obj.Object
00129             self.FaceName = selected_obj.SubElementNames[0]
00130         else:
00131             self.FaceName = Rebar.Base.Support[0][1][0]
00132             self.SelectedObj = Rebar.Base.Support[0][0]
00133             self.form = FreeCADGui.PySideUic.loadUi(os.path.splitext(__file__)[0] + ".ui")
00134             self.form.setWindowTitle(QtGui.QApplication.translate("RebarAddon", "Stirrup Rebar", None))
00135             self.form.bentAngle.addItem(["135", "90"])
00136             self.form.amount_radio.clicked.connect(self.amount_radio_clicked)
00137             self.form.spacing_radio.clicked.connect(self.spacing_radio_clicked)
00138             self.form.image.setPixmap(QtGui.QPixmap(os.path.split(os.path.abspath(__file__))[0]+"
00139 /icons/Stirrup.svg"))
00140             self.form.customSpacing.clicked.connect(lambda: runRebarDistribution(self))
00141             self.form.removeCustomSpacing.clicked.connect(lambda:
00142                 removeRebarDistribution(self))
00143             self.form.PickSelectedFace.clicked.connect(lambda: getSelectedFace(self))
00144             self.form.toolButton.setIcon(self.form.toolButton.style().standardIcon(
00145                 QtGui.QStyle.SP_DialogHelpButton))
00146             self.form.toolButton.clicked.connect(lambda: showPopUpImageDialog(os.path.split
00147 (os.path.abspath(__file__))[0] + "/icons/StirrupDetailed.svg"))
00148             self.Rebar = Rebar
00149
00150
00151
00152
00153
00154
00155
00156
00157
00158
00159
00160
00161
00162
00163
```

7.5.3 Member Function Documentation

7.5.3.1 def Stirrup._StirrupTaskPanel.accept(self, signal=None)

Definition at line 153 of file [Stirrup.py](#).

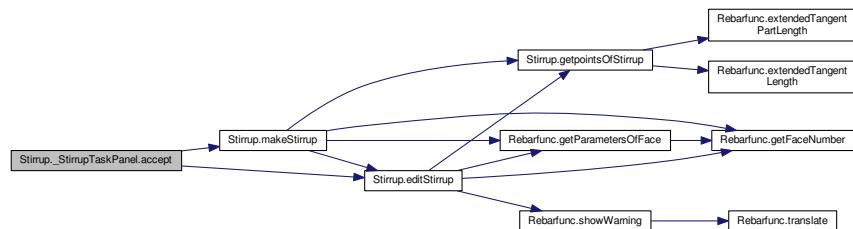
```
00153     def accept(self, signal = None):
00154         l_cover = self.form.l_sideCover.text()
00155         l_cover = FreeCAD.Units.Quantity(l_cover).Value
00156         r_cover = self.form.r_sideCover.text()
00157         r_cover = FreeCAD.Units.Quantity(r_cover).Value
00158         t_cover = self.form.t_sideCover.text()
00159         t_cover = FreeCAD.Units.Quantity(t_cover).Value
00160         b_cover = self.form.b_sideCover.text()
00161         b_cover = FreeCAD.Units.Quantity(b_cover).Value
00162         f_cover = self.form.frontCover.text()
00163         f_cover = FreeCAD.Units.Quantity(f_cover).Value
```

```

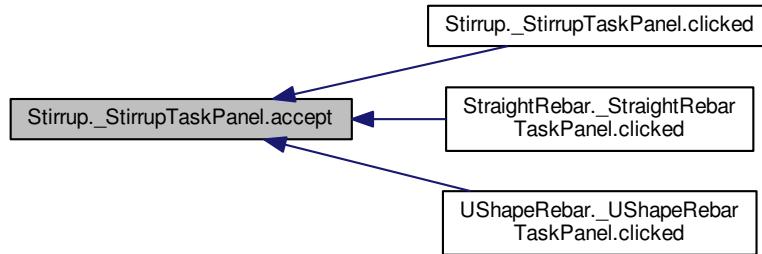
00164     diameter = self.form.diameter.text()
00165     diameter = FreeCAD.Units.Quantity(diameter).Value
00166     bentAngle = int(self.form.bentAngle.currentText())
00167     bentFactor = self.form.bentFactor.value()
00168     rounding = self.form.rounding.value()
00169     amount_check = self.form.amount_radio.isChecked()
00170     spacing_check = self.form.spacing_radio.isChecked()
00171     if not self.Rebar:
00172         if amount_check:
00173             amount = self.form.amount.value()
00174             rebar = makeStirrup(l_cover, r_cover, t_cover, b_cover, f_cover, bentAngle,
00175             bentFactor, diameter,\n                rounding, True, amount, self.SelectedObj, self.
00176             FaceName)
00177             elif spacing_check:
00178                 spacing = self.form.spacing.text()
00179                 spacing = FreeCAD.Units.Quantity(spacing).Value
00180                 rebar = makeStirrup(l_cover, r_cover, t_cover, b_cover, f_cover, bentAngle,
00181                 bentFactor, diameter,\n                rounding, False, spacing, self.SelectedObj, self.
00182                 FaceName)
00183             else:
00184                 if amount_check:
00185                     amount = self.form.amount.value()
00186                     rebar = editStirrup(self.Rebar, l_cover, r_cover, t_cover, b_cover, f_cover
00187                     , bentAngle, bentFactor,\n                diameter, rounding, True, amount, self.SelectedObj, self.
00188                     FaceName)
00189                     elif spacing_check:
00190                         spacing = self.form.spacing.text()
00191                         spacing = FreeCAD.Units.Quantity(spacing).Value
00192                         rebar = editStirrup(self.Rebar, l_cover, r_cover, t_cover, b_cover, f_cover
00193                         , bentAngle, bentFactor,\n                diameter, rounding, False, spacing, self.SelectedObj, self.
00194                         FaceName)
00195             if self.CustomSpacing:
00196                 rebar.CustomSpacing = self.CustomSpacing
00197                 FreeCAD.ActiveDocument.recompute()
00198             self.Rebar = rebar
00199             if signal == int(QtGui.QDialogButtonBox.Apply):
00200                 pass
00201             else:
00202                 FreeCADGui.Control.closeDialog(self)

```

Here is the call graph for this function:



Here is the caller graph for this function:



7.5.3.2 def Stirrup._StirrupTaskPanel.amount_radio_clicked (self)

Definition at line 200 of file [Stirrup.py](#).

```

00200     def amount_radio_clicked(self):
00201         self.form.spacing.setEnabled(False)
00202         self.form.amount.setEnabled(True)
00203
  
```

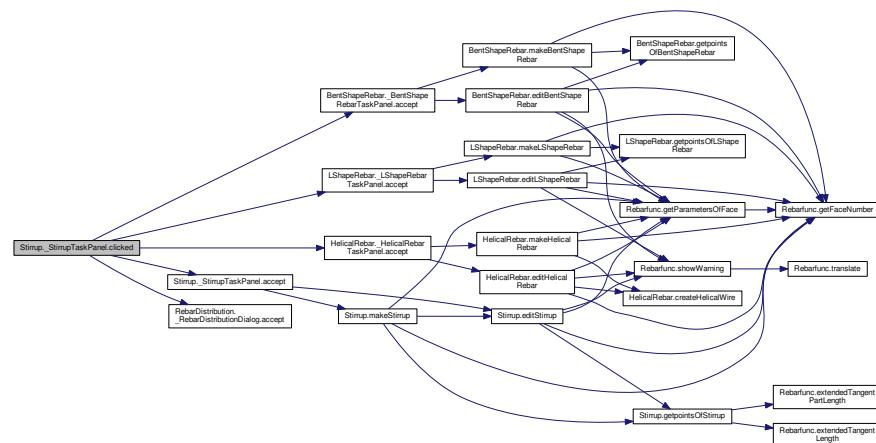
7.5.3.3 def Stirrup._StirrupTaskPanel.clicked (self, button)

Definition at line 149 of file [Stirrup.py](#).

```

00149     def clicked(self, button):
00150         if button == int(QtGui.QDialogButtonBox.Apply):
00151             self.accept(button)
00152
  
```

Here is the call graph for this function:



7.5.3.4 def Stirrup._StirrupTaskPanel.getStandardButtons (self)

Definition at line 146 of file [Stirrup.py](#).

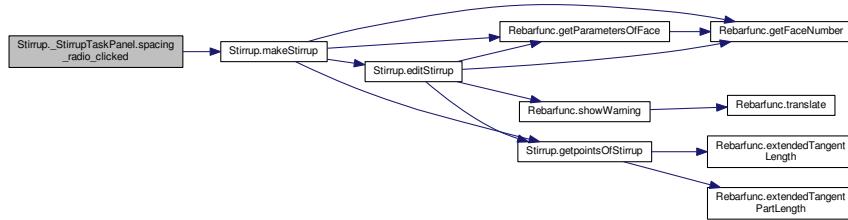
```
00146     def getStandardButtons(self):
00147         return int(QtGui.QDialogButtonBox.Ok) | int(QtGui.QDialogButtonBox.Apply) | int(
00148             QtGui.QDialogButtonBox.Cancel)
```

7.5.3.5 def Stirrup._StirrupTaskPanel.spacing_radio_clicked (self)

Definition at line 204 of file [Stirrup.py](#).

```
00204     def spacing_radio_clicked(self):
00205         self.form.amount.setEnabled(False)
00206         self.form.spacing.setEnabled(True)
00207
00208
```

Here is the call graph for this function:



7.5.4 Member Data Documentation

7.5.4.1 Stirrup._StirrupTaskPanel.CustomSpacing

Definition at line 125 of file [Stirrup.py](#).

7.5.4.2 Stirrup._StirrupTaskPanel.FaceName

Definition at line 129 of file [Stirrup.py](#).

7.5.4.3 Stirrup._StirrupTaskPanel.form

Definition at line 133 of file [Stirrup.py](#).

7.5.4.4 Stirrup._StirrupTaskPanel.Rebar

Definition at line 144 of file [Stirrup.py](#).

7.5.4.5 Stirrup._StirrupTaskPanel.SelectedObj

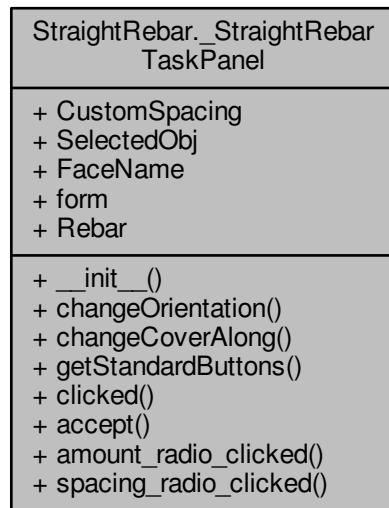
Definition at line 128 of file [Stirrup.py](#).

The documentation for this class was generated from the following file:

- [Stirrup.py](#)

7.6 StraightRebar._StraightRebarTaskPanel Class Reference

Collaboration diagram for StraightRebar._StraightRebarTaskPanel:



Public Member Functions

- def [__init__](#) (self, Rebar=None)
- def [changeOrientation](#) (self)
- def [changeCoverAlong](#) (self)
- def [getStandardButtons](#) (self)
- def [clicked](#) (self, button)
- def [accept](#) (self, signal=None)
- def [amount_radio_clicked](#) (self)
- def [spacing_radio_clicked](#) (self)

Public Attributes

- [CustomSpacing](#)
- [SelectedObj](#)
- [FaceName](#)
- [form](#)
- [Rebar](#)

7.6.1 Detailed Description

Definition at line 75 of file [StraightRebar.py](#).

7.6.2 Constructor & Destructor Documentation

7.6.2.1 def StraightRebar._StraightRebarTaskPanel.__init__(self, Rebar = None)

Definition at line 76 of file [StraightRebar.py](#).

```
00076     def __init__(self, Rebar = None):
00077         self.CustomSpacing = None
00078         if not Rebar:
00079             selected_obj = FreeCADGui.Selection.getSelectionEx()[0]
00080             self.SelectedObj = selected_obj.Object
00081             self.FaceName = selected_obj.SubElementNames[0]
00082         else:
00083             self.FaceName = Rebar.Base.Support[0][1][0]
00084             self.SelectedObj = Rebar.Base.Support[0][0]
00085         self.form = FreeCADGui.PySideUic.loadUi(os.path.splitext(__file__)[0] + ".ui")
00086         self.form.setWindowTitle(QtGui.QApplication.translate("RebarAddon", "Straight Rebar", None))
00087         self.form.orientation.addItem(["Horizontal", "Vertical"])
00088         self.form.coverAlong.addItem(["Bottom Side", "Top Side"])
00089         self.form.amount_radio.clicked.connect(self.amount_radio_clicked)
00090         self.form.spacing_radio.clicked.connect(self.spacing_radio_clicked)
00091         self.form.customSpacing.clicked.connect(lambda: runRebarDistribution(self))
00092         self.form.removeCustomSpacing.clicked.connect(lambda:
00093             removeRebarDistribution(self))
00093         self.form.PickSelectedFace.setCheckable(True)
00094         self.form.PickSelectedFace.toggle()
00095         self.form.PickSelectedFace.clicked.connect(lambda: getSelectedFace(self))
00096         self.form.image.setPixmap(QtGui.QPixmap(os.path.abspath(__file__))[0] + "icons/StraightRebarH.svg")
00097         self.form.orientation.currentIndexChanged.connect(self.changeOrientation)
00098         self.form.coverAlong.currentIndexChanged.connect(self.changeCoverAlong)
00099         self.form.toolButton.setIcon(self.form.toolButton.style().standardIcon(
00100             QtGui.QStyle.SP_DialogHelpButton))
00100         self.form.toolButton.clicked.connect(lambda: showPopUpImageDialog(os.path.split(
00101             os.path.abspath(__file__))[0] + "/icons/StraightRebarDetailed.svg"))
00101         self.Rebar = Rebar
00102
```

7.6.3 Member Function Documentation

7.6.3.1 def StraightRebar._StraightRebarTaskPanel.accept(self, signal = None)

Definition at line 136 of file [StraightRebar.py](#).

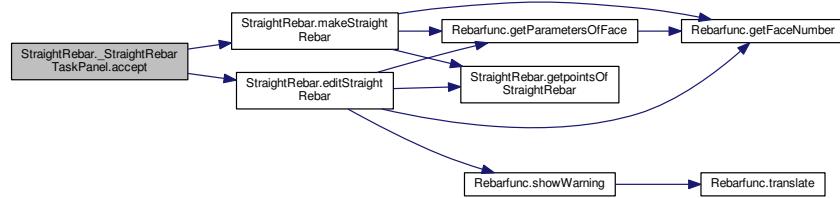
```
00136     def accept(self, signal = None):
00137         f_cover = self.form.frontCover.text()
00138         f_cover = FreeCAD.Units.Quantity(f_cover).Value
00139         cover = self.form.bottomCover.text()
00140         cover = FreeCAD.Units.Quantity(cover).Value
00141         lb_cover = self.form.l_sideCover.text()
00142         lb_cover = FreeCAD.Units.Quantity(lb_cover).Value
00143         rt_cover = self.form.r_sideCover.text()
00144         rt_cover = FreeCAD.Units.Quantity(rt_cover).Value
00145         orientation = self.form.orientation.currentText()
00146         coverAlong = self.form.coverAlong.currentText()
00147         diameter = self.form.diameter.text()
00148         diameter = FreeCAD.Units.Quantity(diameter).Value
00149         amount_check = self.form.amount_radio.isChecked()
00150         spacing_check = self.form.spacing_radio.isChecked()
00151         if not self.Rebar:
00152             if amount_check:
00153                 amount = self.form.amount.value()
00154                 rebar = makeStraightRebar(f_cover, (coverAlong, cover), rt_cover, lb_cover
00155                 , diameter, True, amount, orientation, self.SelectedObj, self.FaceName)
00156             elif spacing_check:
```

```

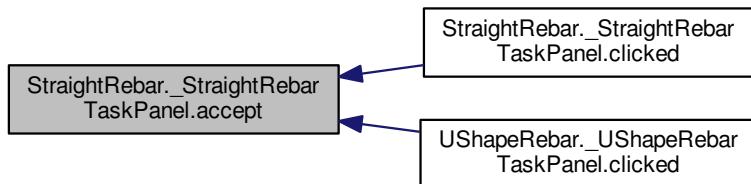
00156             spacing = self.form.spacing.text()
00157             spacing = FreeCAD.Units.Quantity(spacing).Value
00158             rebar = makeStraightRebar(f_cover, (coverAlong, cover), rt_cover, lb_cover
00159             , diameter, False, spacing, orientation, self.SelectedObj, self.
00160             FaceName)
00161             else:
00162                 if amount_check:
00163                     amount = self.form.amount.value()
00164                     rebar = editStraightRebar(self.Rebar, f_cover, (coverAlong, cover),
00165                     rt_cover, lb_cover, diameter, True, amount, orientation, self.SelectedObj, self.
00166                     FaceName)
00167                     elif spacing_check:
00168                         spacing = self.form.spacing.text()
00169                         spacing = FreeCAD.Units.Quantity(spacing).Value
00170                         rebar = editStraightRebar(self.Rebar, f_cover, (coverAlong, cover),
00171                         rt_cover, lb_cover, diameter, False, spacing, orientation, self.SelectedObj, self.
00172                         FaceName)
00173                     if self.CustomSpacing:
00174                         rebar.CustomSpacing = self.CustomSpacing
00175                         FreeCAD.ActiveDocument.recompute()
00176                     self.Rebar = rebar
00177                     if signal == int(QtGui.QDialogButtonBox.Apply):
00178                         pass
00179                     else:
00180                         FreeCADGui.Control.closeDialog(self)
00181

```

Here is the call graph for this function:



Here is the caller graph for this function:



7.6.3.2 def StraightRebar._StraightRebarTaskPanel.amount_radio_clicked (self)

Definition at line 176 of file [StraightRebar.py](#).

```

00176     def amount_radio_clicked(self):
00177         self.form.spacing.setEnabled(False)
00178         self.form.amount.setEnabled(True)
00179

```

7.6.3.3 def StraightRebar._StraightRebarTaskPanel.changeCoverAlong (self)

Definition at line 118 of file [StraightRebar.py](#).

```
00118     def changeCoverAlong(self):
00119         coverAlong = self.form.coverAlong.currentText()
00120         if coverAlong == "Bottom Side":
00121             self.form.bottomCoverLabel.setText("Bottom Cover")
00122         elif coverAlong == "Top Side":
00123             self.form.bottomCoverLabel.setText("Top Cover")
00124         elif coverAlong == "Right Side":
00125             self.form.bottomCoverLabel.setText("Right Cover")
00126         else:
00127             self.form.bottomCoverLabel.setText("Left Cover")
```

7.6.3.4 def StraightRebar._StraightRebarTaskPanel.changeOrientation (self)

Definition at line 103 of file [StraightRebar.py](#).

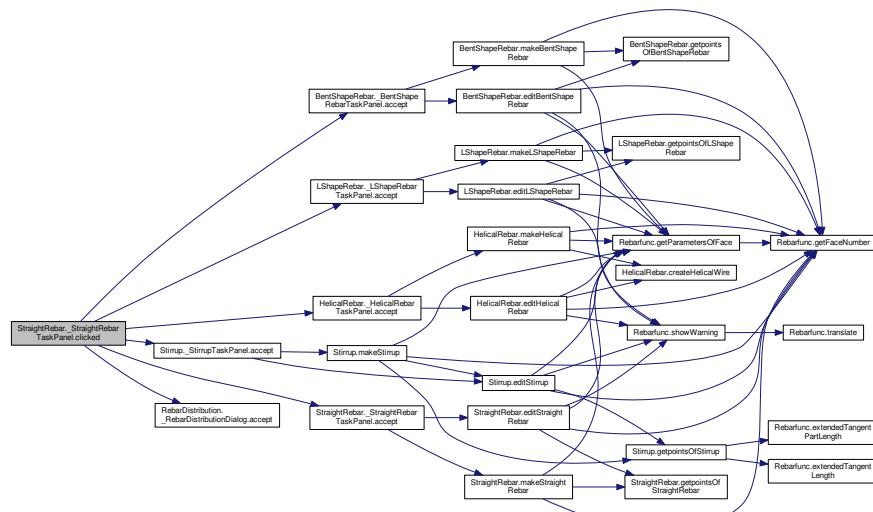
```
00103     def changeOrientation(self):
00104         orientation = self.form.orientation.currentText()
00105         if orientation == "Horizontal":
00106             self.form.image.setPixmap(QtGui.QPixmap(os.path.split(os.path.abspath(__file__))[0] + "icons/StraightRebarH.svg"))
00107             self.form.r_sideCoverLabel.setText("Right Side Cover")
00108             self.form.l_sideCoverLabel.setText("Left Side Cover")
00109             self.form.coverAlong.clear()
00110             self.form.coverAlong.addItem(["Bottom Side", "Top Side"])
00111         else:
00112             self.form.image.setPixmap(QtGui.QPixmap(os.path.split(os.path.abspath(__file__))[0] + "icons/StraightRebarV.svg"))
00113             self.form.r_sideCoverLabel.setText("Top Side Cover")
00114             self.form.l_sideCoverLabel.setText("Bottom Side Cover")
00115             self.form.coverAlong.clear()
00116             self.form.coverAlong.addItem(["Right Side", "Left Side"])
00117
```

7.6.3.5 def StraightRebar._StraightRebarTaskPanel.clicked (self, button)

Definition at line 132 of file [StraightRebar.py](#).

```
00132     def clicked(self, button):
00133         if button == int(QtGui.QDialogButtonBox.Apply):
00134             self.accept(button)
```

Here is the call graph for this function:



7.6.3.6 def StraightRebar._StraightRebarTaskPanel.getStandardButtons (self)

Definition at line 129 of file [StraightRebar.py](#).

```
00129     def getStandardButtons(self):
00130         return int(QtGui.QDialogButtonBox.Ok) | int(QtGui.QDialogButtonBox.Apply) | int(
00131             QtGui.QDialogButtonBox.Cancel)
```

7.6.3.7 def StraightRebar._StraightRebarTaskPanel.spacing_radio_clicked (self)

Definition at line 180 of file [StraightRebar.py](#).

```
00180     def spacing_radio_clicked(self):
00181         self.form.amount.setEnabled(False)
00182         self.form.spacing.setEnabled(True)
00183
00184
```

7.6.4 Member Data Documentation

7.6.4.1 StraightRebar._StraightRebarTaskPanel.CustomSpacing

Definition at line 77 of file [StraightRebar.py](#).

7.6.4.2 StraightRebar._StraightRebarTaskPanel.FaceName

Definition at line 81 of file [StraightRebar.py](#).

7.6.4.3 StraightRebar._StraightRebarTaskPanel.form

Definition at line 85 of file [StraightRebar.py](#).

7.6.4.4 StraightRebar._StraightRebarTaskPanel.Rebar

Definition at line 101 of file [StraightRebar.py](#).

7.6.4.5 StraightRebar._StraightRebarTaskPanel.SelectedObj

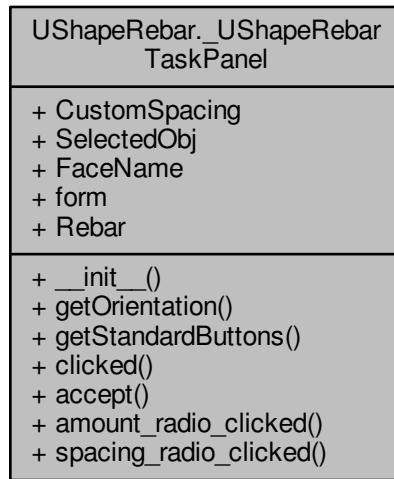
Definition at line 80 of file [StraightRebar.py](#).

The documentation for this class was generated from the following file:

- [StraightRebar.py](#)

7.7 UShapeRebar._UShapeRebarTaskPanel Class Reference

Collaboration diagram for UShapeRebar._UShapeRebarTaskPanel:



Public Member Functions

- def `__init__` (self, Rebar=None)
- def `getOrientation` (self)
- def `getStandardButtons` (self)
- def `clicked` (self, button)
- def `accept` (self, signal=None)
- def `amount_radio_clicked` (self)
- def `spacing_radio_clicked` (self)

Public Attributes

- `CustomSpacing`
- `SelectedObj`
- `FaceName`
- `form`
- `Rebar`

7.7.1 Detailed Description

Definition at line 84 of file [UShapeRebar.py](#).

7.7.2 Constructor & Destructor Documentation

7.7.2.1 def UShapeRebar_UShapeRebarTaskPanel.__init__(self, Rebar = None)

Definition at line 85 of file [UShapeRebar.py](#).

```

00085     def __init__(self, Rebar = None):
00086         self.CustomSpacing = None
00087         if not Rebar:
00088             selected_obj = FreeCADGui.Selection.getSelectionEx()[0]
00089             self.SelectedObj = selected_obj.Object
00090             self.FaceName = selected_obj.SubElementNames[0]
00091         else:
00092             self.FaceName = Rebar.Base.Support[0][1][0]
00093             self.SelectedObj = Rebar.Base.Support[0][0]
00094         self.form = FreeCADGui.PySideUic.loadUi(os.path.splitext(__file__)[0] + ".ui")
00095         self.form.setWindowTitle(QtGui.QApplication.translate("RebarAddon", "U-Shape Rebar", None))
00096         self.form.orientation.addItems(["Bottom", "Top", "Right", "Left"])
00097         self.form.amount_radio.clicked.connect(self.amount_radio_clicked)
00098         self.form.spacing_radio.clicked.connect(self.spacing_radio_clicked)
00099         self.form.customSpacing.clicked.connect(lambda: runRebarDistribution(self))
00100         self.form.removeCustomSpacing.clicked.connect(lambda:
00101             removeRebarDistribution(self))
00102         self.form.PickSelectedFace.clicked.connect(lambda: getSelectedFace(self))
00103         self.form.orientation.currentIndexChanged.connect(self.getOrientation)
00104         self.form.image.setPixmap(QPixmap(os.path.split(os.path.abspath(__file__))[0] + "
00104         /icons/UShapeRebarBottom.svg"))
00104         self.form.toolButton.setIcon(self.form.toolButton.style().standardIcon(
00104             QtGui.QStyle.SP_DialogHelpButton))
00105         self.form.toolButton.clicked.connect(lambda: showPopUpImageDialog(os.path.split(
00105             os.path.abspath(__file__))[0] + "/icons/UShapeRebarDetailed.svg"))
00106         self.Rebar = Rebar
00107

```

7.7.3 Member Function Documentation

7.7.3.1 def UShapeRebar_UShapeRebarTaskPanel.accept(self, signal = None)

Definition at line 126 of file [UShapeRebar.py](#).

```

00126     def accept(self, signal = None):
00127         f_cover = self.form.frontCover.text()
00128         f_cover = FreeCAD.Units.Quantity(f_cover).Value
00129         b_cover = self.form.bottomCover.text()
00130         b_cover = FreeCAD.Units.Quantity(b_cover).Value
00131         r_cover = self.form.r_sideCover.text()
00132         r_cover = FreeCAD.Units.Quantity(r_cover).Value
00133         l_cover = self.form.l_sideCover.text()
00134         l_cover = FreeCAD.Units.Quantity(l_cover).Value
00135         t_cover = self.form.topCover.text()
00136         t_cover = FreeCAD.Units.Quantity(t_cover).Value
00137         diameter = self.form.diameter.text()
00138         diameter = FreeCAD.Units.Quantity(diameter).Value
00139         rounding = self.form.rounding.value()
00140         orientation = self.form.orientation.currentText()
00141         amount_check = self.form.amount_radio.isChecked()
00142         spacing_check = self.form.spacing_radio.isChecked()
00143         if not self.Rebar:
00144             if amount_check:
00145                 amount = self.form.amount.value()
00146                 rebar = makeUShapeRebar(f_cover, b_cover, r_cover, l_cover, diameter,
00146                 t_cover, rounding, True, amount, orientation, self.SelectedObj, self.
00146                 FaceName)
00147             elif spacing_check:
00148                 spacing = self.form.spacing.text()
00149                 spacing = FreeCAD.Units.Quantity(spacing).Value
00150                 rebar = makeUShapeRebar(f_cover, b_cover, r_cover, l_cover, diameter,
00150                 t_cover, rounding, False, spacing, orientation, self.SelectedObj, self.
00150                 FaceName)
00151         else:
00152             if amount_check:
00153                 amount = self.form.amount.value()
00154                 rebar = editUShapeRebar(self.Rebar, f_cover, b_cover, r_cover, l_cover,
00154                 diameter, t_cover, rounding, True, amount, orientation, self.SelectedObj, self.

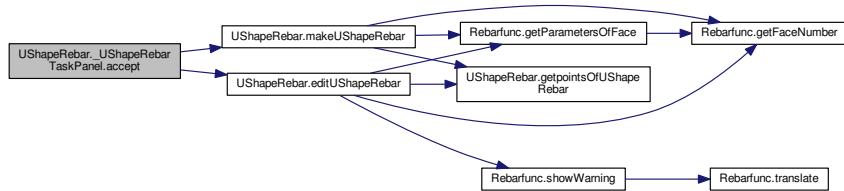
```

```

    FaceName)
00155         elif spacing_check:
00156             spacing = self.form.spacing.text()
00157             spacing = FreeCAD.Units.Quantity(spacing).Value
00158             rebar = editUShapeRebar(self.Rebar, f_cover, b_cover, r_cover, l_cover,
00159             diameter, t_cover, rounding, False, spacing, orientation, self.SelectedObj, self.
    FaceName)
00160             if self.CustomSpacing:
00161                 rebar.CustomSpacing = self.CustomSpacing
00162                 FreeCAD.ActiveDocument.recompute()
00163             self.Rebar = rebar
00164             if signal == int(QtGui.QDialogButtonBox.Apply):
00165                 pass
00166             else:
00167                 FreeCADGui.Control.closeDialog(self)

```

Here is the call graph for this function:



Here is the caller graph for this function:



7.7.3.2 def UShapeRebar._UShapeRebarTaskPanel.amount_radio_clicked (self)

Definition at line 168 of file [UShapeRebar.py](#).

```

00168     def amount_radio_clicked(self):
00169         self.form.spacing.setEnabled(False)
00170         self.form.amount.setEnabled(True)
00171

```

7.7.3.3 def UShapeRebar._UShapeRebarTaskPanel.clicked (self, button)

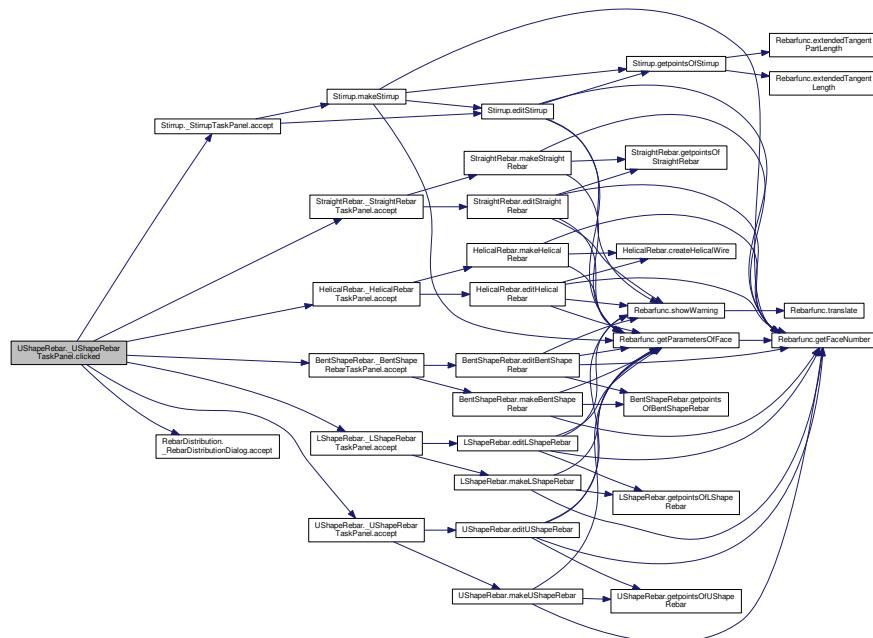
Definition at line 122 of file [UShapeRebar.py](#).

```

00122     def clicked(self, button):
00123         if button == int(QtGui.QDialogButtonBox.Apply):
00124             self.accept(button)
00125

```

Here is the call graph for this function:



7.7.3.4 def UShapeRebar_UShapeRebarTaskPanel.getOrientation (self)

Definition at line 108 of file [UShapeRebar.py](#).

```

00108     def getOrientation(self):
00109         orientation = self.form.orientation.currentText()
00110         if orientation == "Bottom":
00111             self.form.image.setPixmap(QtGui.QPixmap(os.path.split(os.path.abspath(__file__))[0] + "
00112 /icons/UShapeRebarBottom.svg"))
00113             elif orientation == "Top":
00114                 self.form.image.setPixmap(QtGui.QPixmap(os.path.split(os.path.abspath(__file__))[0] + "
00115 /icons/UShapeRebarTop.svg"))
00116             elif orientation == "Right":
00117                 self.form.image.setPixmap(QtGui.QPixmap(os.path.split(os.path.abspath(__file__))[0] + "
00118 /icons/UShapeRebarRight.svg"))
00119             else:
00120                 self.form.image.setPixmap(QtGui.QPixmap(os.path.split(os.path.abspath(__file__))[0] + "
00121 /icons/UShapeRebarLeft.svg"))

```

7.7.3.5 def UShapeRebar_UShapeRebarTaskPanel.getStandardButtons (self)

Definition at line 119 of file [UShapeRebar.py](#).

```

00119     def getStandardButtons(self):
00120         return int(QtGui.QDialogButtonBox.Ok) | int(QtGui.QDialogButtonBox.Apply) | int(
00121             QtGui.QDialogButtonBox.Cancel)

```

7.7.3.6 def UShapeRebar._UShapeRebarTaskPanel.spacing_radio_clicked (self)

Definition at line 172 of file [UShapeRebar.py](#).

```
00172     def spacing_radio_clicked(self):
00173         self.form.amount.setEnabled(False)
00174         self.form.spacing.setEnabled(True)
00175
00176
```

7.7.4 Member Data Documentation

7.7.4.1 UShapeRebar._UShapeRebarTaskPanel.CustomSpacing

Definition at line 86 of file [UShapeRebar.py](#).

7.7.4.2 UShapeRebar._UShapeRebarTaskPanel.FaceName

Definition at line 90 of file [UShapeRebar.py](#).

7.7.4.3 UShapeRebar._UShapeRebarTaskPanel.form

Definition at line 94 of file [UShapeRebar.py](#).

7.7.4.4 UShapeRebar._UShapeRebarTaskPanel.Rebar

Definition at line 106 of file [UShapeRebar.py](#).

7.7.4.5 UShapeRebar._UShapeRebarTaskPanel.SelectedObj

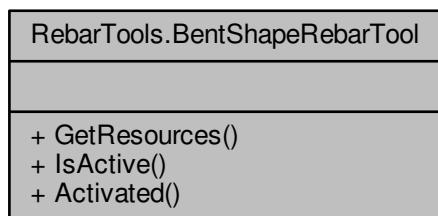
Definition at line 89 of file [UShapeRebar.py](#).

The documentation for this class was generated from the following file:

- [UShapeRebar.py](#)

7.8 RebarTools.BentShapeRebarTool Class Reference

Collaboration diagram for RebarTools.BentShapeRebarTool:



Public Member Functions

- def [GetResources](#) (self)
- def [IsActive](#) (self)
- def [Activated](#) (self)

7.8.1 Detailed Description

Definition at line 104 of file [RebarTools.py](#).

7.8.2 Member Function Documentation

7.8.2.1 def RebarTools.BentShapeRebarTool.Activated (self)

Definition at line 117 of file [RebarTools.py](#).

```
00117     def Activated(self):
00118         import BentShapeRebar
00119         # Call to CommandBentShaepRebar() function
00120         BentShapeRebar.CommandBentShapeRebar()
00121
```

Here is the call graph for this function:



7.8.2.2 def RebarTools.BentShapeRebarTool.GetResources (self)

Definition at line 106 of file [RebarTools.py](#).

```
00106     def GetResources(self):
00107         return {'Pixmap' : os.path.split(os.path.abspath(__file__))[0]+'/icons/dropdown_list/BentShapeRebar.svg',
00108                 'MenuText': QT_TRANSLATE_NOOP("RebarAddon", "Bent-Shape Rebar"),
00109                 'ToolTip' : QT_TRANSLATE_NOOP("RebarAddon", "Creates a BentShape bar reinforcement from the
00110 selected face of the Structural element.")}
```

7.8.2.3 def RebarTools.BentShapeRebarTool.IsActive (self)

Definition at line 111 of file [RebarTools.py](#).

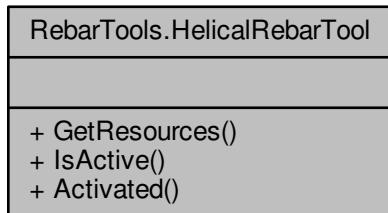
```
00111     def IsActive(self):
00112         if FreeCADGui.ActiveDocument:
00113             return True
00114         else:
00115             return False
00116
```

The documentation for this class was generated from the following file:

- [RebarTools.py](#)

7.9 RebarTools.HelicalRebarTool Class Reference

Collaboration diagram for RebarTools.HelicalRebarTool:



Public Member Functions

- def `GetResources (self)`
- def `IsActive (self)`
- def `Activated (self)`

7.9.1 Detailed Description

Definition at line 122 of file [RebarTools.py](#).

7.9.2 Member Function Documentation

7.9.2.1 def RebarTools.HelicalRebarTool.Activated (self)

Definition at line 135 of file [RebarTools.py](#).

```

00135     def Activated(self):
00136         import HelicalRebar
00137         # Call to CommandHelicalRebar() function
00138         HelicalRebar.CommandHelicalRebar()
00139
00140     FreeCADGui.addCommand('Arch_Rebar_Straight', StraightRebarTool())
00141     FreeCADGui.addCommand('Arch_Rebar_UShape', UShapeRebarTool())
00142     FreeCADGui.addCommand('Arch_Rebar_LShape', LShapeRebarTool())
00143     FreeCADGui.addCommand('Arch_Rebar_Stirrup', StirrupTool())
00144     FreeCADGui.addCommand('Arch_Rebar_BentShape', BentShapeRebarTool())
00145     FreeCADGui.addCommand('Arch_Rebar_Helical', HelicalRebarTool())
00146
00147 # List of all rebar commands
  
```

Here is the call graph for this function:



7.9.2.2 def RebarTools.HelicalRebarTool.GetResources (self)

Definition at line 124 of file [RebarTools.py](#).

```
00124     def GetResources(self):
00125         return {'Pixmap' : os.path.split(os.path.abspath(__file__))[0]+'/'
00126                 'icons/dropdown_list/HelixShapeRebar.svg',
00127                 'MenuText': QT_TRANSLATE_NOOP("RebarAddon", "Helical Rebar"),
00128                 'ToolTip' : QT_TRANSLATE_NOOP("RebarAddon", "Creates a Helical bar reinforcement from the
00129 selected face of the Structural element.")}
00130
00131
00132
00133
00134
```

7.9.2.3 def RebarTools.HelicalRebarTool.IsActive (self)

Definition at line 129 of file [RebarTools.py](#).

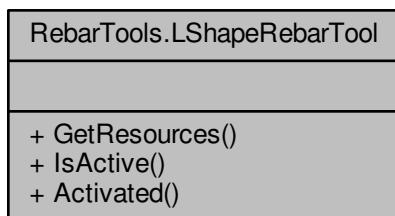
```
00129     def IsActive(self):
00130         if FreeCADGui.ActiveDocument:
00131             return True
00132         else:
00133             return False
00134
```

The documentation for this class was generated from the following file:

- [RebarTools.py](#)

7.10 RebarTools.LShapeRebarTool Class Reference

Collaboration diagram for RebarTools.LShapeRebarTool:



Public Member Functions

- def [GetResources \(self\)](#)
- def [IsActive \(self\)](#)
- def [Activated \(self\)](#)

7.10.1 Detailed Description

Definition at line 68 of file [RebarTools.py](#).

7.10.2 Member Function Documentation

7.10.2.1 def RebarTools.LShapeRebarTool.Activated (self)

Definition at line 81 of file [RebarTools.py](#).

```
00081     def Activated(self):
00082         import LShapeRebar
00083         # Call to CommandUShaeprRebar() function
00084         LShapeRebar.CommandLShapeRebar()
00085
```

Here is the call graph for this function:



7.10.2.2 def RebarTools.LShapeRebarTool.GetResources (self)

Definition at line 70 of file [RebarTools.py](#).

```
00070     def GetResources(self):
00071         return {'Pixmap' : os.path.abspath(__file__))[0]+'/'
00072             /icons/dropdown_list/LShapeRebar.svg',
00073             'MenuText': QT_TRANSLATE_NOOP("RebarAddon", "L-Shape Rebar"),
00074             'ToolTip' : QT_TRANSLATE_NOOP("RebarAddon", "Creates a L-Shape bar reinforcement from the
00075 selected face of the Structural element.")}
00074
```

7.10.2.3 def RebarTools.LShapeRebarTool.IsActive (self)

Definition at line 75 of file [RebarTools.py](#).

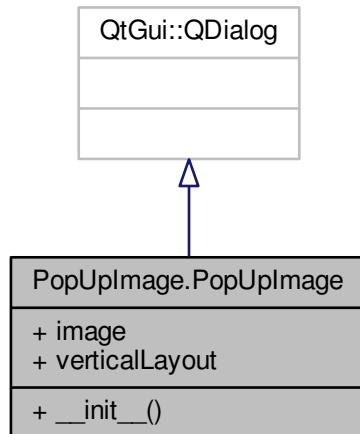
```
00075     def IsActive(self):
00076         if FreeCADGui.ActiveDocument:
00077             return True
00078         else:
00079             return False
00080
```

The documentation for this class was generated from the following file:

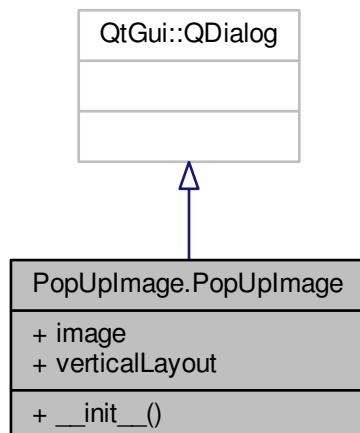
- [RebarTools.py](#)

7.11 PopUpImage.PopUpImage Class Reference

Inheritance diagram for PopUpImage.PopUpImage:



Collaboration diagram for PopUpImage.PopUpImage:



Public Member Functions

- def `__init__` (self, img)

Public Attributes

- [image](#)
- [verticalLayout](#)

7.11.1 Detailed Description

Definition at line [35](#) of file [PopUpImage.py](#).

7.11.2 Constructor & Destructor Documentation

7.11.2.1 def PopUpImage.PopUpImage.__init__(self, img)

Definition at line [36](#) of file [PopUpImage.py](#).

```
00036     def __init__(self, img):
00037         QtGui.QDialog.__init__(self)
00038         self.image = QtSvg.QSvgWidget(img)
00039         self.setWindowTitle(QtGui.QApplication.translate("RebarTool", "Detailed description", None))
00040         self.verticalLayout = QtGui.QVBoxLayout(self)
00041         self.verticalLayout.addWidget(self.image)
00042
```

7.11.3 Member Data Documentation

7.11.3.1 PopUpImage.PopUpImage.image

Definition at line [38](#) of file [PopUpImage.py](#).

7.11.3.2 PopUpImage.PopUpImage.verticalLayout

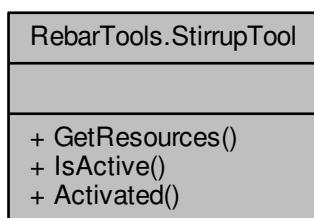
Definition at line [40](#) of file [PopUpImage.py](#).

The documentation for this class was generated from the following file:

- [PopUpImage.py](#)

7.12 RebarTools.StirrupTool Class Reference

Collaboration diagram for RebarTools.StirrupTool:



Public Member Functions

- def [GetResources](#) (*self*)
- def [IsActive](#) (*self*)
- def [Activated](#) (*self*)

7.12.1 Detailed Description

Definition at line [86](#) of file [RebarTools.py](#).

7.12.2 Member Function Documentation

7.12.2.1 def RebarTools.StirrupTool.Activated (*self*)

Definition at line [99](#) of file [RebarTools.py](#).

```
00099     def Activated(self):
00100         import Stirrup
00101         # Call to CommandStirrup() function
00102         Stirrup.CommandStirrup()
00103
```

Here is the call graph for this function:



7.12.2.2 def RebarTools.StirrupTool.GetResources (*self*)

Definition at line [88](#) of file [RebarTools.py](#).

```
00088     def GetResources(self):
00089         return {'Pixmap' : os.path.split(os.path.abspath(__file__))[0]+'/'
00090                 'icons/dropdown_list/StirrupRebar.svg',
00091                 'MenuText': QT_TRANSLATE_NOOP("RebarAddon", "Stirrup"),
00092                 'ToolTip' : QT_TRANSLATE_NOOP("RebarAddon", "Creates a Stirrup bar reinforcement from the
00093 selected face of the Structural element.")}
```

7.12.2.3 def RebarTools.StirrupTool.IsActive (*self*)

Definition at line [93](#) of file [RebarTools.py](#).

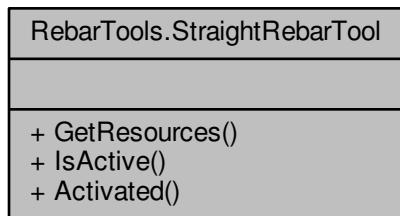
```
00093     def IsActive(self):
00094         if FreeCADGui.ActiveDocument:
00095             return True
00096         else:
00097             return False
00098
```

The documentation for this class was generated from the following file:

- [RebarTools.py](#)

7.13 RebarTools.StraightRebarTool Class Reference

Collaboration diagram for RebarTools.StraightRebarTool:



Public Member Functions

- def [GetResources](#) (self)
- def [IsActive](#) (self)
- def [Activated](#) (self)

7.13.1 Detailed Description

Definition at line [32](#) of file [RebarTools.py](#).

7.13.2 Member Function Documentation

7.13.2.1 def RebarTools.StraightRebarTool.Activated (self)

Definition at line [45](#) of file [RebarTools.py](#).

```

00045     def Activated(self):
00046         import StraightRebar
00047         # Call to CommandStraightRebar() function
00048         StraightRebar.CommandStraightRebar()
00049
  
```

Here is the call graph for this function:



7.13.2.2 def RebarTools.StraightRebarTool.GetResources (self)

Definition at line 34 of file [RebarTools.py](#).

```
00034     def GetResources(self):
00035         return {'Pixmap' : os.path.split(os.path.abspath(__file__))[0]+'/'
00036                 'icons/dropdown_list/StraightRebar.svg',
00037                 'MenuText': QT_TRANSLATE_NOOP("RebarAddon", "Straight Rebar"),
00038                 'ToolTip' : QT_TRANSLATE_NOOP("RebarAddon", "Creates a Striaght bar reinforcement from the
00039 selected face of the Structural element.")}
00040
00041
00042
00043
00044
```

7.13.2.3 def RebarTools.StraightRebarTool.IsActive (self)

Definition at line 39 of file [RebarTools.py](#).

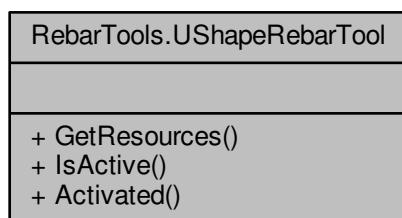
```
00039     def IsActive(self):
00040         if FreeCADGui.ActiveDocument:
00041             return True
00042         else:
00043             return False
00044
```

The documentation for this class was generated from the following file:

- [RebarTools.py](#)

7.14 RebarTools.UShapeRebarTool Class Reference

Collaboration diagram for RebarTools.UShapeRebarTool:



Public Member Functions

- def [GetResources \(self\)](#)
- def [IsActive \(self\)](#)
- def [Activated \(self\)](#)

7.14.1 Detailed Description

Definition at line 50 of file [RebarTools.py](#).

7.14.2 Member Function Documentation

7.14.2.1 def RebarTools.UShapeRebarTool.Activated (self)

Definition at line 63 of file [RebarTools.py](#).

```
00063     def Activated(self):
00064         import UShapeRebar
00065         # Call to CommandUShapeRebar() function
00066         UShapeRebar.CommandUShapeRebar()
00067
```

Here is the call graph for this function:



7.14.2.2 def RebarTools.UShapeRebarTool.GetResources (self)

Definition at line 52 of file [RebarTools.py](#).

```
00052     def GetResources(self):
00053         return {'Pixmap' : os.path.split(os.path.abspath(__file__))[0] +
00054                 '/icons/dropdown_list/UShapeRebar.svg',
00055                 'MenuText': QT_TRANSLATE_NOOP("RebarAddon", "U-Shape Rebar"),
00056                 'ToolTip' : QT_TRANSLATE_NOOP("RebarAddon", "Creates a U-Shape bar reinforcement from the
selected face of the Structural element.")}
00056
```

7.14.2.3 def RebarTools.UShapeRebarTool.IsActive (self)

Definition at line 57 of file [RebarTools.py](#).

```
00057     def IsActive(self):
00058         if FreeCADGui.ActiveDocument:
00059             return True
00060         else:
00061             return False
00062
```

The documentation for this class was generated from the following file:

- [RebarTools.py](#)

Chapter 8

File Documentation

8.1 BentShapeRebar.py File Reference

Classes

- class `BentShapeRebar.BentShapeRebarTaskPanel`

Namespaces

- `BentShapeRebar`

Functions

- def `BentShapeRebar.getpointsOfBentShapeRebar` (FacePRM, l_cover, r_cover, b_cover, t_cover, bentLength, bentAngle, orientation)
- def `BentShapeRebar.makeBentShapeRebar` (f_cover, b_cover, l_cover, r_cover, diameter, t_cover, bentLength, bentAngle, rounding, amount_spacing_check, amount_spacing_value, orientation="Bottom Left", structure=None, facename=None)
- def `BentShapeRebar.editBentShapeRebar` (Rebar, f_cover, b_cover, l_cover, r_cover, diameter, t_cover, bentLength, bentAngle, rounding, amount_spacing_check, amount_spacing_value, orientation, structure=None, facename=None)
- def `BentShapeRebar.editDialog` (vobj)
- def `BentShapeRebar.CommandBentShapeRebar` ()

Variables

- string `BentShapeRebar.__title__` = "BentShapeRebar"
- string `BentShapeRebar.__author__` = "Amritpal Singh"
- string `BentShapeRebar.__url__` = "https://www.freecadweb.org"

8.2 BentShapeRebar.py

```

00001 # --- coding: utf-8 ---
00002 # ****
00003 # *
00004 # * Copyright (c) 2017 - Amritpal Singh <amrit3701@gmail.com>
00005 # *
00006 # * This program is free software; you can redistribute it and/or modify
00007 # * it under the terms of the GNU Lesser General Public License (LGPL)
00008 # * as published by the Free Software Foundation; either version 2 of
00009 # * the License, or (at your option) any later version.
00010 # * for detail see the LICENCE text file.
00011 # *
00012 # * This program is distributed in the hope that it will be useful,
00013 # * but WITHOUT ANY WARRANTY; without even the implied warranty of
00014 # * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
00015 # * GNU Library General Public License for more details.
00016 # *
00017 # * You should have received a copy of the GNU Library General Public
00018 # * License along with this program; if not, write to the Free Software
00019 # * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307
00020 # * USA
00021 # *
00022 # ****
00023
00024 __title__ = "BentShapeRebar"
00025 __author__ = "Amritpal Singh"
00026 __url__ = "https://www.freecadweb.org"
00027
00028 from PySide import QtCore, QtGui
00029 from Rebarfunc import *
00030 from PySide.QtCore import QT_TRANSLATE_NOOP
00031 from RebarDistribution import runRebarDistribution, removeRebarDistribution
00032 from PopUpImage import showPopUpImageDialog
00033 import FreeCAD
00034 import FreeCADGui
00035 import ArchCommands
00036 import os
00037 import sys
00038 import math
00039
00040 def getpointsOfBentShapeRebar(FacePRM, l_cover, r_cover, b_cover, t_cover,
      bentLength, bentAngle, orientation):
00041     """ getpointsOfBentShapeRebar(FacePRM, LeftCover, RightCover, BottomCover, TopCover, BentLength,
      BentAngle, Orientation):
00042     Return points of the LShape rebar in the form of array for sketch.
00043     It takes four different orientations input i.e. 'Bottom', 'Top', 'Left', 'Right'.
00044     """
00045     if orientation == "Bottom":
00046         x1 = FacePRM[1][0] - FacePRM[0][0] / 2 + l_cover
00047         y1 = FacePRM[1][1] + FacePRM[0][1] / 2 - t_cover
00048         x2 = x1 + bentLength
00049         y2 = y1
00050         dis = (FacePRM[0][1] - t_cover - b_cover) * math.tan(math.radians(bentAngle - 90))
00051         x3 = x2 + dis
00052         y3 = FacePRM[1][1] - FacePRM[0][1] / 2 + b_cover
00053         x4 = FacePRM[1][0] + FacePRM[0][0] / 2 - r_cover - bentLength - dis
00054         y4 = y3
00055         x5 = x4 + dis
00056         y5 = y2
00057         x6 = x5 + bentLength
00058         y6 = y5
00059     elif orientation == "Top":
00060         x1 = FacePRM[1][0] - FacePRM[0][0] / 2 + l_cover
00061         y1 = FacePRM[1][1] - FacePRM[0][1] / 2 + b_cover
00062         x2 = x1 + bentLength
00063         y2 = y1
00064         dis = (FacePRM[0][1] - t_cover - b_cover) * math.tan(math.radians(bentAngle - 90))
00065         x3 = x2 + dis
00066         y3 = FacePRM[1][1] + FacePRM[0][1] / 2 - t_cover
00067         x4 = FacePRM[1][0] + FacePRM[0][0] / 2 - r_cover - bentLength - dis
00068         y4 = y3
00069         x5 = x4 + dis
00070         y5 = y2
00071         x6 = x5 + bentLength
00072         y6 = y5
00073     elif orientation == "Left":
00074         x1 = FacePRM[1][0] + FacePRM[0][0] / 2 - r_cover
00075         y1 = FacePRM[1][1] + FacePRM[0][1] / 2 - t_cover
00076         x2 = x1
00077         y2 = y1 - bentLength
00078         dis = (FacePRM[0][0] - r_cover - l_cover) * math.tan(math.radians(bentAngle - 90))
00079         x3 = FacePRM[1][0] - FacePRM[0][0] / 2 + l_cover
00080         y3 = y2 - dis
00081         x4 = x3
00082         y4 = FacePRM[1][1] - FacePRM[0][1] / 2 + b_cover + bentLength + dis

```

```

00083         x5 = x2
00084         y5 = y4 - dis
00085         x6 = x5
00086         y6 = y5 - bentLength
00087     elif orientation == "Right":
00088         x1 = FacePRM[1][0] - FacePRM[0][0] / 2 + l_cover
00089         y1 = FacePRM[1][1] + FacePRM[0][1] / 2 - t_cover
00090         x2 = x1
00091         y2 = y1 - bentLength
00092         dis = (FacePRM[0][0] - r_cover - l_cover) * math.tan(math.radians(bentAngle - 90))
00093         x3 = FacePRM[1][0] + FacePRM[0][0] / 2 - r_cover
00094         y3 = y2 - dis
00095         x4 = x3
00096         y4 = FacePRM[1][1] - FacePRM[0][1] / 2 + b_cover + bentLength + dis
00097         x5 = x2
00098         y5 = y4 - dis
00099         x6 = x5
00100         y6 = y5 - bentLength
00101     return [FreeCAD.Vector(x1, y1, 0), FreeCAD.Vector(x2, y2, 0), \
00102             FreeCAD.Vector(x3, y3, 0), FreeCAD.Vector(x4, y4, 0), \
00103             FreeCAD.Vector(x5, y5, 0), FreeCAD.Vector(x6, y6, 0)]
00104
00105 class _BentShapeRebarTaskPanel:
00106     def __init__(self, Rebar = None):
00107         if not Rebar:
00108             selected_obj = FreeCADGui.Selection.getSelectionEx()[0]
00109             self.SelectedObj = selected_obj.Object
00110             self.FaceName = selected_obj.SubElementNames[0]
00111         else:
00112             self.FaceName = Rebar.Base.Support[0][1][0]
00113             self.SelectedObj = Rebar.Base.Support[0][0]
00114         self.form = FreeCADGui.PySideUic.loadUi(os.path.splitext(__file__)[0] + ".ui")
00115         self.form.setWindowTitle(QtGui.QApplication.translate("RebarAddon", "Bent Shape Rebar", None))
00116         self.form.orientation.addItems(["Bottom", "Top", "Right", "Left"])
00117         self.form.amount_radio.clicked.connect(self.amount_radio_clicked)
00118         self.form.spacing_radio.clicked.connect(self.spacing_radio_clicked)
00119         self.form.customSpacing.clicked.connect(lambda: runRebarDistribution(self))
00120         self.form.removeCustomSpacing.clicked.connect(lambda:
00121             removeRebarDistribution(self))
00122         self.form.PickSelectedFace.clicked.connect(lambda: getSelectedFace(self))
00123         self.form.orientation.currentIndexChanged.connect(self.getOrientation)
00124         self.form.image.setPixmap(QtGui.QPixmap(os.path.split(os.path.abspath(__file__))[0] + \
00125             "/icons/BentShapeRebar.svg"))
00126         self.form.toolButton.setIcon(self.form.toolButton.style().standardIcon(
00127             QtGui.QStyle.SP_DialogHelpButton))
00128         self.form.toolButton.clicked.connect(lambda: showPopUpImageDialog(os.path.split(
00129             os.path.abspath(__file__))[0] + "/icons/BentShapeRebarDetailed.svg"))
00130         self.Rebar = Rebar
00131
00132     def getOrientation(self):
00133         orientation = self.form.orientation.currentText()
00134         #if orientation == "Bottom":
00135         #    self.form.image.setPixmap(QtGui.QPixmap(os.path.split(os.path.abspath(__file__))[0] + \
00136             "/icons/LShapeRebarBR.svg"))
00137         #elif orientation == "Top":
00138         #    self.form.image.setPixmap(QtGui.QPixmap(os.path.split(os.path.abspath(__file__))[0] + \
00139             "/icons/LShapeRebarBL.svg"))
00140         #elif orientation == "Right":
00141         #    self.form.image.setPixmap(QtGui.QPixmap(os.path.split(os.path.abspath(__file__))[0] + \
00142             "/icons/LShapeRebarTR.svg"))
00143         #else:
00144         #    self.form.image.setPixmap(QtGui.QPixmap(os.path.split(os.path.abspath(__file__))[0] + \
00145             "/icons/LShapeRebarTL.svg"))
00146     def getStandardButtons(self):
00147         return int(QtGui.QDialogButtonBox.Ok) | int(QtGui.QDialogButtonBox.Apply) | int(
00148             QtGui.QDialogButtonBox.Cancel)
00149
00150     def clicked(self, button):
00151         if button == int(QtGui.QDialogButtonBox.Apply):
00152             self.accept(button)
00153
00154     def accept(self, signal = None):
00155         f_cover = self.form.frontCover.text()
00156         f_cover = FreeCAD.Units.Quantity(f_cover).Value
00157         b_cover = self.form.bottomCover.text()
00158         b_cover = FreeCAD.Units.Quantity(b_cover).Value
00159         l_cover = self.form.l_sideCover.text()
00160         l_cover = FreeCAD.Units.Quantity(l_cover).Value
00161         r_cover = self.form.r_sideCover.text()
00162         r_cover = FreeCAD.Units.Quantity(r_cover).Value
00163         t_cover = self.form.topCover.text()
00164         t_cover = FreeCAD.Units.Quantity(t_cover).Value
00165         bentLength = self.form.bentLength.text()
00166         bentLength = FreeCAD.Units.Quantity(bentLength).Value
00167         bentAngle = self.form.bentAngle.value()
00168         diameter = self.form.diameter.text()

```

```

00161     diameter = FreeCAD.Units.Quantity(diameter).Value
00162     rounding = self.form.rounding.value()
00163     orientation = self.form.orientation.currentText()
00164     amount_check = self.form.amount_radio.isChecked()
00165     spacing_check = self.form.spacing_radio.isChecked()
00166     if not self.Rebar:
00167         if amount_check:
00168             amount = self.form.amount.value()
00169             rebar = makeBentShapeRebar(f_cover, b_cover, l_cover, r_cover, diameter,
00170             t_cover, bentLength, bentAngle, rounding, True, amount, orientation, self,
00171             SelectedObj, self.FaceName)
00172             elif spacing_check:
00173                 spacing = self.form.spacing.text()
00174                 spacing = FreeCAD.Units.Quantity(spacing).Value
00175                 rebar = makeBentShapeRebar(f_cover, b_cover, l_cover, r_cover, diameter,
00176                 t_cover, bentLength, bentAngle, rounding, False, spacing, orientation, self,
00177                 SelectedObj, self.FaceName)
00178             else:
00179                 if amount_check:
00180                     amount = self.form.amount.value()
00181                     rebar = editBentShapeRebar(self.Rebar, f_cover, b_cover, l_cover,
00182                     r_cover, diameter, t_cover, bentLength, bentAngle, rounding, True, amount, orientation, self,
00183                     SelectedObj, self.FaceName)
00184                     elif spacing_check:
00185                         spacing = self.form.spacing.text()
00186                         spacing = FreeCAD.Units.Quantity(spacing).Value
00187                         rebar = editBentShapeRebar(self.Rebar, f_cover, b_cover, l_cover,
00188                         r_cover, diameter, t_cover, bentLength, bentAngle, rounding, False, spacing, orientation, self,
00189                         SelectedObj, self.FaceName)
00190             if self.CustomSpacing:
00191                 rebar.CustomSpacing = self.CustomSpacing
00192                 FreeCAD.ActiveDocument.recompute()
00193             self.Rebar = rebar
00194             if signal == int(QtGui.QDialogButtonBox.Apply):
00195                 pass
00196             else:
00197                 FreeCADGui.Control.closeDialog(self)
00198
00199     def amount_radio_clicked(self):
00200         self.form.spacing.setEnabled(False)
00201         self.form.amount.setEnabled(True)
00202
00203     def spacing_radio_clicked(self):
00204         self.form.amount.setEnabled(False)
00205         self.form.spacing.setEnabled(True)
00206
00207     def makeBentShapeRebar(f_cover, b_cover, l_cover, r_cover, diameter, t_cover, bentLength,
00208     bentAngle, rounding, amount_spacing_check, amount_spacing_value, orientation = "Bottom Left", structure =
00209     None, facename = None):
00210         """ makeBentShapeRebar(FrontCover, BottomCover, LeftCover, RightCover, Diameter, TopCover, BentLength,
00211         BentAngle, Rounding,
00212         AmountSpacingCheck, AmountSpacingValue, Orientation, Structure, Facename): Adds the Bent-Shape
00213         reinforcement bar to the
00214         selected structural object.
00215         It takes four different orientations input i.e. 'Bottom', 'Top', 'Left', 'Right'.
00216         """
00217         if not structure and not facename:
00218             selected_obj = FreeCADGui.Selection.getSelectionEx()[0]
00219             structure = selected_obj.Object
00220             facename = selected_obj.SubElementNames[0]
00221             face = structure.Shape.Faces[getFaceNumber(facename) - 1]
00222             #StructurePRM = getTrueParametersOfStructure(structure)
00223             FacePRM = getParametersOfFace(structure, facename)
00224             if not FacePRM:
00225                 FreeCAD.Console.PrintError("Cannot identified shape or from which base object sturctural element is
00226 derived\n")
00227             return
00228         # Get points of L-Shape rebar
00229         points = getpointsOfBentShapeRebar(FacePRM, l_cover, r_cover, b_cover, t_cover
00230         , bentLength, bentAngle, orientation)
00231         import Part
00232         import Arch
00233         sketch = FreeCAD.activeDocument().addObject('Sketcher::SketchObject', 'Sketch')
00234         sketch.MapMode = "FlatFace"
00235         sketch.Support = [(structure, facename)]
00236         FreeCAD.ActiveDocument.recompute()
00237         sketch.addGeometry(Part.LineSegment(points[0], points[1]), False)
00238         sketch.addGeometry(Part.LineSegment(points[1], points[2]), False)
00239         sketch.addGeometry(Part.LineSegment(points[2], points[3]), False)
00240         sketch.addGeometry(Part.LineSegment(points[3], points[4]), False)
00241         sketch.addGeometry(Part.LineSegment(points[4], points[5]), False)
00242         import Sketcher
00243         if amount_spacing_check:
00244             rebar = Arch.makeRebar(structure, sketch, diameter, amount_spacing_value, f_cover)
00245             FreeCAD.ActiveDocument.recompute()
00246         else:
00247

```

```

00234     size = (ArchCommands.projectToVector(structure.Shape.copy(), face.normalAt(0, 0))).Length
00235     rebar = Arch.makeRebar(structure, sketch, diameter, int((size - diameter) / amount_spacing_value),
00236     f_cover)
00237     # Adds properties to the rebar object
00238     rebar.ViewObject.addProperty("App::PropertyString", "RebarShape", "RebarDialog", QT_TRANSLATE_NOOP("App::Property", "Shape of rebar")).RebarShape = "BentShapeRebar"
00239     rebar.ViewObject.setEditorMode("RebarShape", 2)
00240     rebar.addProperty("App::PropertyDistance", "FrontCover", "RebarDialog", QT_TRANSLATE_NOOP("App::Property", "Front cover of rebar")).FrontCover = f_cover
00241     rebar.setEditorMode("FrontCover", 2)
00242     rebar.addProperty("App::PropertyDistance", "LeftCover", "RebarDialog", QT_TRANSLATE_NOOP("App::Property", "Left Side cover of rebar")).LeftCover = l_cover
00243     rebar.setEditorMode("LeftCover", 2)
00244     rebar.addProperty("App::PropertyDistance", "RightCover", "RebarDialog", QT_TRANSLATE_NOOP("App::Property", "Right Side cover of rebar")).RightCover = r_cover
00245     rebar.setEditorMode("RightCover", 2)
00246     rebar.addProperty("App::PropertyDistance", "BottomCover", "RebarDialog", QT_TRANSLATE_NOOP("App::Property", "Bottom cover of rebar")).BottomCover = b_cover
00247     rebar.setEditorMode("BottomCover", 2)
00248     rebar.addProperty("App::PropertyBool", "AmountCheck", "RebarDialog", QT_TRANSLATE_NOOP("App::Property", "Amount radio button is checked")).AmountCheck
00249     rebar.setEditorMode("AmountCheck", 2)
00250     rebar.addProperty("App::PropertyDistance", "TopCover", "RebarDialog", QT_TRANSLATE_NOOP("App::Property", "Top cover of rebar")).TopCover = t_cover
00251     rebar.setEditorMode("TopCover", 2)
00252     rebar.addProperty("App::PropertyDistance", "TrueSpacing", "RebarDialog", QT_TRANSLATE_NOOP("App::Property", "Spacing between of rebars")).TrueSpacing = amount_spacing_value
00253     rebar.addProperty("App::PropertyString", "Orientation", "RebarDialog", QT_TRANSLATE_NOOP("App::Property", "Shape of rebar")).Orientation = orientation
00254     rebar.setEditorMode("Orientation", 2)
00255     rebar.setEditorMode("TrueSpacing", 2)
00256     rebar.addProperty("App::PropertyDistance", "BentLength", "RebarDialog", QT_TRANSLATE_NOOP("App::Property", "BentLength cover of rebar")).BentLength = bentLength
00257     rebar.setEditorMode("BentLength", 2)
00258     rebar.addProperty("App::PropertyDistance", "BentAngle", "RebarDialog", QT_TRANSLATE_NOOP("App::Property", "Bent Angle of rebar")).BentAngle = bentAngle
00259     rebar.setEditorMode("BentAngle", 2)
00260
00261     if amount_spacing_check:
00262         rebar.AmountCheck = True
00263     else:
00264         rebar.AmountCheck = False
00265         rebar.TrueSpacing = amount_spacing_value
00266     rebar.Label = "BentShapeRebar"
00267     FreeCAD.ActiveDocument.recompute()
00268     return rebar
00269
00270 def editBentShapeRebar(Rebar, f_cover, b_cover, l_cover, r_cover, diameter, t_cover,
00271     bentlength, bentAngle, rounding, amount_spacing_check, amount_spacing_value, orientation, structure = None,
00272     facename = None):
00273     sketch = Rebar.Base
00274     if structure and facename:
00275         sketch.Support = [(structure, facename)]
00276     # Check if sketch support is empty.
00277     if not sketch.Support:
00278         showWarning("You have checked remove external geometry of base sketchs when needed.\nTo uncheck Edit->Preferences->Arch.")
00279     # Assigned values
00280     facename = sketch.Support[0][1][0]
00281     structure = sketch.Support[0][0]
00282     face = structure.Shape.Faces[getFaceNumber(facename) - 1]
00283     #StructurePRM = getTrueParametersOfStructure(structure)
00284     # Get parameters of the face where sketch of rebar is drawn
00285     FacePRM = getParametersOfFace(structure, facename)
00286     # Get points of L-Shape rebar
00287     points = getpointsOfBentShapeRebar(FacePRM, l_cover, r_cover, b_cover, t_cover,
00288     bentLength, bentAngle, orientation)
00289     sketch.movePoint(0, 1, points[0], 0)
00290     FreeCAD.ActiveDocument.recompute()
00291     sketch.movePoint(0, 2, points[1], 0)
00292     FreeCAD.ActiveDocument.recompute()
00293     sketch.movePoint(1, 1, points[1], 0)
00294     FreeCAD.ActiveDocument.recompute()
00295     sketch.movePoint(1, 2, points[2], 0)
00296     FreeCAD.ActiveDocument.recompute()
00297     sketch.movePoint(2, 1, points[2], 0)
00298     FreeCAD.ActiveDocument.recompute()
00299     sketch.movePoint(2, 2, points[3], 0)
00300     FreeCAD.ActiveDocument.recompute()
00301     sketch.movePoint(3, 1, points[3], 0)
00302     FreeCAD.ActiveDocument.recompute()
00303     sketch.movePoint(3, 2, points[4], 0)
00304     FreeCAD.ActiveDocument.recompute()

```

```

00305     sketch.movePoint(4, 1, points[4], 0)
00306     FreeCAD.ActiveDocument.recompute()
00307     sketch.movePoint(4, 2, points[5], 0)
00308     FreeCAD.ActiveDocument.recompute()
00309
00310     Rebar.OffsetStart = f_cover
00311     Rebar.OffsetEnd = f_cover
00312     if amount_spacing_check:
00313         Rebar.Amount = amount_spacing_value
00314         FreeCAD.ActiveDocument.recompute()
00315         Rebar.AmountCheck = True
00316     else:
00317         size = (ArchCommands.projectToVector(structure.Shape.copy(), face.normalAt(0, 0))).Length
00318         Rebar.Amount = int((size - diameter) / amount_spacing_value)
00319         FreeCAD.ActiveDocument.recompute()
00320         Rebar.AmountCheck = False
00321     Rebar.Diameter = diameter
00322     Rebar.FrontCover = f_cover
00323     Rebar.LeftCover = l_cover
00324     Rebar.RightCover = r_cover
00325     Rebar.BottomCover = b_cover
00326     Rebar.TopCover = t_cover
00327     Rebar.BentLength = bentLength
00328     Rebar.BentAngle = bentAngle
00329     Rebar.Rounding = rounding
00330     Rebar.TrueSpacing = amount_spacing_value
00331     Rebar.Orientation = orientation
00332     FreeCAD.ActiveDocument.recompute()
00333     return Rebar
00334
00335 def editDialog(vobj):
00336     FreeCADGui.Control.closeDialog()
00337     obj = _BentShapeRebarTaskPanel(vobj.Object)
00338     obj.form.frontCover.setText(str(vobj.Object.FrontCover))
00339     obj.form.l_sideCover.setText(str(vobj.Object.LeftCover))
00340     obj.form.r_sideCover.setText(str(vobj.Object.RightCover))
00341     obj.form.bottomCover.setText(str(vobj.Object.BottomCover))
00342     obj.form.diameter.setText(str(vobj.Object.Diameter))
00343     obj.form.topCover.setText(str(vobj.Object.TopCover))
00344     obj.form.bentLength.setText(str(vobj.Object.BentLength))
00345     obj.form.bentAngle.setValue(vobj.Object.BentAngle)
00346     obj.form.rounding.setValue(vobj.Object.Rounding)
00347     obj.form.orientation.setCurrentIndex(obj.form.orientation.findText(str(vobj.Object.Orientation)))
00348     if vobj.Object.AmountCheck:
00349         obj.form.amount.setValue(vobj.Object.Amount)
00350     else:
00351         obj.form.amount_radio.setChecked(False)
00352         obj.form.spacing_radio.setChecked(True)
00353         obj.form.amount.setDisabled(True)
00354         obj.form.spacing.setEnabled(True)
00355         obj.form.spacing.setText(str(vobj.Object.TrueSpacing))
00356     #obj.form.PickSelectedFace.setVisible(False)
00357     FreeCADGui.Control.showDialog(obj)
00358
00359 def CommandBentShapeRebar():
00360     selected_obj = check_selected_face()
00361     if selected_obj:
00362         FreeCADGui.Control.showDialog(_BentShapeRebarTaskPanel())

```

8.3 HelicalRebar.py File Reference

Classes

- class [HelicalRebar._HelicalRebarTaskPanel](#)

Namespaces

- [HelicalRebar](#)

Functions

- def `HelicalRebar.getpointsOfHelicalRebar` (FacePRM, s_cover, b_cover, t_cover, pitch, edges, diameter, size, direction)
- def `HelicalRebar.createHelicalWire` (FacePRM, s_cover, b_cover, t_cover, pitch, size, direction, helix=None)
- def `HelicalRebar.makeHelicalRebar` (s_cover, b_cover, diameter, t_cover, pitch, structure=None, face-name=None)
- def `HelicalRebar.editHelicalRebar` (Rebar, s_cover, b_cover, diameter, t_cover, pitch, structure=None, face-name=None)
- def `HelicalRebar.editDialog` (vobj)
- def `HelicalRebar.CommandHelicalRebar` ()

Variables

- string `HelicalRebar.__title__` = "HelicalRebar"
- string `HelicalRebar.__author__` = "Amritpal Singh"
- string `HelicalRebar.__url__` = "https://www.freecadweb.org"

8.4 HelicalRebar.py

```

00001 # --- coding: utf-8 ---
00002 # ****
00003 # *
00004 # * Copyright (c) 2017 - Amritpal Singh <amrit3701@gmail.com>
00005 # *
00006 # * This program is free software; you can redistribute it and/or modify
00007 # * it under the terms of the GNU Lesser General Public License (LGPL)
00008 # * as published by the Free Software Foundation; either version 2 of
00009 # * the License, or (at your option) any later version.
00010 # * for detail see the LICENCE text file.
00011 # *
00012 # * This program is distributed in the hope that it will be useful,
00013 # * but WITHOUT ANY WARRANTY; without even the implied warranty of
00014 # * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
00015 # * GNU Library General Public License for more details.
00016 # *
00017 # * You should have received a copy of the GNU Library General Public
00018 # * License along with this program; if not, write to the Free Software
00019 # * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307
00020 # * USA
00021 # *
00022 # ****
00023
00024 __title__ = "HelicalRebar"
00025 __author__ = "Amritpal Singh"
00026 __url__ = "https://www.freecadweb.org"
00027
00028 from PySide import QtCore, QtGui
00029 from Rebarfunc import *
00030 from PySide.QtCore import QT_TRANSLATE_NOOP
00031 from PopUpImage import showPopUpImageDialog
00032 import FreeCAD
00033 import FreeCADGui
00034 import ArchCommands
00035 import os
00036 import sys
00037 import math
00038
00039 def getpointsOfHelicalRebar(FacePRM, s_cover, b_cover, t_cover, pitch, edges,
00040     diameter, size, direction):
00041     """ getpointsOfHelicalRebar(FacePRM, s_cover, b_cover, t_cover):
00042         Return points of the Lshape rebar in the form of array for sketch."""
00043     dx = s_cover + diameter / 2
00044     dz = float(pitch) / edges
00045     R = diameter / 2 - dx
00046     R = FacePRM[0][0] / 2 - s_cover
00047     points = []
00048     if direction[2] in {-1,1}:
00049         z = 0
00050         l = 0
00051         if direction[2] == 1:

```

```

00051         zz = FacePRM[1][2] - t_cover
00052     elif direction[2] == -1:
00053         zz = FacePRM[1][2] + b_cover
00054     count = 0
00055     flag = False
00056     while (round(z) < abs(size - b_cover - t_cover)):
00057         for i in range(0, int(edges) + 1):
00058             if not i and flag:
00059                 continue
00060             if not flag:
00061                 z -= dz
00062                 flag = True
00063             iAngle = i * 360 / edges
00064             x = FacePRM[1][0] + R * math.cos(math.radians(iAngle))
00065             y = FacePRM[1][1] + R * math.sin(math.radians(iAngle))
00066             points.append(FreeCAD.Vector(x, y, zz))
00067             count += 1
00068             if direction[2] == 1:
00069                 zz -= dz
00070             elif direction[2] == -1:
00071                 zz += dz
00072             z += dz
00073     return points
00074
00075 def createHelicalWire(FacePRM, s_cover, b_cover, t_cover, pitch, size, direction, helix =
None):
00076     """ createHelicalWire(FacePRM, SideCover, BottomCover, TopCover, Pitch, Size, Direction, Helix = None):
00077     It creates a helical wire."""
00078     import Part
00079     if not helix:
00080         helix = FreeCAD.ActiveDocument.addObject("Part::Helix","Helix")
00081     helix.Pitch = pitch
00082     helix.Radius = FacePRM[0][0] / 2 - s_cover
00083     helix.Angle = 0
00084     helix.LocalCoord = 0
00085     helix.Height = size - b_cover - t_cover
00086     if round(direction.x) == 1:
00087         helix.Placement.Base = FreeCAD.Vector(FacePRM[1][0] - b_cover, FacePRM[1][1], FacePRM[1][2])
00088         helix.Placement.Rotation = FreeCAD.Rotation(FreeCAD.Vector(0, -1, 0), 90)
00089     elif round(direction.x) == -1:
00090         helix.Placement.Base = FreeCAD.Vector(FacePRM[1][0] + t_cover, FacePRM[1][1], FacePRM[1][2])
00091         helix.Placement.Rotation = FreeCAD.Rotation(FreeCAD.Vector(0, -1, 0), -90)
00092     elif round(direction.y) == 1:
00093         helix.Placement.Base = FreeCAD.Vector(FacePRM[1][0], FacePRM[1][1] - b_cover, FacePRM[1][2])
00094         helix.Placement.Rotation = FreeCAD.Rotation(FreeCAD.Vector(1, 0, 0), 90)
00095     elif round(direction.y) == -1:
00096         helix.Placement.Base = FreeCAD.Vector(FacePRM[1][0], FacePRM[1][1] + t_cover, FacePRM[1][2])
00097         helix.Placement.Rotation = FreeCAD.Rotation(FreeCAD.Vector(-1, 0, 0), 90)
00098     elif round(direction.z) == 1:
00099         helix.Placement.Base = FreeCAD.Vector(FacePRM[1][0], FacePRM[1][1], FacePRM[1][2] - size + b_cover)
00100         helix.Placement.Rotation = FreeCAD.Rotation(FreeCAD.Vector(0, 0, 1), 0)
00101     elif round(direction.z) == -1:
00102         helix.Placement.Base = FreeCAD.Vector(FacePRM[1][0], FacePRM[1][1], FacePRM[1][2] + b_cover)
00103         helix.Placement.Rotation = FreeCAD.Rotation(FreeCAD.Vector(0, 0, -1), 0)
00104     FreeCAD.ActiveDocument.recompute()
00105     return helix
00106
00107 class _HelicalRebarTaskPanel:
00108     def __init__(self, Rebar = None):
00109         self.form = FreeCADGui.PySideUic.loadUi(os.path.splitext(__file__)[0] + ".ui")
00110         self.form.setWindowTitle(QtGui.QApplication.translate("Arch", "Helical Rebar", None))
00111         if not Rebar:
00112             normal = facenormalDirection()
00113         else:
00114             normal = facenormalDirection(Rebar.Base.Support[0][0], Rebar.Base.Support[0]
00115 [1][0])
00116             if not round(normal.z) in {1, -1}:
00117                 self.form.topCoverLabel.setText(translate("RebarAddon", "Left Cover"))
00118                 self.form.bottomCoverLabel.setText(translate("RebarAddon", "Right Cover"))
00119                 self.form.PickSelectedFace.clicked.connect(self.getSelectedFace)
00120                 self.form.image.setPixmap(QtGui.QPixmap(os.path.split(os.path.abspath(__file__))[0] +
"/icons/HelicalRebar.svg"))
00121                 self.form.toolButton.clicked.connect(lambda: showPopUpImageDialog(os.path.split
(os.path.abspath(__file__))[0] + "/icons/HelicalRebarDetailed.svg"))
00122                 self.form.toolButton.setIcon(self.form.toolButton.style().standardIcon(
QtGui.QStyle.SP_DialogHelpButton))
00123                 self.Rebar = Rebar
00124                 self.SelectedObj = None
00125                 self.FaceName = None
00126             def getStandardButtons(self):
00127                 return int(QtGui.QDialogButtonBox.Ok) | int(QtGui.QDialogButtonBox.Apply) | int(
QtGui.QDialogButtonBox.Cancel)
00128
00129             def clicked(self, button):
00130                 if button == int(QtGui.QDialogButtonBox.Apply):
00131                     self.accept(button)

```

```

00132
00133     def getSelectedFace(self):
00134         getSelectedFace(self)
00135         normal = facenormalDirection()
00136         if not round(normal.z) in {1, -1}:
00137             self.form.topCoverLabel.setText(translate("RebarAddon", "Left Cover"))
00138             self.form.bottomCoverLabel.setText(translate("RebarAddon", "Right Cover"))
00139         else:
00140             self.form.topCoverLabel.setText(translate("RebarAddon", "Top Cover"))
00141             self.form.bottomCoverLabel.setText(translate("RebarAddon", "Bottom Cover"))
00142
00143
00144     def accept(self, signal = None):
00145         b_cover = self.form.bottomCover.text()
00146         b_cover = FreeCAD.Units.Quantity(b_cover).Value
00147         s_cover = self.form.sideCover.text()
00148         s_cover = FreeCAD.Units.Quantity(s_cover).Value
00149         t_cover = self.form.topCover.text()
00150         t_cover = FreeCAD.Units.Quantity(t_cover).Value
00151         pitch = self.form.pitch.text()
00152         pitch = FreeCAD.Units.Quantity(pitch).Value
00153         diameter = self.form.diameter.text()
00154         diameter = FreeCAD.Units.Quantity(diameter).Value
00155         if not self.Rebar:
00156             rebar = makeHelicalRebar(s_cover, b_cover, diameter, t_cover, pitch, self.
00157             SelectedObj, self.FaceName)
00158         else:
00159             rebar = editHelicalRebar(self.Rebar, s_cover, b_cover, diameter, t_cover,
00160             pitch, self.SelectedObj, self.FaceName)
00161             self.Rebar = rebar
00162             if signal == int(QtGui.QDialogButtonBox.Apply):
00163                 pass
00164             else:
00165                 FreeCADGui.Control.closeDialog(self)
00166
00167     def makeHelicalRebar(s_cover, b_cover, diameter, t_cover, pitch, structure = None, facename
00168     = None):
00169         """ makeHelicalRebar(SideCover, BottomCover, Diameter, TopCover, Pitch, Structure, Facename):
00170             Adds the Helical reinforcement bar to the selected structural object."""
00171         if not structure and not facename:
00172             selected_obj = FreeCADGui.Selection.getSelectionEx()[0]
00173             structure = selected_obj.Object
00174             facename = selected_obj.SubElementNames[0]
00175             face = structure.Shape.Faces[getFaceNumber(facename) - 1]
00176             #StructurePRM = getTrueParametersOfStructure(structure)
00177             FacePRM = getParametersOfFace(structure, facename, False)
00178             if not FacePRM:
00179                 FreeCAD.Console.PrintError("Cannot identified shape or from which base object sturctural element is
00180 derived\n")
00181             return
00182             size = (ArchCommands.projectToVector(structure.Shape.copy(), face.normalAt(0, 0))).Length
00183             normal = face.normalAt(0,0)
00184             #normal = face.Placement.Rotation.inverted().multVec(normal)
00185             import Arch
00186             helix = createHelicalWire(FacePRM, s_cover, b_cover, t_cover, pitch, size, normal)
00187             helix.Support = [(structure, facename)]
00188             rebar = Arch.makeRebar(structure, helix, diameter, 1, 0)
00189             rebar.OffsetStart = 0
00190             rebar.OffsetEnd = 0
00191             FreeCAD.ActiveDocument.recompute()
00192             # Adds properties to the rebar object
00193             rebar.ViewObject.addProperty("App::PropertyString", "RebarShape", "RebarDialog", QT_TRANSLATE_NOOP("App::Property",
00194             "Shape of rebar")).RebarShape = "HelicalRebar"
00195             rebar.ViewObject.setEditorMode("RebarShape", 2)
00196             rebar.addProperty("App::PropertyDistance", "SideCover", "RebarDialog", QT_TRANSLATE_NOOP("App::Property",
00197             "Front cover of rebar")).SideCover = s_cover
00198             rebar.setEditorMode("SideCover", 2)
00199             rebar.addProperty("App::PropertyDistance", "Pitch", "RebarDialog", QT_TRANSLATE_NOOP("App::Property",
00200             "Left Side cover of rebar")).Pitch = pitch
00201             rebar.setEditorMode("Pitch", 2)
00202             rebar.addProperty("App::PropertyDistance", "BottomCover", "RebarDialog", QT_TRANSLATE_NOOP("App::Property",
00203             "Bottom cover of rebar")).BottomCover = b_cover
00204             rebar.setEditorMode("BottomCover", 2)
00205             rebar.addProperty("App::PropertyDistance", "TopCover", "RebarDialog", QT_TRANSLATE_NOOP("App::Property",
00206             "Top cover of rebar")).TopCover = t_cover
00207             rebar.setEditorMode("TopCover", 2)
00208             rebar.Label = "HelicalRebar"
00209             FreeCAD.ActiveDocument.recompute()
00210             return rebar
00211
00212
00213     def editHelicalRebar(Rebar, s_cover, b_cover, diameter, t_cover, pitch, structure = None,
00214     facename = None):
00215         sketch = Rebar.Base
00216         if structure and facename:
00217             sketch.Support = [(structure, facename)]
00218             # Check if sketch support is empty.
00219             if not sketch.Support:

```

```

00209     showWarning("You have checked remove external geometry of base sketches when needed.\nTo
00210     uncheck Edit->Preferences->Arch.")
00211     return
00212     # Assigned values
00213     facename = sketch.Support[0][1][0]
00214     structure = sketch.Support[0][0]
00215     face = structure.Shape.Faces[getFaceNumber(facename) - 1]
00216     #StructurePRM = getTrueParametersOfStructure(structure)
00217     # Get parameters of the face where sketch of rebar is drawn
00218     FacePRM = getParametersOfFace(structure, facename, False)
00219     size = (ArchCommands.projectToVector(structure.Shape.copy(), face.normalAt(0, 0))).Length
00220     normal = face.normalAt(0, 0)
00221     #normal = face.Placement.Rotation.inverted().multVec(normal)
00222     helix = createHelicalWire(FacePRM, s_cover, b_cover, t_cover, pitch, size, normal,
00223     Rebar.Base)
00224     FreeCAD.ActiveDocument.recompute()
00225     Rebar.Diameter = diameter
00226     Rebar.SideCover = s_cover
00227     Rebar.BottomCover = b_cover
00228     Rebar.TopCover = t_cover
00229     Rebar.Pitch = pitch
00230     FreeCAD.ActiveDocument.recompute()
00231     return Rebar
00232
00233 def editDialog(vobj):
00234     FreeCADGui.Control.closeDialog()
00235     obj = _HelicalRebarTaskPanel(vobj.Object)
00236     obj.form.sideCover.setText(str(vobj.Object.SideCover))
00237     obj.form.bottomCover.setText(str(vobj.Object.BottomCover))
00238     obj.form.diameter.setText(str(vobj.Object.Diameter))
00239     obj.form.topCover.setText(str(vobj.Object.TopCover))
00240     obj.form.pitch.setText(str(vobj.Object.Pitch))
00241     FreeCADGui.Control.showDialog(obj)
00242
00243 def CommandHelicalRebar():
00244     selected_obj = check_selected_face()
00245     if selected_obj:
00246         FreeCADGui.Control.showDialog(_HelicalRebarTaskPanel())

```

8.5 LShapeRebar.py File Reference

Classes

- class [LShapeRebar._LShapeRebarTaskPanel](#)

Namespaces

- [LShapeRebar](#)

Functions

- def [LShapeRebar.getpointsOfLShapeRebar](#) (FacePRM, l_cover, r_cover, b_cover, t_cover, orientation)
- def [LShapeRebar.makeLShapeRebar](#) (f_cover, b_cover, l_cover, r_cover, diameter, t_cover, rounding, amount_spacing_check, amount_spacing_value, orientation="Bottom Left", structure=None, facename=None)
- def [LShapeRebar.editLShapeRebar](#) (Rebar, f_cover, b_cover, l_cover, r_cover, diameter, t_cover, rounding, amount_spacing_check, amount_spacing_value, orientation, structure=None, facename=None)
- def [LShapeRebar.editDialog](#) (vobj)
- def [LShapeRebar.CommandLShapeRebar](#) ()

Variables

- string [LShapeRebar.__title__](#) = "LShapeRebar"
- string [LShapeRebar.__author__](#) = "Amritpal Singh"
- string [LShapeRebar.__url__](#) = "https://www.freecadweb.org"

8.6 LShapeRebar.py

```

00001 # --- coding: utf-8 ---
00002 # ****
00003 # *
00004 # * Copyright (c) 2017 - Amritpal Singh <amrit3701@gmail.com>
00005 # *
00006 # * This program is free software; you can redistribute it and/or modify
00007 # * it under the terms of the GNU Lesser General Public License (LGPL)
00008 # * as published by the Free Software Foundation; either version 2 of
00009 # * the License, or (at your option) any later version.
00010 # * for detail see the LICENCE text file.
00011 # *
00012 # * This program is distributed in the hope that it will be useful,
00013 # * but WITHOUT ANY WARRANTY; without even the implied warranty of
00014 # * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
00015 # * GNU Library General Public License for more details.
00016 # *
00017 # * You should have received a copy of the GNU Library General Public
00018 # * License along with this program; if not, write to the Free Software
00019 # * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307
00020 # * USA
00021 # *
00022 # ****
00023
00024 __title__ = "LShapeRebar"
00025 __author__ = "Amritpal Singh"
00026 __url__ = "https://www.freecadweb.org"
00027
00028 from PySide import QtCore, QtGui
00029 from Rebarfunc import *
00030 from PySide.QtCore import QT_TRANSLATE_NOOP
00031 from RebarDistribution import runRebarDistribution, removeRebarDistribution
00032 from PopUpImage import showPopUpImageDialog
00033 import FreeCAD
00034 import FreeCADGui
00035 import ArchCommands
00036 import os
00037 import sys
00038 import math
00039
00040 def getpointsOfLShapeRebar(FacePRM, l_cover, r_cover, b_cover, t_cover, orientation):
00041     """ getpointsOfLShapeRebar(FacePRM, LeftCover, RightCover, BottomCover, TopCover, Orientation):
00042     Return points of the LShape rebar in the form of array for sketch.
00043     It takes four different orientations input i.e. 'Bottom Left', 'Bottom Right', 'Top Left', 'Top Right'
00044     """
00045     if orientation == "Bottom Left":
00046         x1 = FacePRM[1][0] - FacePRM[0][0] / 2 + l_cover
00047         y1 = FacePRM[1][1] + FacePRM[0][1] / 2 - t_cover
00048         x2 = FacePRM[1][0] - FacePRM[0][0] / 2 + l_cover
00049         y2 = FacePRM[1][1] - FacePRM[0][1] / 2 + b_cover
00050         x3 = FacePRM[1][0] - FacePRM[0][0] / 2 + FacePRM[0][0] - r_cover
00051         y3 = FacePRM[1][1] - FacePRM[0][1] / 2 + b_cover
00052     elif orientation == "Bottom Right":
00053         x1 = FacePRM[1][0] - FacePRM[0][0] / 2 + FacePRM[0][0] - r_cover
00054         y1 = FacePRM[1][1] + FacePRM[0][1] / 2 - t_cover
00055         x2 = FacePRM[1][0] - FacePRM[0][0] / 2 + FacePRM[0][0] - r_cover
00056         y2 = FacePRM[1][1] - FacePRM[0][1] / 2 + b_cover
00057         x3 = FacePRM[1][0] - FacePRM[0][0] / 2 + l_cover
00058         y3 = FacePRM[1][1] - FacePRM[0][1] / 2 + b_cover
00059     elif orientation == "Top Left":
00060         x1 = FacePRM[1][0] - FacePRM[0][0] / 2 + l_cover
00061         y1 = FacePRM[1][1] - FacePRM[0][1] / 2 + b_cover
00062         x2 = FacePRM[1][0] - FacePRM[0][0] / 2 + l_cover
00063         y2 = FacePRM[1][1] + FacePRM[0][1] / 2 - t_cover
00064         x3 = FacePRM[1][0] - FacePRM[0][0] / 2 + FacePRM[0][0] - r_cover
00065         y3 = FacePRM[1][1] + FacePRM[0][1] / 2 - t_cover
00066     elif orientation == "Top Right":
00067         x1 = FacePRM[1][0] - FacePRM[0][0] / 2 + FacePRM[0][0] - r_cover
00068         y1 = FacePRM[1][1] - FacePRM[0][1] / 2 + b_cover
00069         x2 = FacePRM[1][0] - FacePRM[0][0] / 2 + FacePRM[0][0] - r_cover
00070         y2 = FacePRM[1][1] + FacePRM[0][1] / 2 - t_cover
00071         x3 = FacePRM[1][0] - FacePRM[0][0] / 2 + l_cover
00072         y3 = FacePRM[1][1] + FacePRM[0][1] / 2 - t_cover
00073     return [FreeCAD.Vector(x1, y1, 0), FreeCAD.Vector(x2, y2, 0), \
00074             FreeCAD.Vector(x3, y3, 0)]
00075
00076 class LShapeRebarTaskPanel:
00077     def __init__(self, Rebar = None):
00078         self.CustomSpacing = None
00079         if not Rebar:
00080             selected_obj = FreeCADGui.Selection.getSelectionEx()[0]
00081             self.SelectedObj = selected_obj.Object
00082             self.FaceName = selected_obj.SubElementNames[0]
00083         else:

```

```

00084         self.FaceName = Rebar.Base.Support[0][1][0]
00085         self.SelectedObj = Rebar.Base.Support[0][0]
00086         self.form = FreeCADGui.PySideUic.loadUi(os.path.splitext(__file__)[0] + ".ui")
00087         self.form.setWindowTitle(QtGui.QApplication.translate("RebarAddon", "L-Shape Rebar", None))
00088         self.form.orientation.addItems(["Bottom Right", "Bottom Left", "Top Right", "Top Left"])
00089         self.form.amount_radio.clicked.connect(self.amount_radio_clicked)
00090         self.form.spacing_radio.clicked.connect(self.spacing_radio_clicked)
00091         self.form.customSpacing.clicked.connect(lambda: runRebarDistribution(self))
00092         self.form.removeCustomSpacing.clicked.connect(lambda:
00093             removeRebarDistribution(self))
00093         self.form.PickSelectedFace.clicked.connect(lambda: getSelectedFace(self))
00094         self.form.orientation.currentIndexChanged.connect(self.getOrientation)
00095         self.form.image.setPixmap(QtGui.QPixmap(os.path.split(os.path.abspath(__file__))[0] + "
00096             /icons/LShapeRebarBR.svg"))
00096         self.form.toolButton.setIcon(self.form.toolButton.style().standardIcon(
00097             QtGui.QStyle.SP_DialogHelpButton))
00098         self.form.toolButton.clicked.connect(lambda: showPopUpImageDialog(os.path.split(
00099             os.path.abspath(__file__))[0] + "/icons/LShapeRebarDetailed.svg"))
00100     self.Rebar = Rebar
00101
00102     def getOrientation(self):
00103         orientation = self.form.orientation.currentText()
00104         if orientation == "Bottom Right":
00105             self.form.image.setPixmap(QtGui.QPixmap(os.path.split(os.path.abspath(__file__))[0] + "
00106                 /icons/LShapeRebarBR.svg"))
00107             elif orientation == "Bottom Left":
00108                 self.form.image.setPixmap(QtGui.QPixmap(os.path.split(os.path.abspath(__file__))[0] + "
00109                     /icons/LShapeRebarBL.svg"))
00110             elif orientation == "Top Right":
00111                 self.form.image.setPixmap(QtGui.QPixmap(os.path.split(os.path.abspath(__file__))[0] + "
00112                     /icons/LShapeRebarTR.svg"))
00113             else:
00114                 self.form.image.setPixmap(QtGui.QPixmap(os.path.split(os.path.abspath(__file__))[0] + "
00115                     /icons/LShapeRebarTL.svg"))
00116
00117     def getStandardButtons(self):
00118         return int(QtGui.QDialogButtonBox.Ok) | int(QtGui.QDialogButtonBox.Apply) | int(
00119             QtGui.QDialogButtonBox.Cancel)
00120
00121     def clicked(self, button):
00122         if button == int(QtGui.QDialogButtonBox.Apply):
00123             self.accept(button)
00124
00125     def accept(self, signal = None):
00126         f_cover = self.form.frontCover.text()
00127         f_cover = FreeCAD.Units.Quantity(f_cover).Value
00128         b_cover = self.form.bottomCover.text()
00129         b_cover = FreeCAD.Units.Quantity(b_cover).Value
00130         l_cover = self.form.l_sideCover.text()
00131         l_cover = FreeCAD.Units.Quantity(l_cover).Value
00132         r_cover = self.form.r_sideCover.text()
00133         r_cover = FreeCAD.Units.Quantity(r_cover).Value
00134         t_cover = self.form.topCover.text()
00135         t_cover = FreeCAD.Units.Quantity(t_cover).Value
00136         diameter = self.form.diameter.text()
00137         diameter = FreeCAD.Units.Quantity(diameter).Value
00138         rounding = self.form.rounding.value()
00139         orientation = self.form.orientation.currentText()
00140         amount_check = self.form.amount_radio.isChecked()
00141         spacing_check = self.form.spacing_radio.isChecked()
00142         if not self.Rebar:
00143             if amount_check:
00144                 amount = self.form.amount.value()
00145                 rebar = makeLShapeRebar(f_cover, b_cover, l_cover, r_cover, diameter,
00146                     t_cover, rounding, True, amount, orientation, self.SelectedObj, self.
00147                         FaceName)
00148                 elif spacing_check:
00149                     spacing = self.form.spacing.text()
00150                     spacing = FreeCAD.Units.Quantity(spacing).Value
00151                     rebar = makeLShapeRebar(f_cover, b_cover, l_cover, r_cover, diameter,
00152                         t_cover, rounding, False, spacing, orientation, self.SelectedObj, self.
00153                             FaceName)
00154             else:
00155                 if amount_check:
00156                     amount = self.form.amount.value()
00157                     rebar = editLShapeRebar(self.Rebar, f_cover, b_cover, l_cover, r_cover,
00158                         diameter, t_cover, rounding, True, amount, orientation, self.SelectedObj, self.
00159                             FaceName)
00160                 elif spacing_check:
00161                     spacing = self.form.spacing.text()
00162                     spacing = FreeCAD.Units.Quantity(spacing).Value
00163                     rebar = editLShapeRebar(self.Rebar, f_cover, b_cover, l_cover, r_cover,
00164                         diameter, t_cover, rounding, False, spacing, orientation, self.SelectedObj, self.
00165                             FaceName)
00166         if self.CustomSpacing:
00167             rebar.CustomSpacing = self.CustomSpacing
00168             FreeCAD.ActiveDocument.recompute()

```

```

00154     self.Rebar = rebar
00155     if signal == int(QtGui.QDialogButtonBox.Apply):
00156         pass
00157     else:
00158         FreeCADGui.Control.closeDialog(self)
00159
00160     def amount_radio_clicked(self):
00161         self.form.spacing.setEnabled(False)
00162         self.form.amount.setEnabled(True)
00163
00164     def spacing_radio_clicked(self):
00165         self.form.amount.setEnabled(False)
00166         self.form.spacing.setEnabled(True)
00167
00168
00169     def makeLShapeRebar(f_cover, b_cover, l_cover, r_cover, diameter, t_cover, rounding,
00170         amount_spacing_check, amount_spacing_value, orientation = "Bottom Left", structure = None, facename = None):
00171         """ makeLShapeRebar(FrontCover, BottomCover, LeftCover, RightCover, Diameter, TopCover, Rounding,
00172             AmountSpacingCheck, AmountSpacingValue,
00173             Orientation, Structure, Facename): Adds the L-Shape reinforcement bar to the selected structural
00174             object.
00175             It takes four different orientations input i.e. 'Bottom Left', 'Bottom Right ', 'Top Left', 'Top Right'
00176
00177             """
00178         if not structure and not facename:
00179             selected_obj = FreeCADGui.Selection.getSelectionEx()[0]
00180             structure = selected_obj.Object
00181             facename = selected_obj.SubElementNames[0]
00182             face = structure.Shape.Faces[getFaceNumber(facename) - 1]
00183             #StructurePRM = getTrueParametersOfStructure(structure)
00184             FacePRM = getParametersOfFace(structure, facename)
00185             if not FacePRM:
00186                 FreeCAD.Console.PrintError("Cannot identified shape or from which base object sturctural element is
00187                 derived\n")
00188                 return
00189             # Get points of L-Shape rebar
00190             points = getpointsOfLShapeRebar(FacePRM, l_cover, r_cover, b_cover, t_cover,
00191                 orientation)
00192             import Part
00193             import Arch
00194             sketch = FreeCAD.activeDocument().addObject('Sketcher::SketchObject', 'Sketch')
00195             sketch.MapMode = "FlatFace"
00196             sketch.Support = [(structure, facename)]
00197             FreeCAD.ActiveDocument.recompute()
00198             sketch.addGeometry(Part.LineSegment(points[0], points[1]), False)
00199             sketch.addGeometry(Part.LineSegment(points[1], points[2]), False)
00200             import Sketcher
00201             if amount_spacing_check:
00202                 rebar = Arch.makeRebar(structure, sketch, diameter, amount_spacing_value, f_cover)
00203                 FreeCAD.ActiveDocument.recompute()
00204             else:
00205                 size = (ArchCommands.projectToVector(structure.Shape.copy(), face.normalAt(0, 0))).Length
00206                 rebar = Arch.makeRebar(structure, sketch, diameter, int((size - diameter) / amount_spacing_value),
00207                     f_cover)
00208                 rebar.Rounding = rounding
00209                 # Adds properties to the rebar object
00210                 rebar.ViewObject.addProperty("App::PropertyString", "RebarShape", "RebarDialog", QT_TRANSLATE_NOOP("App::Property",
00211                     "Shape of rebar")).RebarShape = "LShapeRebar"
00212                 rebar.ViewObject.setEditorMode("RebarShape", 2)
00213                 rebar.addProperty("App::PropertyDistance", "FrontCover", "RebarDialog", QT_TRANSLATE_NOOP("App::Property",
00214                     "Front cover of rebar")).FrontCover = f_cover
00215                 rebar.setEditorMode("FrontCover", 2)
00216                 rebar.addProperty("App::PropertyDistance", "LeftCover", "RebarDialog", QT_TRANSLATE_NOOP("App::Property",
00217                     "Left Side cover of rebar")).LeftCover = l_cover
00218                 rebar.setEditorMode("LeftCover", 2)
00219                 rebar.addProperty("App::PropertyDistance", "RightCover", "RebarDialog", QT_TRANSLATE_NOOP("App::Property",
00220                     "Right Side cover of rebar")).RightCover = r_cover
00221                 rebar.setEditorMode("RightCover", 2)
00222                 rebar.addProperty("App::PropertyDistance", "BottomCover", "RebarDialog", QT_TRANSLATE_NOOP("App::Property",
00223                     "Bottom cover of rebar")).BottomCover = b_cover
00224                 rebar.setEditorMode("BottomCover", 2)
00225                 rebar.addProperty("App::PropertyBool", "AmountCheck", "RebarDialog", QT_TRANSLATE_NOOP("App::Property",
00226                     "Amount radio button is checked")).AmountCheck
00227                 rebar.setEditorMode("AmountCheck", 2)
00228                 rebar.addProperty("App::PropertyDistance", "TopCover", "RebarDialog", QT_TRANSLATE_NOOP("App::Property",
00229                     "Top cover of rebar")).TopCover = t_cover
00230                 rebar.setEditorMode("TopCover", 2)
00231                 rebar.addProperty("App::PropertyDistance", "TrueSpacing", "RebarDialog", QT_TRANSLATE_NOOP("App::Property",
00232                     "Spacing between of rebars")).TrueSpacing = amount_spacing_value
00233                 rebar.addProperty("App::PropertyString", "Orientation", "RebarDialog", QT_TRANSLATE_NOOP("App::Property",
00234                     "Shape of rebar")).Orientation = orientation
00235                 rebar.setEditorMode("Orientation", 2)
00236                 rebar.setEditorMode("TrueSpacing", 2)
00237                 if amount_spacing_check:
00238                     rebar.AmountCheck = True
00239                 else:
00240                     rebar.AmountCheck = False

```

```

00225     rebar.TrueSpacing = amount_spacing_value
00226     rebar.Label = "LShapeRebar"
00227     FreeCAD.ActiveDocument.recompute()
00228     return rebar
00229
00230 def editLShapeRebar(Rebar, f_cover, b_cover, l_cover, r_cover, diameter, t_cover, rounding,
00231     amount_spacing_check, amount_spacing_value, orientation, structure = None, facename = None):
00232     sketch = Rebar.Base
00233     if structure and facename:
00234         sketch.Support = [(structure, facename)]
00235     # Check if sketch support is empty.
00236     if not sketch.Support:
00237         showWarning("You have checked remove external geometry of base sketchs when needed.\nTo
unchecked Edit->Preferences->Arch.")
00238     return
00239     # Assigned values
00240     facename = sketch.Support[0][1][0]
00241     structure = sketch.Support[0][0]
00242     face = structure.Shape.Faces[getFaceNumber(facename) - 1]
00243     #StructurePRM = getTrueParametersOfStructure(structure)
00244     # Get parameters of the face where sketch of rebar is drawn
00245     FacePRM = getParametersOfFace(structure, facename)
00246     # Get points of L-Shape rebar
00247     points = getpointsOfLShapeRebar(FacePRM, l_cover, r_cover, b_cover, t_cover,
00248         orientation)
00249     sketch.movePoint(0, 1, points[0], 0)
00250     FreeCAD.ActiveDocument.recompute()
00251     sketch.movePoint(0, 2, points[1], 0)
00252     FreeCAD.ActiveDocument.recompute()
00253     sketch.movePoint(1, 1, points[1], 0)
00254     FreeCAD.ActiveDocument.recompute()
00255     sketch.movePoint(1, 2, points[2], 0)
00256     FreeCAD.ActiveDocument.recompute()
00257     Rebar.OffsetStart = f_cover
00258     Rebar.OffsetEnd = f_cover
00259     if amount_spacing_check:
00260         Rebar.Amount = amount_spacing_value
00261         FreeCAD.ActiveDocument.recompute()
00262         Rebar.AmountCheck = True
00263     else:
00264         size = (ArchCommands.projectToVector(structure.Shape.copy(), face.normalAt(0, 0))).Length
00265         Rebar.Amount = int((size - diameter) / amount_spacing_value)
00266         FreeCAD.ActiveDocument.recompute()
00267         Rebar.AmountCheck = False
00268     Rebar.Diameter = diameter
00269     Rebar.FrontCover = f_cover
00270     Rebar.LeftCover = l_cover
00271     Rebar.RightCover = r_cover
00272     Rebar.BottomCover = b_cover
00273     Rebar.TopCover = t_cover
00274     Rebar.Rounding = rounding
00275     Rebar.TrueSpacing = amount_spacing_value
00276     Rebar.Orientation = orientation
00277     FreeCAD.ActiveDocument.recompute()
00278     return Rebar
00279
00280 def editDialog(vobj):
00281     FreeCADGui.Control.closeDialog()
00282     obj = _LShapeRebarTaskPanel(vobj.Object)
00283     obj.form.frontCover.setText(str(vobj.Object.FrontCover))
00284     obj.form.l_sideCover.setText(str(vobj.Object.LeftCover))
00285     obj.form.r_sideCover.setText(str(vobj.Object.RightCover))
00286     obj.form.bottomCover.setText(str(vobj.Object.BottomCover))
00287     obj.form.diameter.setText(str(vobj.Object.Diameter))
00288     obj.form.topCover.setText(str(vobj.Object.TopCover))
00289     obj.form.rounding.setValue(vobj.Object.Rounding)
00290     obj.form.orientation.setCurrentIndex(obj.form.orientation.findText(str(vobj.Object.Orientation)))
00291     if vobj.Object.AmountCheck:
00292         obj.form.amount.setValue(vobj.Object.Amount)
00293     else:
00294         obj.form.amount_radio.setChecked(False)
00295         obj.form.spacing_radio.setChecked(True)
00296         obj.form.amount.setDisabled(True)
00297         obj.form.spacing.setEnabled(True)
00298         obj.form.spacing.setText(str(vobj.Object.TrueSpacing))
00299     #obj.form.PickSelectedFace.setVisible(False)
00300     FreeCADGui.Control.showDialog(obj)
00301
00302 def CommandLShapeRebar():
00303     selected_obj = check_selected_face()
00304     if selected_obj:
00305         FreeCADGui.Control.showDialog(_LShapeRebarTaskPanel())

```

8.7 PopUpImage.py File Reference

Classes

- class [PopUpImage.PopUpImage](#)

Namespaces

- [PopUpImage](#)

Functions

- def [PopUpImage.showPopUpImageDialog](#) (img)

Variables

- string [PopUpImage.__title__](#) = "PopUpImage"
- string [PopUpImage.__author__](#) = "Amritpal Singh"
- string [PopUpImage.__url__](#) = "https://www.freecadweb.org"

8.8 PopUpImage.py

```

00001 # -*- coding: utf-8 -*-
00002 # ****
00003 # *
00004 # * Copyright (c) 2017 - Amritpal Singh <amrit3701@gmail.com>
00005 # *
00006 # * This program is free software; you can redistribute it and/or modify
00007 # * it under the terms of the GNU Lesser General Public License (LGPL)
00008 # * as published by the Free Software Foundation; either version 2 of
00009 # * the License, or (at your option) any later version.
00010 # * for detail see the LICENCE text file.
00011 # *
00012 # * This program is distributed in the hope that it will be useful,
00013 # * but WITHOUT ANY WARRANTY; without even the implied warranty of
00014 # * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
00015 # * GNU Library General Public License for more details.
00016 # *
00017 # * You should have received a copy of the GNU Library General Public
00018 # * License along with this program; if not, write to the Free Software
00019 # * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307
00020 # * USA
00021 # *
00022 # ****
00023
00024 __title__ = "PopUpImage"
00025 __author__ = "Amritpal Singh"
00026 __url__ = "https://www.freecadweb.org"
00027
00028
00029 from PySide import QtCore
00030 from PySide import QtGui
00031 from PySide import QtSvg
00032 import FreeCADGui
00033 import os
00034
00035 class PopUpImage(QtGui.QDialog):
00036     def __init__(self, img):
00037         QtGui.QDialog.__init__(self)
00038         self.image = QtSvg.QSvgWidget(img)
00039         self.setWindowTitle(QtGui.QApplication.translate("RebarTool", "Detailed description", None))
00040         self.verticalLayout = QtGui.QVBoxLayout(self)
00041         self.verticalLayout.addWidget(self.image)
00042
00043 def showPopUpImageDialog(img):
00044     """ showPopUpImageDialog(image): This function will show a given image in a pop-up
00045     dialog box."""
00046     dialog = PopUpImage(img)
00047     dialog.exec_()

```

8.9 README.md File Reference

8.10 README.md

```

00001 # Rebar Addon for FreeCAD
00002
00003 Started as a Google Summer of Code ([GSoC](https://en.wikipedia.org/wiki/Google_Summer_of_Code) 2017)
[project](https://summerofcode.withgoogle.com/archive/2017/projects/6536382147198976).
00004
00005 ! [Screenshot] (http://i.imgur.com/r9b5l7K.jpg)
00006
00007 ## Documentation
00008 This project is aimed at easing up the process of rebaring in [FreeCAD](https://www.freecadweb.org).
In this project, list of rebars will be provided to user under Rebar tools in the form of dropdown. This
project covers six different rebar shapes as given below:
00009
00010 -
![icon] (https://www.freecadweb.org/wiki/images/thumb/6/69/Arch_Rebar_Straight.png/32px-Arch_Rebar_Straight.png) **Straight
00011 ! [Screenshot] (https://www.freecadweb.org/wiki/images/f/fd/StraightRebar.png)
00012
00013 -
![icon] (https://www.freecadweb.org/wiki/images/thumb/4/4d/Arch_Rebar_UShape.png/32px-Arch_Rebar_UShape.png) **UShape Rebar
00014 ! [Screenshot] (https://www.freecadweb.org/wiki/images/3/35/Footing_UShapeRebar.png)
00015
00016 -
![icon] (https://www.freecadweb.org/wiki/images/thumb/3/38/Arch_Rebar_LShape.png/32px-Arch_Rebar_LShape.png) **LShape Rebar
00017 ! [Screenshot] (https://www.freecadweb.org/wiki/images/1/10/LShapeRebarNew.png)
00018
00019 -
![icon] (https://www.freecadweb.org/wiki/images/thumb/0/0b/Arch_Rebar_BentShape.png/32px-Arch_Rebar_BentShape.png) **Bent
00020 ! [Screenshot] (https://www.freecadweb.org/wiki/images/e/e3/BentShapeRebar.png)
00021
00022 -
![icon] (https://www.freecadweb.org/wiki/images/thumb/e/ef/Arch_Rebar_Stirrup.png/32px-Arch_Rebar_Stirrup.png) **Stirrup
00023 ! [Screenshot] (https://www.freecadweb.org/wiki/images/9/9b/Stirrup.png)
00024
00025 -
![icon] (https://www.freecadweb.org/wiki/images/thumb/c/c9/Arch_Rebar_Helical.png/32px-Arch_Rebar_Helical.png) **Helical
00026 ! [Screenshot] (https://www.freecadweb.org/wiki/images/2/2f/HelicalRebar.png)
00027
00028 ## Video Tutorial
00029 ! [IMAGE ALT TEXT
HERE] (http://i.imgur.com/ZQGCQoe.png)] (https://www.youtube.com/watch?v=BYQQjEKmx5E&t=1435s)
00030
00031
00032 ## Installation
00033
00034 ### Pre-requisites
00035 - FreeCAD (version >= 0.17): [Installation guide](https://www.freecadweb.org/wiki/Installing)
00036
00037 ### Steps to install Rebar Addon in FreeCAD
00038 1. Open the FreeCAD Addon Manager (''Tool -> Addon manager'').
00039 2. When an addon manager will open, select ''Reinforcement'' from a list of workbenches shown by an
addon manager.
00040 3. After selecting, click on ''Install/Update'' button.
00041 4. Restart FreeCAD.
00042 5. Now you will see different rebars in a drop-down list of rebar tools (''Arch -> Rebar tools ->
Different rebars'').
00043
00044 ## How it works
00045 Each rebar tool has two files, one is ''Python'' file and second is there respective name ''UI'' file
like ''StraightRebar.py'' and ''StraightRebar.ui'' file). Let's take a straight rebar tool. In
''StraightRebar.py'' file, there are two functions. One is ''makeStraightRebar()'' function. This function
creates straight rebar and adds new properties to the default ''Rebar'' object. Second function is
''editStraightRebar''. This function is used when we want to change a new properties(which is created by
''makeStraightRebar'' function) of the rebar object and it will take ''Rebar'' object as input which is created by
''makeStraightRebar'' function. In ''StraightRebar.py'', ''_StraightRebarTaskPanel'' class is
present. This class loads UI(present in ''StraightRebar.ui'' file) in the task panel of FreeCAD. First time when
a user clicks on ''Apply'' or ''Ok'' button, then ''makeStraightRebar'' function is executed and after
that when user want to change the properties of Straight rebar then ''editStraightRebar'' function is
executed.
00046
00047 ## Extras
00048 - [FreeCAD forum thread](https://forum.freecadweb.org/viewtopic.php?f=8&t=22760)
00049 - [GSoC proposal](https://brlcad.org/wiki/User:Amritpal_singh/gsoc_proposal)
00050 - [Development logs](https://brlcad.org/wiki/User:Amritpal_singh/GSoC17/logs)

```

8.11 RebarDistribution.py File Reference

Classes

- class `RebarDistribution._RebarDistributionDialog`

Namespaces

- `RebarDistribution`

Functions

- def `RebarDistribution.getCustomSpacingString` (amount1, spacing1, amount2, spacing2, amount3, spacing3, frontCover, size)
- def `RebarDistribution.gettupleOfCustomSpacing` (span_string)
- def `RebarDistribution.runRebarDistribution` (self)
- def `RebarDistribution.removeRebarDistribution` (self)

Variables

- string `RebarDistribution.__title__` = "DialogDistribution"
- string `RebarDistribution.__author__` = "Amritpal Singh"
- string `RebarDistribution.__url__` = "https://www.freecadweb.org"
- `RebarDistribution.CustomSpacing`

8.12 RebarDistribution.py

```

00001 # -*- coding: utf-8 -*-
00002 # ****
00003 # *
00004 # * Copyright (c) 2017 - Amritpal Singh <amrit3701@gmail.com>
00005 # *
00006 # * This program is free software; you can redistribute it and/or modify
00007 # * it under the terms of the GNU Lesser General Public License (LGPL)
00008 # * as published by the Free Software Foundation; either version 2 of
00009 # * the License, or (at your option) any later version.
00010 # * for detail see the LICENCE text file.
00011 # *
00012 # * This program is distributed in the hope that it will be useful,
00013 # * but WITHOUT ANY WARRANTY; without even the implied warranty of
00014 # * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
00015 # * GNU Library General Public License for more details.
00016 # *
00017 # * You should have received a copy of the GNU Library General Public
00018 # * License along with this program; if not, write to the Free Software
00019 # * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307
00020 # * USA
00021 # *
00022 # ****
00023
00024 __title__ = "DialogDistribution"
00025 __author__ = "Amritpal Singh"
00026 __url__ = "https://www.freecadweb.org"
00027
00028 from PySide import QtCore, QtGui
00029 from Rebarfunc import *
00030 from PySide.QtCore import QT_TRANSLATE_NOOP
00031 import FreeCAD
00032 import FreeCADGui
00033 import ArchCommands
00034 import os
00035 import sys
00036 import math
00037
00038 class _RebarDistributionDialog():
00039     def __init__(self, frontCover, size):
00040         self.FrontCover = frontCover

```

```

00041         self.ExpandingLength = size
00042         self.form = FreeCADGui.PySideUic.loadUi(os.path.splitext(__file__)[0] + ".ui")
00043         self.form.setWindowTitle(QtGui.QApplication.translate("Arch", "Rebar Distribution", None))
00044         self.form.image.setPixmap(QtGui.QPixmap(os.path.abspath(__file__))[0] + "
00045             /icons/RebarDistribution.svg"))
00046     def accept(self):
00047         amount1 = self.form.amount1.value()
00048         spacing1 = self.form.spacing1.text()
00049         spacing1 = FreeCAD.Units.Quantity(spacing1).Value
00050         amount2 = self.form.amount2.value()
00051         spacing2 = self.form.spacing2.text()
00052         spacing2 = FreeCAD.Units.Quantity(spacing2).Value
00053         amount3 = self.form.amount3.value()
00054         spacing3 = self.form.spacing3.text()
00055         spacing3 = FreeCAD.Units.Quantity(spacing3).Value
00056         self.CustomSpacing = getCustomSpacingString(amount1, spacing1,
00057             amount2, spacing2, amount3, spacing3, self.FrontCover, self.
00058             ExpandingLength)
00059     def setupUi(self):
00060         # Connect Signals and Slots
00061         self.form.buttonBox.accepted.connect(self.accept)
00062         pass
00063     def getCustomSpacingString(amount1, spacing1, amount2, spacing2, amount3, spacing3,
00064         frontCover, size):
00065         seg1_area = amount1 * spacing1 - spacing1 / 2
00066         seg3_area = amount3 * spacing3 - spacing3 / 2
00067         seg2_area = size - seg1_area - seg3_area - 2 * frontCover
00068         if seg2_area < 0:
00069             FreeCAD.Console.PrintError("Sum of length of segment 1 and segment 2 is greater than length of
00070             rebar expands.\n")
00071             return
00072         if spacing1 and spacing2 and spacing3 and amount1 and amount2 and amount3:
00073             pass
00074         else:
00075             if spacing1 and spacing2 and spacing3:
00076                 amount2 = math.ceil(seg2_area / spacing2)
00077                 spacing2 = seg2_area / amount2
00078             elif amount1 and amount2 and amount3:
00079                 spacing2 = math.floor(seg2_area / amount2)
00080             CustomSpacing = str(amount1) + "@" + str(spacing1) + "+" + str(int(amount2)) + "@" + str(spacing2) + "+"
00081             " + str(amount3) + "@" + str(spacing3)
00082             return CustomSpacing
00083     def gettupleOfCustomSpacing(span_string):
00084         """ gettupleOfCustomSpacing(span_string): This function take input
00085         in specific syntax and return output in the form of list. For eg.
00086         Input: "3@100+2@200+3@100"
00087         Output: [(3, 100), (2, 200), (3, 100)]"""
00088         import string
00089         span_st = string.strip(span_string)
00090         span_sp = string.split(span_st, '+')
00091         index = 0
00092         spacinglist = []
00093         while index < len(span_sp):
00094             # Find "@" recursively in span_sp array.
00095             in_sp = string.split(span_sp[index], '@')
00096             spacinglist.append((int(in_sp[0]), float(in_sp[1])))
00097             index += 1
00098         return spacinglist
00099     def runRebarDistribution(self):
00100         frontCover = self.form.frontCover.text()
00101         frontCover = FreeCAD.Units.Quantity(frontCover).Value
00102         face = self.SelectedObj.Shape.Faces[getFaceNumber(self.FaceName) - 1]
00103         size = (ArchCommands.projectToVector(self.SelectedObj.Shape.copy(), face.normalAt(0, 0))).Length
00104         dialog = _RebarDistributionDialog(frontCover, size)
00105         dialog.setupUi()
00106         dialog.form.exec_()
00107         self.CustomSpacing = dialog.CustomSpacing
00108     def removeRebarDistribution(self):
00109         self.CustomSpacing = ""
00110         self.Rebar.CustomSpacing = ""
00111         FreeCAD.ActiveDocument.recompute()
00112
00113 #runRebarDistribution(App.ActiveDocument.Rebar)

```

8.13 Rebarfunc.py File Reference

Namespaces

- `Rebarfunc`

Functions

- def `Rebarfunc.getEdgesAngle (edge1, edge2)`
- def `Rebarfunc.checkRectangle (edges)`
- def `Rebarfunc.getBaseStructuralObject (obj)`
- def `Rebarfunc.getBaseObject (obj)`
- def `Rebarfunc.getFaceNumber (s)`
- def `Rebarfunc.facenormalDirection (structure=None, facename=None)`
- def `Rebarfunc.getTrueParametersOfStructure (obj)`
- def `Rebarfunc.getParametersOfFace (structure, facename, sketch=True)`
- def `Rebarfunc.extendedTangentPartLength (rounding, diameter, angle)`
- def `Rebarfunc.extendedTangentLength (rounding, diameter, angle)`
- def `Rebarfunc.check_selected_face ()`
- def `Rebarfunc.getSelectedFace (self)`
- def `Rebarfunc.showWarning (message)`
- def `Rebarfunc.translate (context, text, disambig=None)`

Variables

- string `Rebarfunc.__title__ = "GenericRebarFuctions"`
- string `Rebarfunc.__author__ = "Amritpal Singh"`
- string `Rebarfunc.__url__ = "https://www.freecadweb.org"`
- `Rebarfunc.SelectedObj`
- `Rebarfunc.FaceName`

8.14 Rebarfunc.py

```

00001 # --- coding: utf-8 ---
00002 # ****
00003 # *
00004 # * Copyright (c) 2017 - Amritpal Singh <amrit3701@gmail.com>
00005 # *
00006 # * This program is free software; you can redistribute it and/or modify
00007 # * it under the terms of the GNU Lesser General Public License (LGPL)
00008 # * as published by the Free Software Foundation; either version 2 of
00009 # * the License, or (at your option) any later version.
00010 # * for detail see the LICENCE text file.
00011 # *
00012 # * This program is distributed in the hope that it will be useful,
00013 # * but WITHOUT ANY WARRANTY; without even the implied warranty of
00014 # * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
00015 # * GNU Library General Public License for more details.
00016 # *
00017 # * You should have received a copy of the GNU Library General Public
00018 # * License along with this program; if not, write to the Free Software
00019 # * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307
00020 # * USA
00021 # *
00022 # ****
00023
00024 __title__ = "GenericRebarFuctions"
00025 __author__ = "Amritpal Singh"
00026 __url__ = "https://www.freecadweb.org"
00027
00028 from PySide import QtCore, QtGui
00029 from DraftGeomUtils import vec, isCubic
00030 import FreeCAD
00031 import FreeCADGui

```

```

00032 import math
00033
00034 # -----
00035 # Generic functions
00036 # -----
00037
00038 def getEdgesAngle(edge1, edge2):
00039     """ getEdgesAngle(edge1, edge2): returns a angle between two edges."""
00040     vec1 = vec(edge1)
00041     vec2 = vec(edge2)
00042     angle = vec1.getAngle(vec2)
00043     angle = math.degrees(angle)
00044     return angle
00045
00046 def checkRectangle(edges):
00047     """ checkRectangle(edges[]): This function checks whether the given form rectangle
00048         or not. It will return True when edges form rectangular shape or return False
00049         when edges not form a rectangular."""
00050     angles = [round(getEdgesAngle(edges[0], edges[1])), round(
00051             getEdgesAngle(edges[0], edges[2])),
00052             round(getEdgesAngle(edges[0], edges[3]))]
00053     if angles.count(90) == 2 and (angles.count(180) == 1 or angles.count(0) == 1):
00054         return True
00055     else:
00056         return False
00057
00058 def getBaseStructuralObject(obj):
00059     """ getBaseStructuralObject(obj): This function will return last base
00060         structural object."""
00061     if not obj.Base:
00062         return obj
00063     else:
00064         return getBaseStructuralObject(obj.Base)
00065
00066 def getBaseObject(obj):
00067     """ getBaseObject(obj): This function will return last base object."""
00068     if hasattr(obj, "Base"):
00069         return getBaseObject(obj.Base)
00070     else:
00071         return obj
00072
00073 def getFaceNumber(s):
00074     """ getFaceNumber(facename): This will return a face number from face name.
00075     For eg.:
00076         Input: "Face12"
00077         Output: 12"""
00078     head = s.rstrip('0123456789')
00079     tail = s[len(head):]
00080     return int(tail)
00081
00082 def facenormalDirection(structure = None, facename = None):
00083     if not structure and not facename:
00084         selected_obj = FreeCADGui.Selection.getSelectionEx()[0]
00085         structure = selected_obj.Object
00086         facename = selected_obj.SubElementNames[0]
00087     face = structure.Shape.Faces[getFaceNumber(facename) - 1]
00088     normal = face.normalAt(0,0)
00089     normal = face.Placement.Rotation.inverted().multVec(normal)
00090     return normal
00091
00092 # -----
00093 # Main functions which is use while creating any rebar.
00094
00095 def getTrueParametersOfStructure(obj):
00096     """ getTrueParametersOfStructure(obj): This function return actual length,
00097         width and height of the structural element in the form of array like
00098         [Length, Width, Height]"""
00099     baseObject = getBaseObject(obj)
00100     # If selected_obj is not derived from any base object
00101     if baseObject:
00102         # If selected_obj is derived from SketchObject
00103         if baseObject.isDerivedFrom("Sketcher::SketchObject"):
00104             edges = baseObject.Shape.Edges
00105             if checkRectangle(edges):
00106                 for edge in edges:
00107                     # Representation vector of edge
00108                     rep_vector = edge.Vertexes[1].Point.sub(edge.Vertexes[0].Point)
00109                     rep_vector_angle = round(math.degrees(rep_vector.getAngle(FreeCAD.Vector(1,0,0))))
00110                     if rep_vector_angle in {0, 180}:
00111                         length = edge.Length
00112                     else:
00113                         width = edge.Length
00114             else:
00115                 return None
00116     else:
00117         return None

```

```

00118     height = obj.Height.Value
00119
00120     else:
00121         structuralBaseObject = getBaseStructuralObject(obj)
00122         length = structuralBaseObject.Length.Value
00123         width = structuralBaseObject.Width.Value
00124         height = structuralBaseObject.Height.Value
00125
00126     def getParametersOfFace(structure, facename, sketch = True):
00127         """ getParametersOfFace(structure, facename, sketch = True): This function will return
00128         length, width and points of center of mass of a given face according to the sketch
00129         value in the form of list.
00130
00131         For eg.:
00132         Case 1: When sketch is True: We use True when we want to create rebars from sketch
00133             (planar rebars) and the sketch is strictly based on 2D so we neglected the normal
00134             axis of the face.
00135             Output: [(FaceLength, FaceWidth), (CenterOfMassX, CenterOfMassY)]
00136
00137         Case 2: When sketch is False: When we want to create non-planar rebars(like stirrup)
00138             or we want to create rebar from a wire. Also for creating rebar from wire
00139             we will require three coordinates (x, y, z).
00140             Output: [(FaceLength, FaceWidth), (CenterOfMassX, CenterOfMassY, CenterOfMassZ)]"""
00141     face = structure.Shape.Faces[getFaceNumber(facename) - 1]
00142     center_of_mass = face.CenterOfMass
00143     #center_of_mass = center_of_mass.sub(getBaseStructuralObject(structure).Placement.Base)
00144     center_of_mass = center_of_mass.sub(structure.Placement.Base)
00145     Edges = []
00146     facePRM = []
00147     # When structure is cubic. It support all structure is derived from
00148     # any other object (like a sketch, wire etc).
00149     if isCubic(structure.Shape):
00150         print 423
00151         for edge in face.Edges:
00152             if not Edges:
00153                 Edges.append(edge)
00154             else:
00155                 # Checks whether similar edges is already present in Edges list
00156                 # or not.
00157                 if round((vec(edge)).Length) not in [round((vec(x)).Length) for x in Edges]:
00158                     Edges.append(edge)
00159             if len(Edges) == 1:
00160                 Edges.append(edge)
00161             # facePRM holds length of a edges.
00162             facePRM = [(vec(edge)).Length for edge in Edges]
00163             # Find the orientation of the face. Also eliminating normal axes
00164             # to the edge/face.
00165             # When edge is parallel to x-axis
00166             if round(Edges[0].tangentAt(0)[0]) in {1,-1}:
00167                 x = center_of_mass[0]
00168                 if round(Edges[1].tangentAt(0)[1]) in {1, -1}:
00169                     y = center_of_mass[1]
00170                 else:
00171                     y = center_of_mass[2]
00172             # When edge is parallel to y-axis
00173             elif round(Edges[0].tangentAt(0)[1]) in {1,-1}:
00174                 x = center_of_mass[1]
00175                 if round(Edges[1].tangentAt(0)[0]) in {1, -1}:
00176                     # Change order when edge along x-axis is at second place.
00177                     facePRM.reverse()
00178                     y = center_of_mass[1]
00179                 else:
00180                     y = center_of_mass[2]
00181             elif round(Edges[0].tangentAt(0)[2]) in {1,-1}:
00182                 y = center_of_mass[2]
00183                 if round(Edges[1].tangentAt(0)[0]) in {1, -1}:
00184                     x = center_of_mass[0]
00185                 else:
00186                     x = center_of_mass[1]
00187                 facePRM.reverse()
00188                 facelength = facePRM[0]
00189                 facewidth = facePRM[1]
00190             # When structure is not cubic. For founding parameters of given face
00191             # I have used bounding box.
00192             else:
00193                 boundbox = face.BoundBox
00194                 # Check that one length of bounding box is zero. Here bounding box
00195                 # looks like a plane.
00196                 if 0 in {round(boundbox.XLength), round(boundbox.YLength), round(boundbox.ZLength)}:
00197                     normal = face.normalAt(0,0)
00198                     normal = face.Placement.Rotation.inverted().multVec(normal)
00199                     #print "x: ", boundbox.XLength
00200                     #print "y: ", boundbox.YLength
00201                     #print "z: ", boundbox.ZLength
00202                     # Set length and width of user selected face of structural element
00203                     flag = True
00204                     # FIXME: Improve below logic.

```

```

00205         for i in range(len(normal)):
00206             if round(normal[i]) == 0:
00207                 if flag and i == 0:
00208                     x = center_of_mass[i]
00209                     facelength = boundbox.XLength
00210                     flag = False
00211                 elif flag and i == 1:
00212                     x = center_of_mass[i]
00213                     facelength = boundbox.YLength
00214                     flag = False
00215                 if i == 1:
00216                     y = center_of_mass[i]
00217                     facewidth = boundbox.YLength
00218                 elif i == 2:
00219                     y = center_of_mass[i]
00220                     facewidth = boundbox.ZLength
00221             #print [(facelength, facewidth), (x, y)]
00222     # Return parameter of the face when rebar is not created from the sketch.
00223     # For eg. non-planar rebars like stirrup etc.
00224     if not sketch:
00225         center_of_mass = face.CenterOfMass
00226         return [(facelength, facewidth), center_of_mass]
00227     #TODO: Add support when bounding box have depth. Here bounding box looks
00228     # like cuboid. If we given curved face.
00229     return [(facelength, facewidth), (x, y)]
00230
00231 # -----
00232 # Functions which is mainly used while creating stirrup.
00233 # -----
00234
00235 def extendedTangentPartLength(rounding, diameter, angle):
00236     """ extendedTangentPartLength(rounding, diameter, angle): Get a extended
00237     length of rounding on corners."""
00238     radius = rounding * diameter
00239     x1 = radius / math.tan(math.radians(angle))
00240     x2 = radius / math.cos(math.radians(90 - angle)) - radius
00241     return x1 + x2
00242
00243 def extendedTangentLength(rounding, diameter, angle):
00244     """ extendedTangentLength(rounding, diameter, angle): Get a extended
00245     length of rounding at the end of Stirrup for bent."""
00246     radius = rounding * diameter
00247     x1 = radius / math.sin(math.radians(angle))
00248     x2 = radius * math.tan(math.radians(90 - angle))
00249     return x1 + x2
00250
00251 # -----
00252 # Warning / Alert functions when user do something wrong.
00253 # -----
00254
00255 def check_selected_face():
00256     """ check_selected_face(): This function checks whether user have selected
00257     any face or not."""
00258     selected_objs = FreeCADGui.Selection.getSelectionEx()
00259     if not selected_objs:
00260         showWarning("Select any face of the structural element.")
00261         selected_obj = None
00262     else:
00263         selected_face_names = selected_objs[0].SubElementNames
00264         if not selected_face_names:
00265             selected_obj = None
00266             showWarning("Select any face of the structural element.")
00267         elif "Face" in selected_face_names[0]:
00268             if len(selected_face_names) > 1:
00269                 showWarning("You have selected more than one face of the structural element.")
00270                 selected_obj = None
00271             elif len(selected_face_names) == 1:
00272                 selected_obj = selected_objs[0]
00273         else:
00274             showWarning("Select any face of the selected the face.")
00275             selected_obj = None
00276     return selected_obj
00277
00278 def getSelectedFace(self):
00279     selected_objs = FreeCADGui.Selection.getSelectionEx()
00280     if selected_objs:
00281         if len(selected_objs[0].SubObjects) == 1:
00282             if "Face" in selected_objs[0].SubElementNames[0]:
00283                 self.SelectedObj = selected_objs[0].Object
00284                 self.FaceName = selected_objs[0].SubElementNames[0]
00285                 self.form.PickSelectedFaceLabel.setText("Selected face is " + self.FaceName)
00286             else:
00287                 showWarning("Select any face of the structural element.")
00288         else:
00289             showWarning("Select only one face of the structural element.")
00290     else:
00291         showWarning("Select any face of the structural element.")

```

```

00292
00293 def showWarning(message):
00294     """ showWarning(message): This function is used to produce warning
00295     message for the user."""
00296     msg = QtGui.QMessageBox()
00297     msg.setIcon(QtGui.QMessageBox.Warning)
00298     msg.setText(translate("RebarAddon", message))
00299     msg.setStandardButtons(QtGui.QMessageBox.Ok)
00300     msg.exec_()
00301
00302 # Qt translation handling
00303 def translate(context, text, disambig=None):
00304     return QtCore.QCoreApplication.translate(context, text, disambig)

```

8.15 RebarTools.py File Reference

Classes

- class [RebarTools.StraightRebarTool](#)
- class [RebarTools.UShapeRebarTool](#)
- class [RebarTools.LShapeRebarTool](#)
- class [RebarTools.StirrupTool](#)
- class [RebarTools.BentShapeRebarTool](#)
- class [RebarTools.HelicalRebarTool](#)

Namespaces

- [RebarTools](#)

Variables

- string [RebarTools.__title__](#) = "RebarCommands"
- string [RebarTools.__author__](#) = "Amritpal Singh"
- string [RebarTools.__url__](#) = "https://www.freecadweb.org"
- list [RebarTools.RebarCommands](#) = ["Arch_Rebar_Straight", "Arch_Rebar_UShape", "Arch_Rebar_LShape", "Arch_Rebar_Stirrup", "Arch_Rebar_BentShape", "Arch_Rebar_Helical"]

8.16 RebarTools.py

```

00001 # -*- coding: utf-8 -*-
00002 # ****
00003 # *
00004 # * Copyright (c) 2017 - Amritpal Singh <amrit3701@gmail.com>
00005 # *
00006 # * This program is free software; you can redistribute it and/or modify
00007 # * it under the terms of the GNU Lesser General Public License (LGPL)
00008 # * as published by the Free Software Foundation; either version 2 of
00009 # * the License, or (at your option) any later version.
00010 # * for detail see the LICENCE text file.
00011 # *
00012 # * This program is distributed in the hope that it will be useful,
00013 # * but WITHOUT ANY WARRANTY; without even the implied warranty of
00014 # * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
00015 # * GNU Library General Public License for more details.
00016 # *
00017 # * You should have received a copy of the GNU Library General Public
00018 # * License along with this program; if not, write to the Free Software
00019 # * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307
00020 # * USA
00021 # *
00022 # ****

```

```

00023
00024     __title__ = "RebarCommands"
00025     __author__ = "Amritpal Singh"
00026     __url__ = "https://www.freecadweb.org"
00027
00028     from PySide.QtCore import QT_TRANSLATE_NOOP
00029     import FreeCADGui
00030     import os
00031
00032     class StraightRebarTool:
00033
00034         def GetResources(self):
00035             return {'Pixmap' : os.path.split(os.path.abspath(__file__))[0]+'/'
00036                     'icons/dropdown_list/StraightRebar.svg',
00037                     'MenuText': QT_TRANSLATE_NOOP("RebarAddon", "Straight Rebar"),
00038                     'ToolTip' : QT_TRANSLATE_NOOP("RebarAddon", "Creates a Striaght bar reinforcement from the
00039 selected face of the Structural element.")}
00040
00041         def IsActive(self):
00042             if FreeCADGui.ActiveDocument:
00043                 return True
00044             else:
00045                 return False
00046
00047         def Activated(self):
00048             import StraightRebar
00049             # Call to CommandStraightRebar() function
00050             StraightRebar.CommandStraightRebar()
00051
00052     class UShapeRebarTool:
00053
00054         def GetResources(self):
00055             return {'Pixmap' : os.path.split(os.path.abspath(__file__))[0]+'/'
00056                     'icons/dropdown_list/UShapeRebar.svg',
00057                     'MenuText': QT_TRANSLATE_NOOP("RebarAddon", "U-Shape Rebar"),
00058                     'ToolTip' : QT_TRANSLATE_NOOP("RebarAddon", "Creates a U-Shape bar reinforcement from the
00059 selected face of the Structural element.")}
00060
00061         def IsActive(self):
00062             if FreeCADGui.ActiveDocument:
00063                 return True
00064             else:
00065                 return False
00066
00067         def Activated(self):
00068             import UShapeRebar
00069             # Call to CommandUShaepRebar() function
00070             UShapeRebar.CommandUShapeRebar()
00071
00072     class LShapeRebarTool:
00073
00074         def GetResources(self):
00075             return {'Pixmap' : os.path.split(os.path.abspath(__file__))[0]+'/'
00076                     'icons/dropdown_list/LShapeRebar.svg',
00077                     'MenuText': QT_TRANSLATE_NOOP("RebarAddon", "L-Shape Rebar"),
00078                     'ToolTip' : QT_TRANSLATE_NOOP("RebarAddon", "Creates a L-Shape bar reinforcement from the
00079 selected face of the Structural element.")}
00080
00081         def IsActive(self):
00082             if FreeCADGui.ActiveDocument:
00083                 return True
00084             else:
00085                 return False
00086
00087         def Activated(self):
00088             import LShapeRebar
00089             # Call to CommandUShaepRebar() function
00090             LShapeRebar.CommandLShapeRebar()
00091
00092     class StirrupTool:
00093
00094         def GetResources(self):
00095             return {'Pixmap' : os.path.split(os.path.abspath(__file__))[0]+'/'
00096                     'icons/dropdown_list/StirrupRebar.svg',
00097                     'MenuText': QT_TRANSLATE_NOOP("RebarAddon", "Stirrup"),
00098                     'ToolTip' : QT_TRANSLATE_NOOP("RebarAddon", "Creates a Stirrup bar reinforcement from the
00099 selected face of the Structural element.")}
00100
00101         def IsActive(self):
00102             if FreeCADGui.ActiveDocument:
00103                 return True
00104             else:
00105                 return False
00106
00107         def Activated(self):
00108             import Stirrup
00109             # Call to CommandStirrup() function

```

```

00102     Stirrup.CommandStirrup()
00103
00104 class BentShapeRebarTool:
00105
00106     def GetResources(self):
00107         return {'Pixmap' : os.path.split(os.path.abspath(__file__))[0]+'/'
00108                 'icons/dropdown_list/BentShapeRebar.svg',
00109                 'MenuText': QT_TRANSLATE_NOOP("RebarAddon", "Bent-Shape Rebar"),
00110                 'ToolTip' : QT_TRANSLATE_NOOP("RebarAddon", "Creates a BentShape bar reinforcement from the
00111                 selected face of the Structural element.")}
00112
00113     def IsActive(self):
00114         if FreeCADGui.ActiveDocument:
00115             return True
00116         else:
00117             return False
00118
00119     def Activated(self):
00120         import BentShapeRebar
00121         # Call to CommandBentShaepRebar() function
00122         BentShapeRebar.CommandBentShapeRebar()
00123
00124 class HelicalRebarTool:
00125
00126     def GetResources(self):
00127         return {'Pixmap' : os.path.split(os.path.abspath(__file__))[0]+'/'
00128                 'icons/dropdown_list/HelixShapeRebar.svg',
00129                 'MenuText': QT_TRANSLATE_NOOP("RebarAddon", "Helical Rebar"),
00130                 'ToolTip' : QT_TRANSLATE_NOOP("RebarAddon", "Creates a Helical bar reinforcement from the
00131                 selected face of the Structural element.")}
00132
00133     def IsActive(self):
00134         if FreeCADGui.ActiveDocument:
00135             return True
00136         else:
00137             return False
00138
00139     def Activated(self):
00140         import HelicalRebar
00141         # Call to CommandHelicalRebar() function
00142         HelicalRebar.CommandHelicalRebar()
00143
00144 FreeCADGui.addCommand('Arch_Rebar_Straight', StraightRebarTool())
00145 FreeCADGui.addCommand('Arch_Rebar_UShape', UShapeRebarTool())
00146 FreeCADGui.addCommand('Arch_Rebar_LShape', LShapeRebarTool())
00147 FreeCADGui.addCommand('Arch_Rebar_Stirrup', StirrupTool())
00148 FreeCADGui.addCommand('Arch_Rebar_BentShape', BentShapeRebarTool())
00149 FreeCADGui.addCommand('Arch_Rebar_Helical', HelicalRebarTool())
00150
00151 # List of all rebar commands
00152 RebarCommands = ["Arch_Rebar_Straight", "Arch_Rebar_UShape", "Arch_Rebar_LShape", "Arch_Rebar_Stirrup", "Arch_Rebar_BentShape", "Arch_Rebar_Helical"]

```

8.17 Stirrup.py File Reference

Classes

- class [Stirrup._StirrupTaskPanel](#)

Namespaces

- [Stirrup](#)

Functions

- def [Stirrup.getpointsOfStirrup](#) (FacePRM, l_cover, r_cover, t_cover, b_cover, bentAngle, bentFactor, diameter, rounding, facenormal)
- def [Stirrup.makeStirrup](#) (l_cover, r_cover, t_cover, b_cover, f_cover, bentAngle, bentFactor, diameter, rounding, amount_spacing_check, amount_spacing_value, structure=None, facename=None)
- def [Stirrup.editStirrup](#) (Rebar, l_cover, r_cover, t_cover, b_cover, f_cover, bentAngle, bentFactor, diameter, rounding, amount_spacing_check, amount_spacing_value, structure=None, facename=None)
- def [Stirrup.editDialog](#) (vobj)
- def [Stirrup.CommandStirrup](#) ()

Variables

- string `Stirrup.__title__` = "StirrupRebar"
- string `Stirrup.__author__` = "Amritpal Singh"
- string `Stirrup.__url__` = "https://www.freecadweb.org"

8.18 Stirrup.py

```

00001 # -*- coding: utf-8 -*-
00002 # ****
00003 # *
00004 # * Copyright (c) 2017 - Amritpal Singh <amrit3701@gmail.com>
00005 # *
00006 # * This program is free software; you can redistribute it and/or modify
00007 # * it under the terms of the GNU Lesser General Public License (LGPL)
00008 # * as published by the Free Software Foundation; either version 2 of
00009 # * the License, or (at your option) any later version.
00010 # * for detail see the LICENCE text file.
00011 # *
00012 # * This program is distributed in the hope that it will be useful,
00013 # * but WITHOUT ANY WARRANTY; without even the implied warranty of
00014 # * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
00015 # * GNU Library General Public License for more details.
00016 # *
00017 # * You should have received a copy of the GNU Library General Public
00018 # * License along with this program; if not, write to the Free Software
00019 # * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307
00020 # * USA
00021 # *
00022 # ****
00023
00024 __title__ = "StirrupRebar"
00025 __author__ = "Amritpal Singh"
00026 __url__ = "https://www.freecadweb.org"
00027
00028 from PySide import QtCore, QtGui
00029 from Rebarfunc import *
00030 from PySide.QtCore import QT_TRANSLATE_NOOP
00031 from RebarDistribution import runRebarDistribution, removeRebarDistribution
00032 from PopUpImage import showPopUpImageDialog
00033 import FreeCAD
00034 import FreeCADGui
00035 import ArchCommands
00036 import os
00037 import sys
00038 import math
00039
00040 def getpointsOfStirrup(FacePRM, l_cover, r_cover, t_cover, b_cover, bentAngle,
00041     bentFactor, diameter, rounding, facenormal):
00042     """ getpointsOfStirrup(FacePRM, LeftCover, RightCover, TopCover, BottomCover, BentAngle, BentFactor,
00043     Diameter, Rounding, FaceNormal):
00044         Return the coordinates points of the Stirrup in the form of array."""
00045     angle = 180 - bentAngle
00046     tangent_part_length = extendedTangentPartLength(rounding, diameter, angle)
00047     tangent_length = extendedTangentLength(rounding, diameter, angle)
00048     if round(facenormal[0]) in {1,-1}:
00049         x1 = FacePRM[1][0]
00050         y1 = FacePRM[1][1] - FacePRM[0][0] / 2 + l_cover
00051         z1 = FacePRM[1][2] + FacePRM[0][1] / 2 - t_cover + tangent_part_length
00052         y2 = FacePRM[1][1] - FacePRM[0][0] / 2 + l_cover
00053         z2 = FacePRM[1][2] - FacePRM[0][1] / 2 + b_cover
00054         y3 = FacePRM[1][1] + FacePRM[0][0] / 2 - r_cover
00055         z3 = FacePRM[1][2] - FacePRM[0][1] / 2 + b_cover
00056         y4 = FacePRM[1][1] + FacePRM[0][0] / 2 - r_cover
00057         z4 = FacePRM[1][2] + FacePRM[0][1] / 2 - t_cover
00058         y5 = FacePRM[1][1] - FacePRM[0][0] / 2 + l_cover - tangent_part_length
00059         z5 = FacePRM[1][2] + FacePRM[0][1] / 2 - t_cover
00060         side_length = abs(y5 - y4) - tangent_part_length
00061         normal_dis = (diameter * (side_length + tangent_part_length)) / side_length
00062         x2 = x1 - normal_dis / 4
00063         x3 = x2 - normal_dis / 4
00064         x4 = x3 - normal_dis / 4
00065         x5 = x4 - normal_dis / 4
00066         x0 = x1 + normal_dis / 4
00067         y0 = y1 + (tangent_length + bentFactor * diameter) * math.sin(math.radians(angle))
00068         z0 = z1 - (tangent_length + bentFactor * diameter) * math.cos(math.radians(angle))
00069         x6 = x5 - normal_dis / 4
00070         y6 = y5 + (tangent_length + bentFactor * diameter) * math.sin(math.radians(90 - angle))
00071         z6 = z5 - (tangent_length + bentFactor * diameter) * math.cos(math.radians(90 - angle))

```

```

00070     elif round(facenormal[1]) in {1,-1}:
00071         x1 = FacePRM[1][0] - FacePRM[0][0] / 2 + l_cover
00072         y1 = FacePRM[1][1]
00073         z1 = FacePRM[1][2] + FacePRM[0][1] / 2 - t_cover + tangent_part_length
00074         x2 = FacePRM[1][0] - FacePRM[0][0] / 2 + l_cover
00075         z2 = FacePRM[1][2] - FacePRM[0][1] / 2 + b_cover
00076         x3 = FacePRM[1][0] + FacePRM[0][0] / 2 - r_cover
00077         z3 = FacePRM[1][2] - FacePRM[0][1] / 2 + b_cover
00078         x4 = FacePRM[1][0] + FacePRM[0][0] / 2 - r_cover
00079         z4 = FacePRM[1][2] + FacePRM[0][1] / 2 - t_cover
00080         x5 = FacePRM[1][0] - FacePRM[0][0] / 2 + l_cover - tangent_part_length
00081         z5 = FacePRM[1][2] + FacePRM[0][1] / 2 - t_cover
00082         side_length = abs(x5 - x4) - tangent_part_length
00083         normal_dis = (diameter * (side_length + tangent_part_length)) / side_length
00084         y2 = y1 - normal_dis / 4
00085         y3 = y2 - normal_dis / 4
00086         y4 = y3 - normal_dis / 4
00087         y5 = y4 - normal_dis / 4
00088         y0 = y1 + normal_dis / 4
00089         x0 = x1 + (tangent_length + bentFactor * diameter) * math.sin(math.radians(angle))
00090         z0 = z1 - (tangent_length + bentFactor * diameter) * math.cos(math.radians(angle))
00091         x6 = x5 + (tangent_length + bentFactor * diameter) * math.sin(math.radians(90 - angle))
00092         y6 = y5 - normal_dis / 4
00093         z6 = z5 - (tangent_length + bentFactor * diameter) * math.cos(math.radians(90 - angle))
00094     elif round(facenormal[2]) in {1,-1}:
00095         x1 = FacePRM[1][0] - FacePRM[0][0] / 2 + l_cover
00096         y1 = FacePRM[1][1] + FacePRM[0][1] / 2 - t_cover + tangent_part_length
00097         z1 = FacePRM[1][2]
00098         x2 = FacePRM[1][0] - FacePRM[0][0] / 2 + l_cover
00099         y2 = FacePRM[1][1] - FacePRM[0][1] / 2 + b_cover
00100         x3 = FacePRM[1][0] + FacePRM[0][0] / 2 - r_cover
00101         y3 = FacePRM[1][1] - FacePRM[0][1] / 2 + b_cover
00102         x4 = FacePRM[1][0] + FacePRM[0][0] / 2 - r_cover
00103         y4 = FacePRM[1][1] + FacePRM[0][1] / 2 - t_cover
00104         x5 = FacePRM[1][0] - FacePRM[0][0] / 2 + l_cover - tangent_part_length
00105         y5 = FacePRM[1][1] + FacePRM[0][1] / 2 - t_cover
00106         side_length = abs(x5 - x4) - tangent_part_length
00107         normal_dis = (diameter * (side_length + tangent_part_length)) / side_length
00108         z2 = z1 - normal_dis / 4
00109         z3 = z2 - normal_dis / 4
00110         z4 = z3 - normal_dis / 4
00111         z5 = z4 - normal_dis / 4
00112         z0 = z1 + normal_dis / 4
00113         x0 = x1 + (tangent_length + bentFactor * diameter) * math.sin(math.radians(angle))
00114         y0 = y1 - (tangent_length + bentFactor * diameter) * math.cos(math.radians(angle))
00115         x6 = x5 + (tangent_length + bentFactor * diameter) * math.sin(math.radians(90 - angle))
00116         y6 = y5 - (tangent_length + bentFactor * diameter) * math.cos(math.radians(90 - angle))
00117         z6 = z5 - normal_dis / 4
00118     return [FreeCAD.Vector(x0, y0, z0), FreeCAD.Vector(x1, y1, z1), \
00119             FreeCAD.Vector(x2, y2, z2), FreeCAD.Vector(x3, y3, z3), \
00120             FreeCAD.Vector(x4, y4, z4), FreeCAD.Vector(x5, y5, z5), \
00121             FreeCAD.Vector(x6, y6, z6)]
00122
00123 class _StirrupTaskPanel:
00124     def __init__(self, Rebar = None):
00125         self.CustomSpacing = None
00126         if not Rebar:
00127             selected_obj = FreeCADGui.Selection.getSelectionEx()[0]
00128             self.SelectedObj = selected_obj.Object
00129             self.FaceName = selected_obj.SubElementNames[0]
00130         else:
00131             self.FaceName = Rebar.Base.Support[0][1][0]
00132             self.SelectedObj = Rebar.Base.Support[0][0]
00133         self.form = FreeCADGui.PySideUic.loadUi(os.path.splitext(__file__)[0] + ".ui")
00134         self.form.setWindowTitle(QtGui.QApplication.translate("RebarAddon", "Stirrup Rebar", None))
00135         self.form.bentAngle.addItem("135", "90")
00136         self.form.amount_radio.clicked.connect(self.amount_radio_clicked)
00137         self.form.spacing_radio.clicked.connect(self.spacing_radio_clicked)
00138         self.form.image.setPixmap(QtGui.QPixmap(os.path.split(os.path.abspath(__file__))[0]+\
00139             "/icons/Stirrup.svg"))
00140         self.form.customSpacing.clicked.connect(lambda: runRebarDistribution(self))
00141         self.form.removeCustomSpacing.clicked.connect(lambda:\
00142             removeRebarDistribution(self))
00143         self.form.PickSelectedFace.clicked.connect(lambda: getSelectedFace(self))
00144         self.form.toolButton.setIcon(self.form.toolButton.style().standardIcon(
00145             QtGui.QStyle.SP_DialogHelpButton))
00146         self.form.toolButton.clicked.connect(lambda: showPopUpImageDialog(os.path.split(
00147             (os.path.abspath(__file__))[0] + "/icons/StirrupDetailed.svg")))
00148         self.Rebar = Rebar
00149         def getStandardButtons(self):
00150             return int(QtGui.QDialogButtonBox.Ok) | int(QtGui.QDialogButtonBox.Apply) | int(
00151             QtGui.QDialogButtonBox.Cancel)
00152         def clicked(self, button):
00153             if button == int(QtGui.QDialogButtonBox.Apply):
00154                 self.accept(button)

```

```

00152
00153     def accept(self, signal = None):
00154         l_cover = self.form.l_sideCover.text()
00155         l_cover = FreeCAD.Units.Quantity(l_cover).Value
00156         r_cover = self.form.r_sideCover.text()
00157         r_cover = FreeCAD.Units.Quantity(r_cover).Value
00158         t_cover = self.form.t_sideCover.text()
00159         t_cover = FreeCAD.Units.Quantity(t_cover).Value
00160         b_cover = self.form.b_sideCover.text()
00161         b_cover = FreeCAD.Units.Quantity(b_cover).Value
00162         f_cover = self.form.frontCover.text()
00163         f_cover = FreeCAD.Units.Quantity(f_cover).Value
00164         diameter = self.form.diameter.text()
00165         diameter = FreeCAD.Units.Quantity(diameter).Value
00166         bentAngle = int(self.form.bentAngle.currentText())
00167         bentFactor = self.form.bentFactor.value()
00168         rounding = self.form.rounding.value()
00169         amount_check = self.form.amount_radio.isChecked()
00170         spacing_check = self.form.spacing_radio.isChecked()
00171         if not self.Rebar:
00172             if amount_check:
00173                 amount = self.form.amount.value()
00174                 rebar = makeStirrup(l_cover, r_cover, t_cover, b_cover, f_cover, bentAngle,
00175                                     bentFactor, diameter,\n                                         rounding, True, amount, self.SelectedObj, self.
00176                                     FaceName)
00177             elif spacing_check:
00178                 spacing = self.form.spacing.text()
00179                 spacing = FreeCAD.Units.Quantity(spacing).Value
00180                 rebar = makeStirrup(l_cover, r_cover, t_cover, b_cover, f_cover, bentAngle,
00181                                     bentFactor, diameter,\n                                         rounding, False, spacing, self.SelectedObj, self.
00182                                     FaceName)
00183         else:
00184             if amount_check:
00185                 amount = self.form.amount.value()
00186                 rebar = editStirrup(self.Rebar, l_cover, r_cover, t_cover, b_cover, f_cover
00187                                     , bentAngle, bentFactor,\n                                         diameter, rounding, True, amount, self.SelectedObj, self.
00188                                     FaceName)
00189             elif spacing_check:
00190                 spacing = self.form.spacing.text()
00191                 spacing = FreeCAD.Units.Quantity(spacing).Value
00192                 rebar = editStirrup(self.Rebar, l_cover, r_cover, t_cover, b_cover, f_cover
00193                                     , bentAngle, bentFactor,\n                                         diameter, rounding, False, spacing, self.SelectedObj, self.
00194                                     FaceName)
00195             if self.CustomSpacing:
00196                 rebar.CustomSpacing = self.CustomSpacing
00197                 FreeCAD.ActiveDocument.recompute()
00198             if signal == int(QtGui.QDialogButtonBox.Apply):
00199                 pass
00200             else:
00201                 FreeCADGui.Control.closeDialog(self)
00202
00203     def amount_radio_clicked(self):
00204         self.form.spacing.setEnabled(False)
00205         self.form.amount.setEnabled(True)
00206
00207     def spacing_radio_clicked(self):
00208         self.form.amount.setEnabled(False)
00209         self.form.spacing.setEnabled(True)
00210
00211     def makeStirrup(l_cover, r_cover, t_cover, b_cover, f_cover, bentAngle, bentFactor, diameter,
00212                     rounding,\n                         amount_spacing_check, amount_spacing_value, structure = None, facename = None):
00213         """ makeStirrup(LeftCover, RightCover, TopCover, BottomCover, FrontCover, BentAngle,
00214         BentFactor, Diameter, Rounding, AmountSpacingCheck, AmountSpacingValue, Structure, Facename):
00215         Adds the Stirrup reinforcement bar to the selected structural object."""
00216         if not structure and not facename:
00217             selected_obj = FreeCADGui.Selection.getSelectionEx()[0]
00218             structure = selected_obj.Object
00219             facename = selected_obj.SubElementNames[0]
00220             face = structure.Shape.Faces[getFaceNumber(facename) - 1]
00221             #StructurePRM = getTrueParametersOfStructure(structure)
00222             FacePRM = getParametersOfFace(structure, facename, False)
00223             FaceNormal = face.normalAt(0,0)
00224             #FaceNormal = face.Placement.Rotation.inverted().multVec(FaceNormal)
00225             if not FacePRM:
00226                 FreeCAD.Console.PrintError("Cannot identified shape or from which base object sturctural element is
00227 derived\n")
00228             return
00229         # Calculate the coordinate values of Stirrup
00230         points = getPointsOfStirrup(FacePRM, l_cover, r_cover, t_cover, b_cover, bentAngle,
00231                                     bentFactor, diameter, rounding, FaceNormal)

```

```

00228 import Draft
00229 line = Draft.makeWire(points, closed = False, face = True, support = None)
00230 import Arch
00231 line.Support = [(structure, facename)]
00232 if amount_spacing_check:
00233     rebar = Arch.makeRebar(structure, line, diameter, amount_spacing_value, f_cover)
00234 else:
00235     size = (ArchCommands.projectToVector(structure.Shape.copy(), face.normalAt(0, 0))).Length
00236     rebar = Arch.makeRebar(structure, line, diameter,\n        int((size - diameter) / amount_spacing_value), f_cover)
00237     rebar.Direction = FaceNormal.negative()
00238     rebar.Rounding = rounding
00239 # Adds properties to the rebar object
00240 rebar.ViewObject.addProperty("App::PropertyString", "RebarShape", "RebarDialog",\
00241     QT_TRANSLATE_NOOP("App::Property", "Shape of rebar")).RebarShape = "Stirrup"
00242 rebar.ViewObject.setEditorMode("RebarShape", 2)
00243 rebar.addProperty("App::PropertyDistance", "LeftCover", "RebarDialog",\
00244     QT_TRANSLATE_NOOP("App::Property", "Left Side cover of rebar")).LeftCover = l_cover
00245 rebar.setEditorMode("LeftCover", 2)
00246 rebar.addProperty("App::PropertyDistance", "RightCover", "RebarDialog",\
00247     QT_TRANSLATE_NOOP("App::Property", "Right Side cover of rebar")).RightCover = r_cover
00248 rebar.setEditorMode("RightCover", 2)
00249 rebar.addProperty("App::PropertyDistance", "TopCover", "RebarDialog",\
00250     QT_TRANSLATE_NOOP("App::Property", "Top Side cover of rebar")).TopCover = t_cover
00251 rebar.setEditorMode("TopCover", 2)
00252 rebar.addProperty("App::PropertyDistance", "BottomCover", "RebarDialog",\
00253     QT_TRANSLATE_NOOP("App::Property", "Bottom Side cover of rebar")).BottomCover = b_cover
00254 rebar.setEditorMode("BottomCover", 2)
00255 rebar.addProperty("App::PropertyDistance", "FrontCover", "RebarDialog",\
00256     QT_TRANSLATE_NOOP("App::Property", "Top cover of rebar")).FrontCover = f_cover
00257 rebar.setEditorMode("FrontCover", 2)
00258 rebar.addProperty("App::PropertyInteger", "BentAngle", "RebarDialog",\
00259     QT_TRANSLATE_NOOP("App::Property", "Bent angle between at the end of rebar")).BentAngle = bentAngle
00260 rebar.setEditorMode("BentAngle", 2)
00261 rebar.addProperty("App::PropertyInteger", "BentFactor", "RebarDialog",\
00262     QT_TRANSLATE_NOOP("App::Property", "Bent Length is the equal to BentFactor * Diameter")).BentFactor
00263 = bentFactor
00264 rebar.setEditorMode("BentFactor", 2)
00265 rebar.addProperty("App::PropertyBool", "AmountCheck", "RebarDialog",\
00266     QT_TRANSLATE_NOOP("App::Property", "Amount radio button is checked")).AmountCheck
00267 rebar.setEditorMode("AmountCheck", 2)
00268 rebar.addProperty("App::PropertyDistance", "TrueSpacing", "RebarDialog",\
00269     QT_TRANSLATE_NOOP("App::Property", "Spacing between of rebars")).TrueSpacing = amount_spacing_value
00270 rebar.setEditorMode("TrueSpacing", 2)
00271 if amount_spacing_check:
00272     rebar.AmountCheck = True
00273 else:
00274     rebar.AmountCheck = False
00275     rebar.TrueSpacing = amount_spacing_value
00276 rebar.Label = "Stirrup"
00277 FreeCAD.ActiveDocument.recompute()
00278 return rebar
00279
00280 def editStirrup(Rebar, l_cover, r_cover, t_cover, b_cover, f_cover, bentAngle, bentFactor,
00281     diameter, rounding,\n        amount_spacing_check, amount_spacing_value, structure = None, facename = None):
00282     sketch = Rebar.Base
00283     if structure and facename:
00284         sketch.Support = [(structure, facename)]
00285     # Check if sketch support is empty.
00286     if not sketch.Support:
00287         showWarning("You have checked remove external geometry of base sketchs when needed.\nTo\nunchecked Edit->Preferences->Arch.")
00288         return
00289     # Assigned values
00290     facename = sketch.Support[0][1][0]
00291     structure = sketch.Support[0][0]
00292     face = structure.Shape.Faces[getFaceNumber(facename) - 1]
00293     #StructurePRM = getTrueParametersOfStructure(structure)
00294     # Get parameters of the face where sketch of rebar is drawn
00295     FacePRM = getParametersOfFace(structure, facename, False)
00296     FaceNormal = face.normalAt(0, 0)
00297     #FaceNormal = face.Placement.Rotation.inverted().multVec(FaceNormal)
00298     # Calculate the coordinates value of Stirrup rebar
00299     points = getpointsOfStirrup(FacePRM, l_cover, r_cover, t_cover, b_cover, bentAngle,
00300         bentFactor, diameter, rounding, FaceNormal)
00301     Rebar.Base.Points = points
00302     FreeCAD.ActiveDocument.recompute()
00303     Rebar.Direction = FaceNormal.negative()
00304     Rebar.OffsetStart = f_cover
00305     Rebar.OffsetEnd = f_cover
00306     Rebar.BentAngle = bentAngle
00307     Rebar.BentFactor = bentFactor
00308     Rebar.Rounding = rounding
00309     Rebar.Diameter = diameter
00310     if amount_spacing_check:
00311         Rebar.Amount = amount_spacing_value

```

```

00311     FreeCAD.ActiveDocument.recompute()
00312     Rebar.AmountCheck = True
00313     else:
00314         size = (ArchCommands.projectToVector(structure.Shape.copy(), face.normalAt(0, 0))).Length
00315         Rebar.Amount = int((size - diameter) / amount_spacing_value)
00316         FreeCAD.ActiveDocument.recompute()
00317         Rebar.AmountCheck = False
00318     Rebar.FrontCover = f_cover
00319     Rebar.LeftCover = l_cover
00320     Rebar.RightCover = r_cover
00321     Rebar.TopCover = t_cover
00322     Rebar.BottomCover = b_cover
00323     Rebar.TrueSpacing = amount_spacing_value
00324     FreeCAD.ActiveDocument.recompute()
00325     return Rebar
00326
00327 def editDialog(vobj):
00328     FreeCADGui.Control.closeDialog()
00329     obj = _StirrupTaskPanel(vobj.Object)
00330     obj.form.frontCover.setText(str(vobj.Object.FrontCover))
00331     obj.form.l_sideCover.setText(str(vobj.Object.LeftCover))
00332     obj.form.r_sideCover.setText(str(vobj.Object.RightCover))
00333     obj.form.t_sideCover.setText(str(vobj.Object.TopCover))
00334     obj.form.b_sideCover.setText(str(vobj.Object.BottomCover))
00335     obj.form.diameter.setText(str(vobj.Object.Diameter))
00336     obj.form.bentAngle.setCurrentIndex(obj.form.bentAngle.findText(str(vobj.Object.BentAngle)))
00337     obj.form.bentFactor.setValue(vobj.Object.BentFactor)
00338     obj.form.rounding.setValue(vobj.Object.Rounding)
00339     if vobj.Object.AmountCheck:
00340         obj.form.amount.setValue(vobj.Object.Amount)
00341     else:
00342         obj.form.amount_radio.setChecked(False)
00343         obj.form.spacing_radio.setChecked(True)
00344         obj.form.amount.setDisabled(True)
00345         obj.form.spacing.setEnabled(True)
00346         obj.form.spacing.setText(str(vobj.Object.TrueSpacing))
00347     #obj.form.PickSelectedFace.setVisible(False)
00348     FreeCADGui.Control.showDialog(obj)
00349
00350 def CommandStirrup():
00351     selected_obj = check_selected_face()
00352     if selected_obj:
00353         FreeCADGui.Control.showDialog(_StirrupTaskPanel())

```

8.19 StraightRebar.py File Reference

Classes

- class [StraightRebar._StraightRebarTaskPanel](#)

Namespaces

- [StraightRebar](#)

Functions

- def [StraightRebar.getpointsOfStraightRebar](#) (FacePRM, rt_cover, lb_cover, coverAlong, orientation)
- def [StraightRebar.makeStraightRebar](#) (f_cover, coverAlong, rt_cover, lb_cover, diameter, amount_spacing←_check, amount_spacing_value, orientation="Horizontal", structure=None, facename=None)
- def [StraightRebar.editStraightRebar](#) (Rebar, f_cover, coverAlong, rt_cover, lb_cover, diameter, amount←_spacing_check, amount_spacing_value, orientation, structure=None, facename=None)
- def [StraightRebar.editDialog](#) (vobj)
- def [StraightRebar.CommandStraightRebar](#) ()

Variables

- string `StraightRebar.__title__` = "StraightRebar"
- string `StraightRebar.__author__` = "Amritpal Singh"
- string `StraightRebar.__url__` = "https://www.freecadweb.org"

8.20 StraightRebar.py

```

00001 # -*- coding: utf-8 -*-
00002 # ****
00003 # *
00004 # * Copyright (c) 2017 - Amritpal Singh <amrit3701@gmail.com>
00005 # *
00006 # * This program is free software; you can redistribute it and/or modify
00007 # * it under the terms of the GNU Lesser General Public License (LGPL)
00008 # * as published by the Free Software Foundation; either version 2 of
00009 # * the License, or (at your option) any later version.
00010 # * for detail see the LICENCE text file.
00011 # *
00012 # * This program is distributed in the hope that it will be useful,
00013 # * but WITHOUT ANY WARRANTY; without even the implied warranty of
00014 # * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
00015 # * GNU Library General Public License for more details.
00016 # *
00017 # * You should have received a copy of the GNU Library General Public
00018 # * License along with this program; if not, write to the Free Software
00019 # * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307
00020 # * USA
00021 # *
00022 # ****
00023
00024 __title__ = "StraightRebar"
00025 __author__ = "Amritpal Singh"
00026 __url__ = "https://www.freecadweb.org"
00027
00028 from PySide import QtCore, QtGui
00029 from Rebarfunc import *
00030 from PySide.QtCore import QT_TRANSLATE_NOOP
00031 from RebarDistribution import runRebarDistribution, removeRebarDistribution
00032 from PopUpImage import showPopUpImageDialog
00033 import FreeCAD
00034 import FreeCADGui
00035 import ArchCommands
00036 import os
00037 import sys
00038 import math
00039
00040 def getpointsOfStraightRebar(FacePRM, rt_cover, lb_cover, coverAlong, orientation):
00041     """ getpointsOfStraightRebar(FacePRM, RightTopcover, LeftBottomcover, CoverAlong, Orientation):
00042     Return points of the Straight rebar in the form of array for sketch.
00043
00044     Case I: When Orientation is 'Horizontal':
00045         We have two option in CoverAlong i.e. 'Bottom Side' or 'Top Side'
00046     Case II: When Orientation is 'Vertical':
00047         We have two option in CoverAlong i.e. 'Left Side' or 'Right Side'
00048     """
00049     if orientation == "Horizontal":
00050         if coverAlong[0] == "Bottom Side":
00051             x1 = FacePRM[1][0] - FacePRM[0][0] / 2 + lb_cover
00052             y1 = FacePRM[1][1] - FacePRM[0][1] / 2 + coverAlong[1]
00053             x2 = FacePRM[1][0] - FacePRM[0][0] / 2 + FacePRM[0][0] - rt_cover
00054             y2 = FacePRM[1][1] - FacePRM[0][1] / 2 + coverAlong[1]
00055         elif coverAlong[0] == "Top Side":
00056             cover = FacePRM[0][1] - coverAlong[1]
00057             x1 = FacePRM[1][0] - FacePRM[0][0] / 2 + lb_cover
00058             y1 = FacePRM[1][1] - FacePRM[0][1] / 2 + cover
00059             x2 = FacePRM[1][0] - FacePRM[0][0] / 2 + FacePRM[0][0] - rt_cover
00060             y2 = FacePRM[1][1] - FacePRM[0][1] / 2 + cover
00061     elif orientation == "Vertical":
00062         if coverAlong[0] == "Left Side":
00063             x1 = FacePRM[1][0] - FacePRM[0][0] / 2 + coverAlong[1]
00064             y1 = FacePRM[1][1] - FacePRM[0][1] / 2 + lb_cover
00065             x2 = FacePRM[1][0] - FacePRM[0][0] / 2 + coverAlong[1]
00066             y2 = FacePRM[1][1] - FacePRM[0][1] / 2 + FacePRM[0][1] - rt_cover
00067         elif coverAlong[0] == "Right Side":
00068             cover = FacePRM[0][0] - coverAlong[1]
00069             x1 = FacePRM[1][0] - FacePRM[0][0] / 2 + cover
00070             y1 = FacePRM[1][1] - FacePRM[0][1] / 2 + lb_cover
00071             x2 = FacePRM[1][0] - FacePRM[0][0] / 2 + cover

```

```

00072         y2 = FacePRM[1][1] - FacePRM[0][1] / 2 + FacePRM[0][1] - rt_cover
00073     return [FreeCAD.Vector(x1, y1, 0), FreeCAD.Vector(x2, y2, 0)]
00074
00075 class _StraightRebarTaskPanel:
00076     def __init__(self, Rebar = None):
00077         self.CustomSpacing = None
00078         if not Rebar:
00079             selected_obj = FreeCADGui.Selection.getSelectionEx() [0]
00080             self.SelectedObj = selected_obj.Object
00081             self.FaceName = selected_obj.SubElementNames[0]
00082         else:
00083             self.FaceName = Rebar.Base.Support[0][1][0]
00084             self.SelectedObj = Rebar.Base.Support[0][0]
00085         self.form = FreeCADGui.PySideUic.loadUi(os.path.splitext(__file__)[0] + ".ui")
00086         self.form.setWindowTitle(QtGui.QApplication.translate("RebarAddon", "Straight Rebar", None))
00087         self.form.orientation.addItems(["Horizontal", "Vertical"])
00088         self.form.coverAlong.addItems(["Bottom Side", "Top Side"])
00089         self.form.amount_radio.clicked.connect(self.amount_radio_clicked)
00090         self.form.spacing_radio.clicked.connect(self.spacing_radio_clicked)
00091         self.form.customSpacing.clicked.connect(lambda: runRebarDistribution(self))
00092         self.form.removeCustomSpacing.clicked.connect(lambda:
00093             removeRebarDistribution(self))
00094         self.form.PickSelectedFace.setCheckable(True)
00095         self.form.PickSelectedFace.toggle()
00096         self.form.PickSelectedFace.clicked.connect(lambda: getSelectedFace(self))
00097         self.form.image.setPixmap(QtGui.QPixmap(os.path.split(os.path.abspath(__file__))[0] + "
00098             /icons/StraightRebarH.svg"))
00099         self.form.orientation.currentIndexChanged.connect(self.changeOrientation)
00100         self.form.coverAlong.currentIndexChanged.connect(self.changeCoverAlong)
00101         self.form.toolButton.setIcon(self.form.toolButton.style().standardIcon(
00102             QtGui.QStyle.SP_DialogHelpButton))
00103         self.form.toolButton.clicked.connect(lambda: showPopUpImageDialog(os.path.split(
00104             (os.path.abspath(__file__))[0] + "/icons/StraightRebarDetailed.svg")))
00105         self.Rebar = Rebar
00106
00107     def changeOrientation(self):
00108         orientation = self.form.orientation.currentText()
00109         if orientation == "Horizontal":
00110             self.form.image.setPixmap(QtGui.QPixmap(os.path.split(os.path.abspath(__file__))[0] + "
00111                 /icons/StraightRebarH.svg"))
00112             self.form.r_sideCoverLabel.setText("Right Side Cover")
00113             self.form.l_sideCoverLabel.setText("Left Side Cover")
00114             self.form.coverAlong.clear()
00115             self.form.coverAlong.addItem(["Bottom Side", "Top Side"])
00116         else:
00117             self.form.image.setPixmap(QtGui.QPixmap(os.path.split(os.path.abspath(__file__))[0] + "
00118                 /icons/StraightRebarV.svg"))
00119             self.form.r_sideCoverLabel.setText("Top Side Cover")
00120             self.form.l_sideCoverLabel.setText("Bottom Side Cover")
00121             self.form.coverAlong.clear()
00122             self.form.coverAlong.addItem(["Right Side", "Left Side"])
00123
00124     def changeCoverAlong(self):
00125         coverAlong = self.form.coverAlong.currentText()
00126         if coverAlong == "Bottom Side":
00127             self.form.bottomCoverLabel.setText("Bottom Cover")
00128         elif coverAlong == "Top Side":
00129             self.form.bottomCoverLabel.setText("Top Cover")
00130         elif coverAlong == "Right Side":
00131             self.form.bottomCoverLabel.setText("Right Cover")
00132         else:
00133             self.form.bottomCoverLabel.setText("Left Cover")
00134
00135     def getStandardButtons(self):
00136         return int(QtGui.QDialogButtonBox.Ok) | int(QtGui.QDialogButtonBox.Apply) | int(
00137             QtGui.QDialogButtonBox.Cancel)
00138
00139     def clicked(self, button):
00140         if button == int(QtGui.QDialogButtonBox.Apply):
00141             self.accept(button)
00142
00143     def accept(self, signal = None):
00144         f_cover = self.form.frontCover.text()
00145         f_cover = FreeCAD.Units.Quantity(f_cover).Value
00146         cover = self.form.bottomCover.text()
00147         cover = FreeCAD.Units.Quantity(cover).Value
00148         lb_cover = self.form.l_sideCover.text()
00149         lb_cover = FreeCAD.Units.Quantity(lb_cover).Value
00150         rt_cover = self.form.r_sideCover.text()
00151         rt_cover = FreeCAD.Units.Quantity(rt_cover).Value
00152         orientation = self.form.orientation.currentText()
00153         coverAlong = self.form.coverAlong.currentText()
00154         diameter = self.form.diameter.text()
00155         diameter = FreeCAD.Units.Quantity(diameter).Value
00156         amount_check = self.form.amount_radio.isChecked()
00157         spacing_check = self.form.spacing_radio.isChecked()
00158         if not self.Rebar:
00159

```

```

00152         if amount_check:
00153             amount = self.form.amount.value()
00154             rebar = makeStraightRebar(f_cover, (coverAlong, cover), rt_cover, lb_cover
00155             , diameter, True, amount, orientation, self.SelectedObj, self.FaceName)
00156             elif spacing_check:
00157                 spacing = self.form.spacing.text()
00158                 spacing = FreeCAD.Units.Quantity(spacing).Value
00159                 rebar = makeStraightRebar(f_cover, (coverAlong, cover), rt_cover, lb_cover
00160                 , diameter, False, spacing, orientation, self.SelectedObj, self.
00161                 FaceName)
00162             else:
00163                 if amount_check:
00164                     amount = self.form.amount.value()
00165                     rebar = editStraightRebar(self.Rebar, f_cover, (coverAlong, cover),
00166                     rt_cover, lb_cover, diameter, True, amount, orientation, self.SelectedObj, self.
00167                     FaceName)
00168                 elif spacing_check:
00169                     spacing = self.form.spacing.text()
00170                     spacing = FreeCAD.Units.Quantity(spacing).Value
00171                     rebar = editStraightRebar(self.Rebar, f_cover, (coverAlong, cover),
00172                     rt_cover, lb_cover, diameter, False, spacing, orientation, self.SelectedObj, self.
00173                     FaceName)
00174             if self.CustomSpacing:
00175                 rebar.CustomSpacing = self.CustomSpacing
00176                 FreeCAD.ActiveDocument.recompute()
00177             self.Rebar = rebar
00178             if signal == int(QtGui.QDialogButtonBox.Apply):
00179                 pass
00180             else:
00181                 FreeCADGui.Control.closeDialog(self)
00182
00183
00184
00185 def makeStraightRebar(f_cover, coverAlong, rt_cover, lb_cover, diameter,
00186     amount_spacing_check, amount_spacing_value, orientation = "Horizontal", structure = None, facename = None):
00187     """ Adds the straight reinforcement bar to the selected structural object.
00188
00189     Case I: When orientation of straight rebar is 'Horizontal':
00190         makeStraightRebar(FrontCover, CoverAlong, RightCover, LeftCover, Diameter, AmountSpacingCheck,
00191             AmountSpacingValue, Orientation = "Horizontal",
00192             Structure, Facename)
00193             Note: Type of CoverAlong argument is a tuple. Syntax: (<Along>, <Value>). Here we have horizontal
00194             orientation so we can pass Top Side
00195             and Bottom Side to <Along> arguments.
00196             For eg. ("Top Side", 20) and ("Bottom Side", 20)
00197
00198     Case II: When orientation of straight rebar is 'Vertical':
00199         makeStraightRebar(FrontCover, CoverAlong, TopCover, BottomCover, Diameter, AmountSpacingCheck,
00200             AmountSpacingValue, Orientation = "Vertical",
00201             Structure, Facename)
00202             Note: Type of CoverAlong argument is a tuple. Syntax: (<Along>, <Value>). Here we have vertical
00203             orientation so we can pass Left Side
00204             and Right Side to <Along> arguments.
00205             For eg. ("Left Side", 20) and ("Right Side", 20)
00206
00207     """
00208     if not structure and not facename:
00209         selected_obj = FreeCADGui.Selection.getSelectionEx() [0]
00210         structure = selected_obj.Object
00211         facename = selected_obj.SubElementNames[0]
00212         face = structure.Shape.Faces[getFaceNumber(facename) - 1]
00213         #StructurePRM = getTrueParametersOfStructure(structure)
00214         FacePRM = getParametersOfFace(structure, facename)
00215         if not FacePRM:
00216             FreeCAD.Console.PrintError("Cannot identified shape or from which base object sturctural element is
00217             derived\n")
00218             return
00219             # Get points of Striaght rebar
00220             points = getpointsOfStraightRebar(FacePRM, rt_cover, lb_cover, coverAlong,
00221             orientation)
00222             import Part
00223             import Arch
00224             sketch = FreeCAD.activeDocument().addObject('Sketcher::SketchObject', 'Sketch')
00225             sketch.MapMode = "FlatFace"
00226             sketch.Support = [(structure, facename)]
00227             FreeCAD.ActiveDocument.recompute()
00228             sketch.addGeometry(Part.LineSegment(points[0], points[1]), False)
00229             if amount_spacing_check:
00230                 rebar = Arch.makeRebar(structure, sketch, diameter, amount_spacing_value, f_cover)
00231                 FreeCAD.ActiveDocument.recompute()
00232             else:

```

```

00225     size = (ArchCommands.projectToVector(structure.Shape.copy(), face.normalAt(0, 0))).Length
00226     rebar = Arch.makeRebar(structure, sketch, diameter, int((size - diameter) / amount_spacing_value),
00227     f_cover)
00228     # Adds properties to the rebar object
00229     rebar.ViewObject.addProperty("App::PropertyString", "RebarShape", "RebarDialog", QT_TRANSLATE_NOOP("App::Property", "Shape of rebar")).RebarShape = "StraightRebar"
00230     rebar.ViewObject.setEditorMode("RebarShape", 2)
00231     rebar.addProperty("App::PropertyDistance", "FrontCover", "RebarDialog", QT_TRANSLATE_NOOP("App::Property", "Front cover of rebar")).FrontCover = f_cover
00232     rebar.setEditorMode("FrontCover", 2)
00233     rebar.addProperty("App::PropertyDistance", "RightTopCover", "RebarDialog", QT_TRANSLATE_NOOP("App::Property", "Right/Top Side cover of rebar")).RightTopCover = rt_cover
00234     rebar.setEditorMode("RightTopCover", 2)
00235     rebar.addProperty("App::PropertyDistance", "LeftBottomCover", "RebarDialog", QT_TRANSLATE_NOOP("App::Property", "Left/Bottom Side cover of rebar")).LeftBottomCover = lb_cover
00236     rebar.setEditorMode("LeftBottomCover", 2)
00237     rebar.addProperty("App::PropertyString", "CoverAlong", "RebarDialog", QT_TRANSLATE_NOOP("App::Property", "Cover along")).CoverAlong = coverAlong[0]
00238     rebar.setEditorMode("CoverAlong", 2)
00239     rebar.addProperty("App::PropertyDistance", "Cover", "RebarDialog", QT_TRANSLATE_NOOP("App::Property", "Cover of rebar along user selected side")).Cover = coverAlong[1]
00240     rebar.setEditorMode("Cover", 2)
00241     rebar.addProperty("App::PropertyBool", "AmountCheck", "RebarDialog", QT_TRANSLATE_NOOP("App::Property", "Amount radio button is checked")).AmountCheck
00242     rebar.setEditorMode("AmountCheck", 2)
00243     rebar.addProperty("App::PropertyDistance", "TrueSpacing", "RebarDialog", QT_TRANSLATE_NOOP("App::Property", "Spacing between of rebars")).TrueSpacing = amount_spacing_value
00244     rebar.setEditorMode("TrueSpacing", 2)
00245     rebar.addProperty("App::PropertyString", "Orientation", "RebarDialog", QT_TRANSLATE_NOOP("App::Property", "Shape of rebar")).Orientation = orientation
00246     rebar.setEditorMode("Orientation", 2)
00247     if amount_spacing_check:
00248         rebar.AmountCheck = True
00249     else:
00250         rebar.AmountCheck = False
00251     rebar.TrueSpacing = amount_spacing_value
00252     rebar.Label = "StraightRebar"
00253     FreeCAD.ActiveDocument.recompute()
00254     return rebar
00255 def editStraightRebar(Rebar, f_cover, coverAlong, rt_cover, lb_cover, diameter,
00256     amount_spacing_check, amount_spacing_value, orientation, structure = None, facename = None):
00257     sketch = Rebar.Base
00258     if structure and facename:
00259         sketch.Support = [(structure, facename)]
00260         FreeCAD.ActiveDocument.recompute()
00261     # Check if sketch support is empty.
00262     if not sketch.Support:
00263         showWarning("You have checked remove external geometry of base sketchs when needed.\nTo
unchecked Edit->Preferences->Arch.")
00264     return
00265     # Assigned values
00266     facename = sketch.Support[0][1][0]
00267     structure = sketch.Support[0][0]
00268     face = structure.Shape.Faces[getFaceNumber(facename) - 1]
00269     #StructurePRM = getTrueParametersOfStructure(structure)
00270     # Get parameters of the face where sketch of rebar is drawn
00271     FacePRM = getParametersOfFace(structure, facename)
00272     # Get points of Striaght rebar
00273     points = getpointsOfStraightRebar(FacePRM, rt_cover, lb_cover, coverAlong,
00274     orientation)
00275     sketch.movePoint(0, 1, points[0], 0)
00276     FreeCAD.ActiveDocument.recompute()
00277     sketch.movePoint(0, 2, points[1], 0)
00278     FreeCAD.ActiveDocument.recompute()
00279     Rebar.OffsetStart = f_cover
00280     Rebar.OffsetEnd = f_cover
00281     if amount_spacing_check:
00282         Rebar.Amount = amount_spacing_value
00283         FreeCAD.ActiveDocument.recompute()
00284         Rebar.AmountCheck = True
00285     else:
00286         size = (ArchCommands.projectToVector(structure.Shape.copy(), face.normalAt(0, 0))).Length
00287         Rebar.Amount = int((size - diameter) / amount_spacing_value)
00288         FreeCAD.ActiveDocument.recompute()
00289         Rebar.AmountCheck = False
00290         Rebar.FrontCover = f_cover
00291         Rebar.RightTopCover = rt_cover
00292         Rebar.LeftBottomCover = lb_cover
00293         Rebar.CoverAlong = coverAlong[0]
00294         Rebar.Cover = coverAlong[1]
00295         Rebar.TrueSpacing = amount_spacing_value
00296         Rebar.Diameter = diameter
00297         Rebar.Orientation = orientation
00298         FreeCAD.ActiveDocument.recompute()
00299     return Rebar

```

```

00299 def editDialog(vobj):
00300     FreeCADGui.Control.closeDialog()
00301     obj = _StraightRebarTaskPanel(vobj.Object)
00302     obj.form.frontCover.setText(str(vobj.Object.FrontCover))
00303     obj.form.r_sideCover.setText(str(vobj.Object.RightTopCover))
00304     obj.form.l_sideCover.setText(str(vobj.Object.LeftBottomCover))
00305     obj.form.bottomCover.setText(str(vobj.Object.Cover))
00306     obj.form.diameter.setText(str(vobj.Object.Diameter))
00307     obj.form.orientation.setCurrentIndex(obj.form.orientation.findText(str(vobj.Object.Orientation)))
00308     obj.form.coverAlong.setCurrentIndex(obj.form.coverAlong.findText(str(vobj.Object.CoverAlong)))
00309     if vobj.Object.AmountCheck:
00310         obj.form.amount.setValue(vobj.Object.Amount)
00311     else:
00312         obj.form.amount_radio.setChecked(False)
00313         obj.form.spacing_radio.setChecked(True)
00314         obj.form.amount.setDisabled(True)
00315         obj.form.spacing.setEnabled(True)
00316         obj.form.spacing.setText(str(vobj.Object.TrueSpacing))
00317     #obj.form.PickSelectedFace.setVisible(False)
00318     FreeCADGui.Control.showDialog(obj)
00319
00320 def CommandStraightRebar():
00321     selected_obj = check_selected_face()
00322     if selected_obj:
00323         FreeCADGui.Control.showDialog(_StraightRebarTaskPanel())

```

8.21 UShapeRebar.py File Reference

Classes

- class [UShapeRebar._UShapeRebarTaskPanel](#)

Namespaces

- [UShapeRebar](#)

Functions

- def [UShapeRebar.getpointsOfUShapeRebar](#) (FacePRM, r_cover, l_cover, b_cover, t_cover, orientation)
- def [UShapeRebar.makeUShapeRebar](#) (f_cover, b_cover, r_cover, l_cover, diameter, t_cover, rounding, amount_spacing_check, amount_spacing_value, orientation="Bottom", structure=None, facename=None)
- def [UShapeRebar.editUShapeRebar](#) (Rebar, f_cover, b_cover, r_cover, l_cover, diameter, t_cover, rounding, amount_spacing_check, amount_spacing_value, orientation, structure=None, facename=None)
- def [UShapeRebar.editDialog](#) (vobj)
- def [UShapeRebar.CommandUShapeRebar](#) ()

Variables

- string [UShapeRebar.__title__](#) = "UShapeRebar"
- string [UShapeRebar.__author__](#) = "Amritpal Singh"
- string [UShapeRebar.__url__](#) = "https://www.freecadweb.org"

8.22 UShapeRebar.py

```

00001 # --- coding: utf-8 ---
00002 # ****
00003 # *
00004 # * Copyright (c) 2017 - Amritpal Singh <amrit3701@gmail.com>
00005 # *
00006 # * This program is free software; you can redistribute it and/or modify
00007 # * it under the terms of the GNU Lesser General Public License (LGPL)
00008 # * as published by the Free Software Foundation; either version 2 of
00009 # * the License, or (at your option) any later version.
00010 # * for detail see the LICENCE text file.
00011 # *
00012 # * This program is distributed in the hope that it will be useful,
00013 # * but WITHOUT ANY WARRANTY; without even the implied warranty of
00014 # * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
00015 # * GNU Library General Public License for more details.
00016 # *
00017 # * You should have received a copy of the GNU Library General Public
00018 # * License along with this program; if not, write to the Free Software
00019 # * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307
00020 # * USA
00021 # *
00022 # ****
00023
00024 __title__ = "UShapeRebar"
00025 __author__ = "Amritpal Singh"
00026 __url__ = "https://www.freecadweb.org"
00027
00028 from PySide import QtCore, QtGui
00029 from Rebarfunc import *
00030 from PySide.QtCore import QT_TRANSLATE_NOOP
00031 from RebarDistribution import runRebarDistribution, removeRebarDistribution
00032 from PopUpImage import showPopUpImageDialog
00033 import FreeCAD
00034 import FreeCADGui
00035 import ArchCommands
00036 import os
00037 import sys
00038 import math
00039
00040 def getpointsOfUShapeRebar(FacePRM, r_cover, l_cover, b_cover, t_cover, orientation):
00041     """ getpointsOfUShapeRebar(FacePRM, RightCover, LeftCover, BottomCover, TopCover, Orientation):
00042     Return points of the Ushape rebar in the form of array for sketch.
00043     It takes four different orientations input i.e. 'Bottom', 'Top', 'Left', 'Right'.
00044     """
00045     if orientation == "Bottom":
00046         x1 = FacePRM[1][0] - FacePRM[0][0] / 2 + l_cover
00047         y1 = FacePRM[1][1] + FacePRM[0][1] / 2 - t_cover
00048         x2 = FacePRM[1][0] - FacePRM[0][0] / 2 + l_cover
00049         y2 = FacePRM[1][1] - FacePRM[0][1] / 2 + b_cover
00050         x3 = FacePRM[1][0] - FacePRM[0][0] / 2 + FacePRM[0][0] - r_cover
00051         y3 = FacePRM[1][1] - FacePRM[0][1] / 2 + b_cover
00052         x4 = FacePRM[1][0] - FacePRM[0][0] / 2 + FacePRM[0][0] - r_cover
00053         y4 = FacePRM[1][1] + FacePRM[0][1] / 2 - t_cover
00054     elif orientation == "Top":
00055         x1 = FacePRM[1][0] - FacePRM[0][0] / 2 + l_cover
00056         y1 = FacePRM[1][1] - FacePRM[0][1] / 2 + b_cover
00057         x2 = FacePRM[1][0] - FacePRM[0][0] / 2 + l_cover
00058         y2 = FacePRM[1][1] + FacePRM[0][1] / 2 - t_cover
00059         x3 = FacePRM[1][0] - FacePRM[0][0] / 2 + FacePRM[0][0] - r_cover
00060         y3 = FacePRM[1][1] + FacePRM[0][1] / 2 - t_cover
00061         x4 = FacePRM[1][0] - FacePRM[0][0] / 2 + FacePRM[0][0] - r_cover
00062         y4 = FacePRM[1][1] - FacePRM[0][1] / 2 + b_cover
00063     elif orientation == "Left":
00064         x1 = FacePRM[1][0] - FacePRM[0][0] / 2 + FacePRM[0][0] - r_cover
00065         y1 = FacePRM[1][1] + FacePRM[0][1] / 2 - t_cover
00066         x2 = FacePRM[1][0] - FacePRM[0][0] / 2 + l_cover
00067         y2 = FacePRM[1][1] + FacePRM[0][1] / 2 - t_cover
00068         x3 = FacePRM[1][0] - FacePRM[0][0] / 2 + l_cover
00069         y3 = FacePRM[1][1] - FacePRM[0][1] / 2 + b_cover
00070         x4 = FacePRM[1][0] - FacePRM[0][0] / 2 + FacePRM[0][0] - r_cover
00071         y4 = FacePRM[1][1] - FacePRM[0][1] / 2 + b_cover
00072     elif orientation == "Right":
00073         x1 = FacePRM[1][0] - FacePRM[0][0] / 2 + l_cover
00074         y1 = FacePRM[1][1] + FacePRM[0][1] / 2 - t_cover
00075         x2 = FacePRM[1][0] - FacePRM[0][0] / 2 + FacePRM[0][0] - r_cover
00076         y2 = FacePRM[1][1] + FacePRM[0][1] / 2 - t_cover
00077         x3 = FacePRM[1][0] - FacePRM[0][0] / 2 + FacePRM[0][0] - r_cover
00078         y3 = FacePRM[1][1] - FacePRM[0][1] / 2 + b_cover
00079         x4 = FacePRM[1][0] - FacePRM[0][0] / 2 + l_cover
00080         y4 = FacePRM[1][1] - FacePRM[0][1] / 2 + b_cover
00081     return [FreeCAD.Vector(x1, y1, 0), FreeCAD.Vector(x2, y2, 0),
00082             FreeCAD.Vector(x3, y3, 0), FreeCAD.Vector(x4, y4, 0)]
00083
00084 class _UShapeRebarTaskPanel:

```

```

00085     def __init__(self, Rebar = None):
00086         self.CustomSpacing = None
00087         if not Rebar:
00088             selected_obj = FreeCADGui.Selection.getSelectionEx()[0]
00089             self.SelectedObj = selected_obj.Object
00090             self.FaceName = selected_obj.SubElementNames[0]
00091         else:
00092             self.FaceName = Rebar.Base.Support[0][1][0]
00093             self.SelectedObj = Rebar.Base.Support[0][0]
00094         self.form = FreeCADGui.PySideUic.loadUi(os.path.splitext(__file__)[0] + ".ui")
00095         self.form.setWindowTitle(QtGui.QApplication.translate("RebarAddon", "U-Shape Rebar", None))
00096         self.form.orientation.addItems(["Bottom", "Top", "Right", "Left"])
00097         self.form.amount_radio.clicked.connect(self.amount_radio_clicked)
00098         self.form.spacing_radio.clicked.connect(self.spacing_radio_clicked)
00099         self.form.customSpacing.clicked.connect(lambda: runRebarDistribution(self))
00100         self.form.removeCustomSpacing.clicked.connect(lambda:
00101             removeRebarDistribution(self))
00102         self.form.PickSelectedFace.clicked.connect(lambda: getSelectedFace(self))
00103         self.form.orientation.currentIndexChanged.connect(self.getOrientation)
00104         self.form.image.setPixmap(QtGui.QPixmap(os.path.split(os.path.abspath(__file__))[0] + "icons/UShapeRebarBottom.svg"))
00105         self.form.toolButton.setIcon(self.form.toolButton.style().standardIcon(
00106             QtGui.QStyle.SP_DialogHelpButton))
00107         self.form.toolButton.clicked.connect(lambda: showPopUpImageDialog(os.path.split(
00108             os.path.abspath(__file__))[0] + "/icons/UShapeRebarDetailed.svg"))
00109         self.Rebar = Rebar
00110
00111     def getOrientation(self):
00112         orientation = self.form.orientation.currentText()
00113         if orientation == "Bottom":
00114             self.form.image.setPixmap(QtGui.QPixmap(os.path.split(os.path.abspath(__file__))[0] + "icons/UShapeRebarBottom.svg"))
00115         elif orientation == "Top":
00116             self.form.image.setPixmap(QtGui.QPixmap(os.path.split(os.path.abspath(__file__))[0] + "icons/UShapeRebarTop.svg"))
00117         elif orientation == "Right":
00118             self.form.image.setPixmap(QtGui.QPixmap(os.path.split(os.path.abspath(__file__))[0] + "icons/UShapeRebarRight.svg"))
00119         else:
00120             self.form.image.setPixmap(QtGui.QPixmap(os.path.split(os.path.abspath(__file__))[0] + "icons/UShapeRebarLeft.svg"))
00121
00122     def getStandardButtons(self):
00123         return int(QtGui.QDialogButtonBox.Ok) | int(QtGui.QDialogButtonBox.Apply) | int(
00124             QtGui.QDialogButtonBox.Cancel)
00125
00126     def clicked(self, button):
00127         if button == int(QtGui.QDialogButtonBox.Apply):
00128             self.accept(button)
00129
00130     def accept(self, signal = None):
00131         f_cover = self.form.frontCover.text()
00132         f_cover = FreeCAD.Units.Quantity(f_cover).Value
00133         b_cover = self.form.bottomCover.text()
00134         b_cover = FreeCAD.Units.Quantity(b_cover).Value
00135         r_cover = self.form.r_sideCover.text()
00136         r_cover = FreeCAD.Units.Quantity(r_cover).Value
00137         l_cover = self.form.l_sideCover.text()
00138         l_cover = FreeCAD.Units.Quantity(l_cover).Value
00139         t_cover = self.form.topCover.text()
00140         t_cover = FreeCAD.Units.Quantity(t_cover).Value
00141         diameter = self.form.diameter.text()
00142         diameter = FreeCAD.Units.Quantity(diameter).Value
00143         rounding = self.form.rounding.value()
00144         orientation = self.form.orientation.currentText()
00145         amount_check = self.form.amount_radio.isChecked()
00146         spacing_check = self.form.spacing_radio.isChecked()
00147         if not self.Rebar:
00148             if amount_check:
00149                 amount = self.form.amount.value()
00150                 rebar = makeUShapeRebar(f_cover, b_cover, r_cover, l_cover, diameter,
00151                 t_cover, rounding, True, amount, orientation, self.SelectedObj, self.
00152                 FaceName)
00153             elif spacing_check:
00154                 spacing = self.form.spacing.text()
00155                 spacing = FreeCAD.Units.Quantity(spacing).Value
00156                 rebar = makeUShapeRebar(f_cover, b_cover, r_cover, l_cover, diameter,
00157                 t_cover, rounding, False, spacing, orientation, self.SelectedObj, self.
00158                 FaceName)
00159         else:
00160             if amount_check:
00161                 amount = self.form.amount.value()
00162                 rebar = editUShapeRebar(self.Rebar, f_cover, b_cover, r_cover, l_cover,
00163                 diameter, t_cover, rounding, True, amount, orientation, self.SelectedObj, self.
00164                 FaceName)
00165             elif spacing_check:
00166                 spacing = self.form.spacing.text()

```

```

00157         spacing = FreeCAD.Units.Quantity(spacing).Value
00158         rebar = editUShapeRebar(self.Rebar, f_cover, b_cover, r_cover, l_cover,
00159             diameter, t_cover, rounding, False, spacing, orientation, self.SelectedObj, self.
00160             FaceName)
00161             if self.CustomSpacing:
00162                 rebar.CustomSpacing = self.CustomSpacing
00163                 FreeCAD.ActiveDocument.recompute()
00164             self.Rebar = rebar
00165             if signal == int(QtGui.QDialogButtonBox.Apply):
00166                 pass
00167             else:
00168                 FreeCADGui.Control.closeDialog(self)
00169
00170     def amount_radio_clicked(self):
00171         self.form.spacing.setEnabled(False)
00172         self.form.amount.setEnabled(True)
00173
00174     def spacing_radio_clicked(self):
00175         self.form.amount.setEnabled(False)
00176         self.form.spacing.setEnabled(True)
00177
00178 def makeUShapeRebar(f_cover, b_cover, r_cover, l_cover, diameter, t_cover, rounding,
00179     amount_spacing_check, amount_spacing_value, orientation = "Bottom", structure = None, facename = None):
00180     """ makeUShapeRebar(FrontCover, BottomCover, RightCover, LeftCover, Diameter, Topcover, Rounding,
00181     AmountSpacingCheck, AmountSpacingValue,
00182     Orientation, Structure, Facename): Adds the U-Shape reinforcement bar to the selected structural
00183     object.
00184     It takes four different types of orientations as input i.e 'Bottom', 'Top', 'Right', 'Left'.
00185     """
00186     if not structure and not facename:
00187         selected_obj = FreeCADGui.Selection.getSelectionEx()[0]
00188         structure = selected_obj.Object
00189         facename = selected_obj.SubElementNames[0]
00190         face = structure.Shape.Faces[getFaceNumber(facename) - 1]
00191         #StructurePRM = getTrueParametersOfStructure(structure)
00192         FacePRM = getParametersOfFace(structure, facename)
00193         if not FacePRM:
00194             FreeCAD.Console.PrintError("Cannot identified shape or from which base object sturctural element is
00195             derived\n")
00196             return
00197         # Get points of U-Shape rebar
00198         points = getpointsOfUShapeRebar(FacePRM, r_cover, l_cover, b_cover, t_cover,
00199             orientation)
00200         import Part
00201         import Arch
00202         sketch = FreeCAD.activeDocument().addObject('Sketcher::SketchObject', 'Sketch')
00203         sketch.MapMode = "FlatFace"
00204         sketch.Support = [(structure, facename)]
00205         FreeCAD.ActiveDocument.recompute()
00206         sketch.addGeometry(Part.LineSegment(points[0], points[1]), False)
00207         sketch.addGeometry(Part.LineSegment(points[1], points[2]), False)
00208         import Sketcher
00209         sketch.addGeometry(Part.LineSegment(points[2], points[3]), False)
00210         if amount_spacing_check:
00211             rebar = Arch.makeRebar(structure, sketch, diameter, amount_spacing_value, f_cover)
00212             FreeCAD.ActiveDocument.recompute()
00213         else:
00214             size = (ArchCommands.projectToVector(structure.Shape.copy(), face.normalAt(0, 0))).Length
00215             rebar = Arch.makeRebar(structure, sketch, diameter, int((size - diameter) / amount_spacing_value),
00216             f_cover)
00217             rebar.Rounding = rounding
00218             # Adds properties to the rebar object
00219             rebar.ViewObject.addProperty("App::PropertyString", "RebarShape", "RebarDialog", QT_TRANSLATE_NOOP("App::Property", "Shape of rebar")).RebarShape = "UShapeRebar"
00220             rebar.ViewObject.setEditorMode("RebarShape", 2)
00221             rebar.addProperty("App::PropertyDistance", "FrontCover", "RebarDialog", QT_TRANSLATE_NOOP("App::Property", "Front cover of rebar")).FrontCover = f_cover
00222             rebar.setEditorMode("FrontCover", 2)
00223             rebar.addProperty("App::PropertyDistance", "RightCover", "RebarDialog", QT_TRANSLATE_NOOP("App::Property", "Right Side cover of rebar")).RightCover = r_cover
00224             rebar.setEditorMode("RightCover", 2)
00225             rebar.addProperty("App::PropertyDistance", "LeftCover", "RebarDialog", QT_TRANSLATE_NOOP("App::Property", "Left Side cover of rebar")).LeftCover = l_cover
00226             rebar.setEditorMode("LeftCover", 2)
00227             rebar.addProperty("App::PropertyDistance", "BottomCover", "RebarDialog", QT_TRANSLATE_NOOP("App::Property", "Bottom cover of rebar")).BottomCover = b_cover
00228             rebar.setEditorMode("BottomCover", 2)
00229             rebar.addProperty("App::PropertyBool", "AmountCheck", "RebarDialog", QT_TRANSLATE_NOOP("App::Property", "Amount radio button is checked")).AmountCheck
00230             rebar.setEditorMode("AmountCheck", 2)
00231             rebar.addProperty("App::PropertyDistance", "TopCover", "RebarDialog", QT_TRANSLATE_NOOP("App::Property", "Top cover of rebar")).TopCover = t_cover
00232             rebar.setEditorMode("TopCover", 2)
00233             rebar.addProperty("App::PropertyDistance", "TrueSpacing", "RebarDialog", QT_TRANSLATE_NOOP("App::Property", "Spacing between of rebars")).TrueSpacing = amount_spacing_value
00234             rebar.setEditorMode("TrueSpacing", 2)

```

```

00228     rebar.addProperty("App::PropertyString", "Orientation", "RebarDialog", QT_TRANSLATE_NOOP("App::Property
00229     ", "Shape of rebar")).Orientation = orientation
00230     rebar.setEditorMode("Orientation", 2)
00231     if amount_spacing_check:
00232         rebar.AmountCheck = True
00233     else:
00234         rebar.AmountCheck = False
00235     rebar.TrueSpacing = amount_spacing_value
00236     rebar.Label = "UShapeRebar"
00237     FreeCAD.ActiveDocument.recompute()
00238     return rebar
00239 def editUShapeRebar(Rebar, f_cover, b_cover, r_cover, l_cover, diameter, t_cover, rounding,
00240     amount_spacing_check, amount_spacing_value, orientation, structure = None, facename = None):
00241     sketch = Rebar.Base
00242     if structure and facename:
00243         sketch.Support = [(structure, facename)]
00244     # Check if sketch support is empty.
00245     if not sketch.Support:
00246         showWarning("You have checked remove external geometry of base sketchs when needed.\nTo
unchecked Edit->Preferences->Arch.")
00247         return
00248     # Assigned values
00249     facename = sketch.Support[0][1][0]
00250     structure = sketch.Support[0][0]
00251     face = structure.Shape.Faces[getFaceNumber(facename) - 1]
00252     #StructurePRM = getTrueParametersOfStructure(structure)
00253     # Get parameters of the face where sketch of rebar is drawn
00254     FacePRM = getParametersOfFace(structure, facename)
00255     # Get points of U-Shape rebar
00256     points = getpointsOfUShapeRebar(FacePRM, r_cover, l_cover, b_cover, t_cover,
00257     orientation)
00258     sketch.movePoint(0, 1, points[0], 0)
00259     FreeCAD.ActiveDocument.recompute()
00260     sketch.movePoint(0, 2, points[1], 0)
00261     FreeCAD.ActiveDocument.recompute()
00262     sketch.movePoint(1, 1, points[1], 0)
00263     FreeCAD.ActiveDocument.recompute()
00264     sketch.movePoint(2, 1, points[2], 0)
00265     FreeCAD.ActiveDocument.recompute()
00266     sketch.movePoint(2, 2, points[3], 0)
00267     FreeCAD.ActiveDocument.recompute()
00268     Rebar.OffsetStart = f_cover
00269     Rebar.OffsetEnd = f_cover
00270     if amount_spacing_check:
00271         Rebar.Amount = amount_spacing_value
00272         FreeCAD.ActiveDocument.recompute()
00273         Rebar.AmountCheck = True
00274     else:
00275         size = (ArchCommands.projectToVector(structure.Shape.copy(), face.normalAt(0, 0))).Length
00276         Rebar.Amount = int((size - diameter) / amount_spacing_value)
00277         FreeCAD.ActiveDocument.recompute()
00278         Rebar.AmountCheck = False
00279     Rebar.Diameter = diameter
00280     Rebar.FrontCover = f_cover
00281     Rebar.RightCover = r_cover
00282     Rebar.LeftCover = l_cover
00283     Rebar.BottomCover = b_cover
00284     Rebar.TopCover = t_cover
00285     Rebar.Rounding = rounding
00286     Rebar.TrueSpacing = amount_spacing_value
00287     Rebar.Orientation = orientation
00288     FreeCAD.ActiveDocument.recompute()
00289     return Rebar
00290
00291 def editDialog(vobj):
00292     FreeCADGui.Control.closeDialog()
00293     obj = _UShapeRebarTaskPanel(vobj.Object)
00294     obj.form.frontCover.setText(str(vobj.Object.FrontCover))
00295     obj.form.r_sideCover.setText(str(vobj.Object.RightCover))
00296     obj.form.l_sideCover.setText(str(vobj.Object.LeftCover))
00297     obj.form.bottomCover.setText(str(vobj.Object.BottomCover))
00298     obj.form.diameter.setText(str(vobj.Object.Diameter))
00299     obj.form.topCover.setText(str(vobj.Object.TopCover))
00300     obj.form.rounding.setValue(vobj.Object.Rounding)
00301     obj.form.orientation.setCurrentIndex(obj.form.orientation.findText(str(vobj.Object.Orientation)))
00302     if vobj.Object.AmountCheck:
00303         obj.form.amount.setValue(vobj.Object.Amount)
00304     else:
00305         obj.form.amount_radio.setChecked(False)
00306         obj.form.spacing_radio.setChecked(True)
00307         obj.form.amount.setDisabled(True)
00308         obj.form.spacing.setEnabled(True)
00309         obj.form.spacing.setText(str(vobj.Object.TrueSpacing))
00310     #obj.form.PickSelectedFace.setVisible(False)

```

```
00311     FreeCADGui.Control.showDialog(obj)
00312
00313 def CommandUShapeRebar():
00314     selected_obj = check_selected_face()
00315     if selected_obj:
00316         FreeCADGui.Control.showDialog(_UShapeRebarTaskPanel())
```