Rebar Addon for FreeCAD

Generated by Doxygen 1.8.11

Contents

1	Reb	ar Addo	on for Free	eCAD	1
2	Nam	nespace	Index		3
	2.1	Names	space List		3
3	Hier	archica	l Index		5
	3.1	Class	Hierarchy		5
4	Clas	ss Index			7
	4.1	Class	List		7
5	File	Index			9
	5.1	File Lis	st		9
6	Nam	nespace	Docume	ntation	11
	6.1	BentSl	hapeReba	r 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_cvalue, orientation, structure=None, facename=None)	12
			6.1.1.3	editDialog(vobj)	14
			6.1.1.4	getpointsOfBentShapeRebar(FacePRM, I_cover, r_cover, b_cover, t_cover, bentLength, bentAngle, orientation)	14
			6.1.1.5	makeBentShapeRebar(f_cover, b_cover, I_cover, r_cover, diameter, t_cover, bentLength, bentAngle, rounding, amount_spacing_check, amount_spacing_cvalue, orientation=""Bottom Left"", structure=None, facename=None)	16
		612	Variable	Documentation	18

iv CONTENTS

		6.1.2.1	author	18
		6.1.2.2	title	18
		6.1.2.3	url	18
6.2	Helical	Rebar Nar	mespace 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	LShape	eRebar Na	amespace 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, I_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, I_cover, r_cover, b_cover, t_cover, orientation)	27
		6.3.1.5	makeLShapeRebar(f_cover, b_cover, I_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

CONTENTS

6.4	PopUp	Image Nar	mespace Reference	30
	6.4.1	Function	Documentation	30
		6.4.1.1	showPopUpImageDialog(img)	30
	6.4.2	Variable I	Documentation	30
		6.4.2.1	author	30
		6.4.2.2	title	31
		6.4.2.3	url	31
6.5	Rebarl	Distribution	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	getupleOfCustomSpacing(span_string)	32
		6.5.1.3	removeRebarDistribution(self)	32
		6.5.1.4	runRebarDistribution(self)	32
	6.5.2	Variable I	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	Rebart	unc Name	space 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

vi

		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	Rebar	Tools Nam	espace 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	Namespa	ce 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, I_cover, r_cover, t_cover, b_cover, f_cover, bentAngle, bent← Factor, diameter, rounding, amount_spacing_check, amount_spacing_value, structure=None, facename=None)	47
		6.8.1.4	getpointsOfStirrup(FacePRM, I_cover, r_cover, t_cover, b_cover, bentAngle, bentFactor, diameter, rounding, facenormal)	48
		6.8.1.5	makeStirrup(I_cover, r_cover, t_cover, b_cover, f_cover, bentAngle, bent← Factor, 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	Straigh	ntRebar Na	amespace Reference	52

CONTENTS vii

	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	UShap	eRebar Na	amespace 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_ \leftarrow 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

viii CONTENTS

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

CONTENTS

		7.2.4.4	SelectedObj	72
7.3	LShap	eRebarL	ShapeRebarTaskPanel Class Reference	72
	7.3.1	Detailed	Description	73
	7.3.2	Construc	ctor & 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	Rebarl	Distribution	nRebarDistributionDialog Class Reference	77
	7.4.1	Detailed	Description	78
	7.4.2	Construc	ctor & 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

CONTENTS

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

CONTENTS xi

		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	UShap	eRebarl	UShapeRebarTaskPanel Class Reference	. 90
	7.7.1	Detailed	Description	. 90
	7.7.2	Construc	ctor & 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	Rebar ⁻	Tools.Bent	tShapeRebarTool 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	Rebar ⁻	Tools.Helic	calRebarTool Class Reference	. 96
	7.9.1	Detailed	Description	. 96
	7.9.2	Member	Function Documentation	. 96

xii CONTENTS

		7.9.2.1	Activated(self)	 96
		7.9.2.2	GetResources(self)	 97
		7.9.2.3	IsActive(self)	 97
7.10	RebarT	ools.LSha	apeRebarTool Class Reference	 97
	7.10.1	Detailed I	Description	 98
	7.10.2	Member I	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	PopUp	lmage.Pop	pUpImage Class Reference	 99
	7.11.1	Detailed I	Description	 100
	7.11.2	Construc	ctor & Destructor Documentation	 100
		7.11.2.1	init(self, img)	 100
	7.11.3	Member I	Data Documentation	 100
		7.11.3.1	image	 100
		7.11.3.2	verticalLayout	 100
7.12	RebarT	ools.Stirru	upTool Class Reference	 100
	7.12.1	Detailed I	Description	 101
	7.12.2	Member I	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	RebarT	ools.Straig	ghtRebarTool Class Reference	 102
	7.13.1	Detailed I	Description	 102
	7.13.2	Member I	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	RebarT	ools.USha	apeRebarTool Class Reference	 103
	7.14.1	Detailed I	Description	 104
	7.14.2	Member I	Function Documentation	 104
		7.14.2.1	Activated(self)	 104
		7.14.2.2	GetResources(self)	 104
		7.14.2.3	IsActive(self)	 104

CONTENTS xiii

8	File I	e Documentation							
	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. \leftarrow py, _StraightRebarTaskPanel class is present. This class loads UI(present in StriaghtRebar.ui file) in the task panel of FreeCAD. First time when a user clicks on Apply or Ok button, then makeStraight \leftarrow Rebar function is executed and after that when user want to change the properties of Straight rebar then edit \leftarrow StraightRebar function is executed.

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:

entShapeRebar	1
elicalRebar	8
ShapeRebar	24
pUpImage	30
ebarDistribution	
ebarfunc	
ebarTools	
irrup	16
raightRebar	
ShapeRebar	57

4 Namespace Index

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

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

BentShapeRebarBentShapeRebarTaskPanel
HelicalRebarTaskPanel
LShapeRebar_LShapeRebarTaskPanel
RebarDistributionRebarDistributionDialog
StirrupStirrupTaskPanel
StraightRebar_StraightRebarTaskPanel
UShapeRebar_UShapeRebarTaskPanel
RebarTools.BentShapeRebarTool
RebarTools.HelicalRebarTool
RebarTools.LShapeRebarTool
QDialog
PopUpImage.PopUpImage
RebarTools.StirrupTool
RebarTools.StraightRebarTool
RebarTools.UShapeRebarTool

6 Hierarchical Index

Chapter 4

Class Index

4.1 Class List

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

BentShapeRebarBentShapeRebarTaskPanel
HelicalRebar_HelicalRebarTaskPanel
LShapeRebar_LShapeRebarTaskPanel
RebarDistributionRebarDistributionDialog
StirrupStirrupTaskPanel
StraightRebar_StraightRebarTaskPanel
UShapeRebar_UShapeRebarTaskPanel
RebarTools.BentShapeRebarTool
RebarTools.HelicalRebarTool
RebarTools.LShapeRebarTool
PopUpImage.PopUpImage
RebarTools.StirrupTool
RebarTools.StraightRebarTool
RebarTools.UShapeRebarTool

8 Class Index

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

tShapeRebar.py	105
calRebar.py	110
apeRebar.py	114
Uplmage.py	119
parDistribution.py	120
parfunc.py	122
parTools.py	127
rup.py	129
ightRebar.py	
napeRebar.py	139

10 File Index

Chapter 6

Namespace Documentation

6.1 BentShapeRebar Namespace Reference

Classes

• class _BentShapeRebarTaskPanel

Functions

- def getpointsOfBentShapeRebar (FacePRM, I_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.

Here is the call graph for this function:



Here is the caller graph for this function:



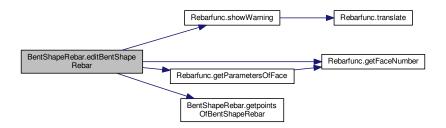
6.1.1.2 def BentShapeRebar.editBentShapeRebar (Rebar, f_cover, b_cover, l_cover, r_cover, diameter, t_cover, bentLength, bentAngle, rounding, amount_spacing_check, amount_spacing_value, orientation, structure = None, facename = None)

Definition at line 270 of file BentShapeRebar.py.

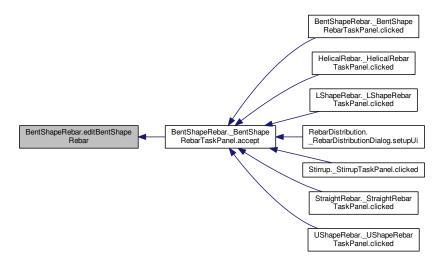
```
00270 def editBentShapeRebar(Rebar, f cover, b cover, l cover, r cover, diameter, t cover,
      bentLength, bentAngle, rounding, amount_spacing_check, amount_spacing_value, orientation, structure = None,
      facename = None):
00271
          sketch = Rebar.Base
          if structure and facename:
00272
00273
              sketch.Support = [(structure, facename)]
00274
          # Check if sketch support is empty.
00275
         if not sketch.Support:
              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]
          structure = sketch.Support[0][0]
00280
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
          points = getpointsOfBentShapeRebar(FacePRM, l_cover, r_cover, b_cover, t_cover
00286
      , 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()
          sketch.movePoint(1, 1, points[1], 0)
00291
00292
          FreeCAD.ActiveDocument.recompute()
00293
          sketch.movePoint(1, 2, points[2], 0)
00294
          FreeCAD.ActiveDocument.recompute()
00295
00296
          sketch.movePoint(2, 1, points[2], 0)
00297
          FreeCAD.ActiveDocument.recompute()
00298
          sketch.movePoint(2, 2, points[3], 0)
00299
          FreeCAD.ActiveDocument.recompute()
```

```
00300
            sketch.movePoint(3, 1, points[3], 0)
00301
            FreeCAD.ActiveDocument.recompute()
00302
            sketch.movePoint(3, 2, points[4], 0)
00303
           {\tt 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
            if amount_spacing_check:
    Rebar.Amount = amount_spacing_value
00312
00313
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
           Rebar.FrontCover = f_cover
Rebar.LeftCover = 1_cover
00322
00323
00324
            Rebar.RightCover = r_cover
00325
            Rebar.BottomCover = b_cover
00326
            Rebar.TopCover = t_cover
           Rebar.BentLength = bentLength
Rebar.BentAngle = bentAngle
Rebar.Rounding = rounding
Rebar.TrueSpacing = amount_spacing_value
Rebar.Orientation = orientation
00327
00328
00329
00330
00331
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, I_cover, r_cover, b_cover, t_cover, bentLength, bentAngle, orientation)

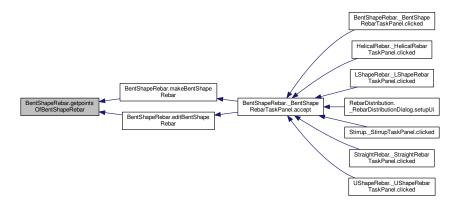
```
getpointsOfBentShapeRebar(FacePRM, LeftCover, RightCover, BottomCover, TopCover, BentLength, BentAngle, Orient
Return points of the LShape rebar in the form of array for sketch.
It takes four different orientations input i.e. 'Bottom', 'Top', 'Left', 'Right'.
```

Definition at line 40 of file BentShapeRebar.py.

```
00040 def getpointsOfBentShapeRebar(FacePRM, l_cover, r_cover, b_cover, t_cover,
      bentLength, bentAngle, orientation):
00041
            """ 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'.
00042
00043
00045
            if orientation == "Bottom":
               x1 = FacePRM[1][0] - FacePRM[0][0] / 2 + 1_cover
y1 = FacePRM[1][1] + FacePRM[0][1] / 2 - t_cover
00046
00047
                x2 = x1 + bentLength
00048
                y2 = y1
dis = (FacePRM[0][1] - t_cover - b_cover) * math.tan(math.radians(bentAngle - 90))
00049
00050
00051
                x3 = x2 + dis
00052
                 y3 = FacePRM[1][1] - FacePRM[0][1] / 2 + b_cover
                 x4 = FacePRM[1][0] + FacePRM[0][0] / 2 - r_cover - bentLength - dis
00053
                 y4 = y3
00054
00055
                 x5 = x4 + dis
                y5 = y2
00056
                x6 = x5 + bentLength
00057
                y6 = y5
00058
           elif orientation == "Top":
00059
              x1 = FacePRM[1][0] - FacePRM[0][0] / 2 + 1_cover
y1 = FacePRM[1][1] - FacePRM[0][1] / 2 + b_cover
00060
00061
00062
                x2 = x1 + bentLength
00063
                y2 = y1
00064
                 dis = (FacePRM[0][1] - t_cover - b_cover) * math.tan(math.radians(bentAngle - 90))
                x3 = x2 + dis
00065
                x3 = FacePRM[1][1] + FacePRM[0][1] / 2 - t_cover
x4 = FacePRM[1][0] + FacePRM[0][0] / 2 - r_cover - bentLength - dis
00066
00067
00068
                y4 = y3
00069
                 x5 = x4 + dis
00070
                y5 = y2
00071
                 x6 = x5 + bentLength
           y6 = y5

elif orientation == "Left":
00072
00073
               x1 = FacePRM[1][0] + FacePRM[0][0] / 2 - r_cover
y1 = FacePRM[1][1] + FacePRM[0][1] / 2 - t_cover
00074
00075
                x^2 = x^1
00076
                y2 = y1 - bentLength
00077
                 dis = (FacePRM[0][0] - r_cover - l_cover) * math.tan(math.radians(bentAngle - 90))
00078
00079
                x3 = FacePRM[1][0] - FacePRM[0][0] / 2 + 1_cover
                y3 = y2 - dis
00080
00081
                 x4 = x3
                y4 = FacePRM[1][1] - FacePRM[0][1] / 2 + b_cover + bentLength + dis
00082
00083
                 x5 = x2
00084
                y5 = y4 - dis
                 x6 = x5
00085
                y6 = y5 - bentLength
00086
           elif orientation == "Right":
    x1 = FacePRM[1][0] - FacePRM[0][0] / 2 + 1_cover
    y1 = FacePRM[1][1] + FacePRM[0][1] / 2 - t_cover
00087
00088
00089
00090
                 x2 = x1
                y2 = y1 - bentLength
dis = (FacePRM[0][0] - r_cover - 1_cover) * math.tan(math.radians(bentAngle - 90))
00091
00092
00093
                 x3 = FacePRM[1][0] + FacePRM[0][0] / 2 - r_cover
00094
                y3 = y2 - dis
00095
                 x4 = x3
                 y4 = FacePRM[1][1] - FacePRM[0][1] / 2 + b\_cover + bentLength + dis
00096
00097
                 x5 = x2
                y5 = y4 - dis
00098
                x6 = x5
00099
00100
                y6 = y5 - bentLength
           return [FreeCAD.Vector(x1, y1, 0), FreeCAD.Vector(x2, y2, 0), FreeCAD.Vector(x3, y3, 0), FreeCAD.Vector(x4, y4, 0),
00101
00102
00103
                    FreeCAD.Vector(x5, y5, 0), FreeCAD.Vector(x6, y6, 0)]
00104
```

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 & selected structural object.

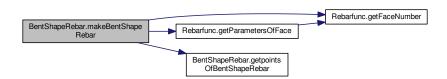
It takes four different orientations input i.e. 'Bottom', 'Top', 'Left', 'Right'.

Definition at line 200 of file BentShapeRebar.py.

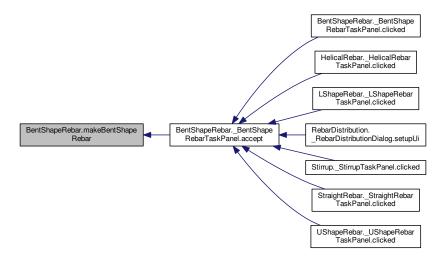
```
00200 def makeBentShapeRebar(f_cover, b_cover, l_cover, r_cover, diameter, t_cover, bentLength,
       bentAngle, rounding, amount_spacing_check, amount_spacing_value, orientation = "Bottom Left", structure =
      None, facename = None):
""" makeBentShapeRebar(FrontCover, BottomCover, LeftCover, RightCover, Diameter, TopCover, BentLength,
00201
       BentAngle, Rounding,
00202
         AmountSpacingCheck, AmountSpacingValue, Orientation, Structure, Facename): Adds the Bent-Shape
       reinforcement bar to the
00203
         selected structural object.
          It takes four different orientations input i.e. 'Bottom', 'Top', 'Left', 'Right'.
00204
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)
          FacePRM = getParametersOfFace(structure, facename)
00212
00213
          if not FacePRM:
00214
              FreeCAD.Console.PrintError("Cannot identified shape or from which base object sturctural element is
       derived(n")
00215
00216
          # Get points of L-Shape rebar
          points = getpointsOfBentShapeRebar(FacePRM, l_cover, r_cover, b_cover, t_cover
00217
      , bentLength, bentAngle, orientation)
00218
          import Part
00219
          import Arch
00220
          sketch = FreeCAD.activeDocument().addObject('Sketcher::SketchObject', 'Sketch')
00221
          sketch.MapMode = "FlatFace"
          sketch.Support = [(structure, facename)]
00222
00223
          FreeCAD.ActiveDocument.recompute()
00224
          sketch.addGeometry(Part.LineSegment(points[0], points[1]), False)
00225
          sketch.addGeometry(Part.LineSegment(points[1], points[2]), False)
00226
          sketch.addGeometry(Part.LineSegment(points[2], points[3]), False)
00227
          sketch.addGeometry(Part.LineSegment(points[3], points[4]), False)
00228
          sketch.addGeometry(Part.LineSegment(points[4], points[5]), False)
00229
          import Sketcher
00230
          if amount_spacing_check:
00231
              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),
       f_cover)
00236
            rebar.Rounding = rounding
            # Adds properties to the rebar object
00237
00238
            rebar.ViewObject.addProperty("App::PropertyString", "RebarShape", "RebarDialog", QT_TRANSLATE_NOOP("
       App::Property", "Shape of rebar")).RebarShape = "BentShapeRebar"
       rebar.ViewObject.setEditorMode("RebarShape", 2)
rebar.addProperty("App::PropertyDistance", "FrontCover", "RebarDialog", QT_TRANSLATE_NOOP("
App::Property", "Front cover of rebar")).FrontCover = f_cover
00239
00240
            rebar.setEditorMode("FrontCover", 2)
00241
            rebar.addProperty("App::PropertyDistance", "LeftCover", "RebarDialog", QT_TRANSLATE_NOOP("App::Property
00242
           "Left Side cover of rebar")).LeftCover = 1_cover
00243
            rebar.setEditorMode("LeftCover", 2)
       rebar.addProperty("App::PropertyDistance", "RightCover", "RebarDialog", QT_TRANSLATE_NOOP("App::Property", "Right Side cover of rebar")).RightCover = r_cover rebar.setEditorMode("RightCover", 2)
00244
00245
            rebar.addProperty("App::PropertyDistance", "BottomCover", "RebarDialog", QT_TRANSLATE_NOOP("
00246
                           "Bottom cover of rebar")).BottomCover = b_cover
       App::Property",
            rebar.setEditorMode("BottomCover", 2)
rebar.addProperty("App::PropertyBool", "AmountCheck", "RebarDialog", QT_TRANSLATE_NOOP("App::Property",
00247
00248
         "Amount radio button is checked")).AmountCheck
00249
            rebar.setEditorMode("AmountCheck", 2)
            rebar.addProperty("App::PropertyDistance", "TopCover", "RebarDialog", QT_TRANSLATE_NOOP("App::Property"
          "Top cover of rebar")).TopCover = t_cover
            rebar.setEditorMode("TopCover", 2)
00251
       rebar.addProperty("App::PropertyDistance", "TrueSpacing", "RebarDialog", QT_TRANSLATE_NOOP("
App::Property", "Spacing between of rebars")).TrueSpacing = amount_spacing_value
rebar.addProperty("App::PropertyString", "Orientation", "RebarDialog", QT_TRANSLATE_NOOP("App::Property
00252
00253
       ", "Shape of rebar")).Orientation = orientation
00254
            rebar.setEditorMode("Orientation", 2)
00255
            rebar.setEditorMode("TrueSpacing", 2)
            rebar.addProperty("App::PropertyDistance", "BentLength", "RebarDialog", QT_TRANSLATE_NOOP(":Property", "BentLength cover of rebar")).BentLength = bentLength
00256
       App::Property",
00257
            rebar.setEditorMode("BentLength", 2)
            rebar.addProperty("App::PropertyDistance", "BentAngle", "RebarDialog", QT_TRANSLATE_NOOP("App::Property
00258
       ", "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
            {\tt 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.

Here is the call graph for this function:



Here is the caller graph for this function:

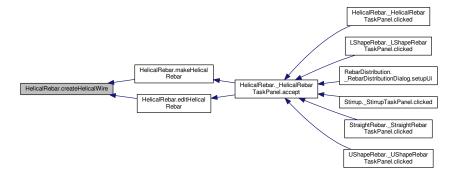
```
HelicalRebar.CommandHelical Rebar Re
```

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

Definition at line 75 of file HelicalRebar.py.

```
00075 def createHelicalWire(FacePRM, s_cover, b_cover, t_cover, pitch, size, direction, helix =
      None):
""" createHelicalWire(FacePRM, SideCover, BottomCover, TopCover, Pitch, Size, Direction, Helix = None):
00076
00077
           It creates a helical wire.""
00078
           import Part
00079
           if not helix:
               helix = FreeCAD.ActiveDocument.addObject("Part::Helix","Helix")
00080
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
if round(direction.x) == 1:
00086
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)
           elif round(direction.x) == -1:
   helix.Placement.Base = FreeCAD.Vector(FacePRM[1][0] + t_cover, FacePRM[1][1], FacePRM[1][2])
00089
00090
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:
               helix.Placement.Base = FreeCAD.Vector(FacePRM[1][0], FacePRM[1][1] + t_cover, FacePRM[1][2]) helix.Placement.Rotation = FreeCAD.Rotation(FreeCAD.Vector(-1, 0, 0), 90)
00096
00097
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:
                \label{eq:helix.Placement.Base = FreeCAD.Vector(FacePRM[1][0], FacePRM[1][1], FacePRM[1][2] + b\_cover) \\ helix.Placement.Rotation = FreeCAD.Rotation(FreeCAD.Vector(0, 0, -1), 0) \\
00102
00103
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.

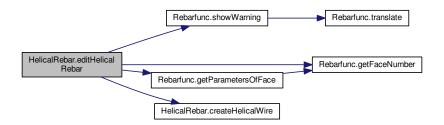
```
00231 def editDialog(vobj):
00232
          FreeCADGui.Control.closeDialog()
          obj = _HelicalRebarTaskPanel(vobj.Object)
00234
          obj.form.sideCover.setText(str(vobj.Object.SideCover))
00235
          obj.form.bottomCover.setText(str(vobj.Object.BottomCover))
00236
          obj.form.diameter.setText(str(vobj.Object.Diameter))
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)

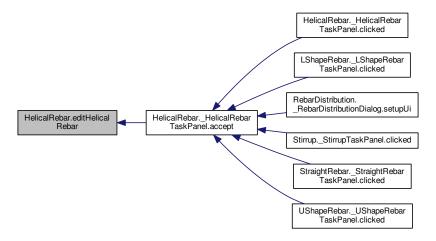
Definition at line 203 of file HelicalRebar.py.

```
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)]
                             # Check if sketch support is empty.
                       if not sketch.Support:
00208
00209
                                       {\tt showWarning} \ ( {\tt ``You} \ {\tt have} \ {\tt checked} \ {\tt remove} \ {\tt external} \ {\tt geometry} \ {\tt of} \ {\tt base} \ {\tt sketchs} \ {\tt when} \ {\tt needed.} \\ {\tt \nTo} \ {\tt \
                   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
FacePRM = getParametersOfFace(structure, facename, False)
00217
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,
                 Rebar.Base)
00222
                            FreeCAD.ActiveDocument.recompute()
00223
                             Rebar.Diameter = diameter
00224
                             Rebar.SideCover = s_cover
00225
                             Rebar.BottomCover = b_cover
00226
                             Rebar.TopCover = t_cover
00227
                             Rebar.Pitch = pitch
                            FreeCAD.ActiveDocument.recompute()
00228
00229
                            return Rebar
00230
```

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

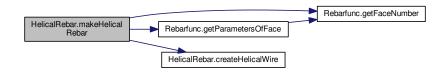
6.2.1.6 def HelicalRebar.makeHelicalRebar (s_cover, b_cover, diameter, t_cover, pitch, structure = None, facename = None)

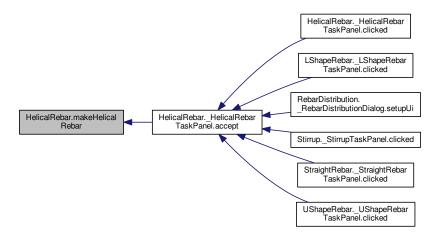
makeHelicalRebar(SideCover, BottomCover, Diameter, TopCover, Pitch, Structure, Facename):
Adds the Helical reinforcement bar to the selected structural object.

Definition at line 165 of file HelicalRebar.py.

```
00165 def makeHelicalRebar(s_cover, b_cover, diameter, t_cover, pitch, structure = None, facename
00166
           """ makeHelicalRebar(SideCover, BottomCover, Diameter, TopCover, Pitch, Structure, Facename): Adds the Helical reinforcement bar to the selected structural object."""
00167
00168
           if not structure and not facename:
              selected_obj = FreeCADGui.Selection.getSelectionEx()[0]
00169
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)
           if not FacePRM:
00175
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)
           #normal = face.Placement.Rotation.inverted().multVec(normal)
00180
00181
           import Arch
00182
           helix = createHelicalWire(FacePRM, s_cover, b_cover, t_cover, pitch, size, normal)
00183
           helix.Support = [(structure, facename)]
           rebar = Arch.makeRebar(structure, helix, diameter, 1, 0)
00184
           rebar.OffsetStart = 0
00185
00186
           rebar.OffsetEnd = 0
00187
           FreeCAD.ActiveDocument.recompute()
           # Adds properties to the rebar object
00188
      rebar.ViewObject.addProperty("App::PropertyString", "RebarShape", "RebarDialog", QT_TRANSLATE_NOOP("App::Property", "Shape of rebar")).RebarShape = "HelicalRebar"
00189
          rebar.ViewObject.setEditorMode("RebarShape", 2)
rebar.addProperty("App::PropertyDistance", "SideCover", "RebarDialog", QT_TRANSLATE_NOOP("App::Property
00190
00191
      ", "Front cover of rebar")).SideCover = s_cover rebar.setEditorMode("SideCover", 2)
00192
00193
           rebar.addProperty("App::PropertyDistance", "Pitch", "RebarDialog", QT_TRANSLATE_NOOP("App::Property", "
      Left Side cover of rebar")).Pitch = pitch
rebar.setEditorMode("Pitch", 2)
00194
           rebar.addProperty("App::PropertyDistance", "BottomCover", "RebarDialog", QT_TRANSLATE_NOOP("
00195
      App::Property", "Bottom cover of rebar")).BottomCover = b_cover
00196
           rebar.setEditorMode("BottomCover", 2)
00197
           "Top cover of rebar")).TopCover = t_cover rebar.setEditorMode("TopCover", 2)
00198
           rebar.Label = "HelicalRebar"
00199
           FreeCAD.ActiveDocument.recompute()
00200
00201
           return rebar
00202
```

Here is the call 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, I_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 editDialog (vobj)
- def CommandLShapeRebar ()

Variables

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

6.3.1 Function Documentation

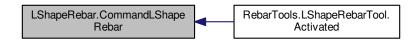
6.3.1.1 def LShapeRebar.CommandLShapeRebar ()

Definition at line 300 of file LShapeRebar.py.

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.

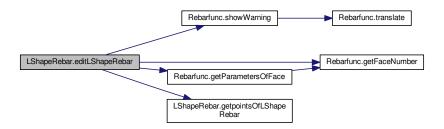
```
00278 def editDialog(vobj):
00279
           FreeCADGui.Control.closeDialog()
00280
           obj = _LShapeRebarTaskPanel(vobj.Object)
00281
           obj.form.frontCover.setText(str(vobj.Object.FrontCover))
00282
           obj.form.l_sideCover.setText(str(vobj.Object.LeftCover))
00283
           obj.form.r_sideCover.setText(str(vobj.Object.RightCover))
obj.form.bottomCover.setText(str(vobj.Object.BottomCover))
00284
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)
           \verb|obj.form.orientation.setCurrentIndex(obj.form.orientation.findText(str(vobj.Object.Orientation))||
00288
00289
           if vobj.Object.AmountCheck:
00290
               obj.form.amount.setValue(vobj.Object.Amount)
00291
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)
           obj.form.spacing.setText(str(vobj.Object.TrueSpacing))
#obj.form.PickSelectedFace.setVisible(False)
00296
00297
00298
           FreeCADGui.Control.showDialog(obj)
00299
```

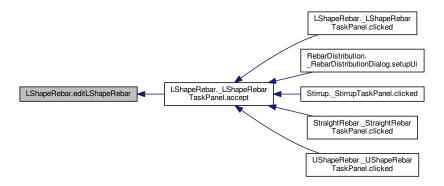
6.3.1.3 def LShapeRebar.editLShapeRebar (Rebar, f_cover, b_cover, l_cover, r_cover, diameter, t_cover, rounding, amount_spacing_check, amount_spacing_value, orientation, structure = None, facename = None)

Definition at line 230 of file LShapeRebar.py.

```
00230 def editLShapeRebar(Rebar, f_cover, b_cover, l_cover, r_cover, diameter, t_cover, rounding,
      amount_spacing_check, amount_spacing_value, orientation, structure = None, facename = None):
           sketch = Rebar.Base
00231
00232
           if structure and facename:
00233
               sketch.Support = [(structure, facename)]
00234
           # Check if sketch support is empty.
00235
           if not sketch.Support:
               {\tt showWarning} \ (\text{``You have checked remove external geometry of base sketchs when needed.} \ \ \texttt{`nTo} \\
00236
       unchecked Edit->Preferences->Arch.")
00237
00238
           # Assigned values
00239
           facename = sketch.Support[0][1][0]
00240
           structure = sketch.Support[0][0]
           face = structure.Shape.Faces[getFaceNumber(facename) - 1]
00241
00242
           #StructurePRM = getTrueParametersOfStructure(structure)
00243
           # Get parameters of the face where sketch of rebar is drawn
00244
           FacePRM = getParametersOfFace(structure, facename)
00245
           # Get points of L-Shape rebar
           points = getpointsOfLShapeRebar(FacePRM, l_cover, r_cover, b_cover, t_cover,
00246
      orientation)
00247
           sketch.movePoint(0, 1, points[0], 0)
00248
           FreeCAD.ActiveDocument.recompute()
00249
           sketch.movePoint(0, 2, points[1], 0)
00250
           FreeCAD.ActiveDocument.recompute()
00251
           sketch.movePoint(1, 1, points[1], 0)
00252
           FreeCAD.ActiveDocument.recompute()
           sketch.movePoint(1, 2, points[2], 0)
FreeCAD.ActiveDocument.recompute()
00253
00254
          Rebar.OffsetStart = f_cover
Rebar.OffsetEnd = f_cover
00255
00256
00257
           \verb|if| amount_spacing_check: \\
00258
               Rebar.Amount = amount_spacing_value
FreeCAD.ActiveDocument.recompute()
00259
00260
               Rebar.AmountCheck = True
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 = 1_cover
00269
           Rebar.RightCover = r_cover
           Rebar.BottomCover = b_cover
00270
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:





6.3.1.4 def LShapeRebar.getpointsOfLShapeRebar (FacePRM, I_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'.
```

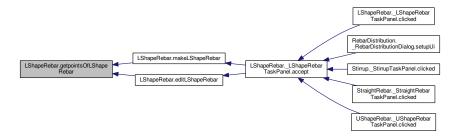
Definition at line 40 of file LShapeRebar.py.

```
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.
             It takes four different orientations input i.e. 'Bottom Left', 'Bottom Right', 'Top Left', 'Top Right'
00043
00044
00045
             if orientation == "Bottom Left":
                  x1 = FacePRM[1][0] - FacePRM[0][0] / 2 + 1_cover
00046
00047
                  y1 = FacePRM[1][1] + FacePRM[0][1] / 2 - t_cover
x2 = FacePRM[1][0] - FacePRM[0][0] / 2 + 1_cover
00048
00049
                  y2 = FacePRM[1][1] - FacePRM[0][1] / 2 + b_cover
                 x3 = FacePRM[1][0] - FacePRM[0][0] / 2 + FacePRM[0][0] - r_cover
y3 = FacePRM[1][1] - FacePRM[0][1] / 2 + b_cover
00050
00051
           elif orientation == "Bottom Right":
    x1 = FacePRM[1][0] - FacePRM[0][0] / 2 + FacePRM[0][0] - r_cover
    v1 = FacePRM[1][1] + FacePRM[0][1] / 2 + FacePRM[0][0]
00052
00053
                 y1 = FacePRM[1][0] - FacePRM[0][1] / 2 - t_cover

y2 = FacePRM[1][0] - FacePRM[0][0] / 2 + FacePRM[0][0] - r_cover
00054
00055
                  y2 = FacePRM[1][1] - FacePRM[0][1] / 2 + b_cover
00056
                  x3 = FacePRM[1][0] - FacePRM[0][0] / 2 + 1_cover
y3 = FacePRM[1][1] - FacePRM[0][1] / 2 + b_cover
00057
00058
            elif orientation == "Top Left":

x1 = FacePRM[1][0] - FacePRM[0][0] / 2 + 1_cover

y1 = FacePRM[1][1] - FacePRM[0][1] / 2 + b_cover
00059
00060
00061
                  x2 = FacePRM[1][0] - FacePRM[0][0] / 2 + 1_cover
00062
00063
                  y2 = FacePRM[1][1] + FacePRM[0][1] / 2 - t_cover
            x3 = FacePRM[1][0] - FacePRM[0][0] / 2 + FacePRM[0][0] - r_cover
y3 = FacePRM[1][1] + FacePRM[0][1] / 2 - t_cover
elif orientation == "Top Right":
00064
00065
00066
              x1 = FacePRM[1][0] - FacePRM[0][0] / 2 + FacePRM[0][0] - r_cover
y1 = FacePRM[1][1] - FacePRM[0][1] / 2 + b_cover
00067
00068
                  x2 = FacePRM[1][0] - FacePRM[0][0] / 2 + FacePRM[0][0] - r_cover
00069
00070
                  y2 = FacePRM[1][1] + FacePRM[0][1] / 2 - t_cover
                 x3 = FacePRM[1][0] - FacePRM[0][0] / 2 + 1_cover
y3 = FacePRM[1][1] + FacePRM[0][1] / 2 - t_cover
00071
00072
             return [FreeCAD.Vector(x1, y1, 0), FreeCAD.Vector(x2, y2, 0),\
FreeCAD.Vector(x3, y3, 0)]
00073
00074
```



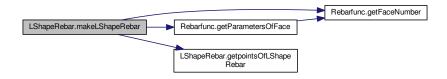
6.3.1.5 def LShapeRebar.makeLShapeRebar (f_cover, b_cover, l_cover, r_cover, diameter, t_cover, rounding, amount_spacing_check, amount_spacing_value, orientation = "Bottom Left", structure = None, facename = None)

makeLShapeRebar(FrontCover, BottomCover, LeftCover, RightCover, Diameter, TopCover, Rounding, AmountSpacingChe 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'.

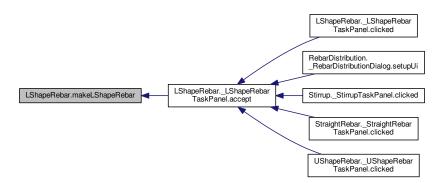
Definition at line 169 of file LShapeRebar.py.

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

```
App::Property", "Front cover of rebar")).FrontCover = f_cover
00206
            rebar.setEditorMode("FrontCover", 2)
             rebar.addProperty("App::PropertyDistance", "LeftCover", "RebarDialog", QT_TRANSLATE_NOOP("App::Property
00207
        ", "Left Side cover of rebar")).LeftCover = 1_cover
00208
            rebar.setEditorMode("LeftCover", 2)
            rebar.addProperty("App::PropertyDistance", "RightCover", "RebarDialog", QT_TRANSLATE_NOOP("
00209
       App::Property", "Right Side cover of rebar")).RightCover = r_cover
00210
            rebar.setEditorMode("RightCover", 2)
00211
            rebar.addProperty("App::PropertyDistance", "BottomCover", "RebarDialog", QT_TRANSLATE_NOOP("
       App::Property", "Bottom cover of rebar")).BottomCover = b_cover
    rebar.setEditorMode("BottomCover", 2)
00212
         rebar.addProperty("App::PropertyBool", "Amou
"Amount radio button is checked")).AmountCheck
                                                             "AmountCheck", "RebarDialog", QT_TRANSLATE_NOOP("App::Property",
00213
00214
           rebar.setEditorMode("AmountCheck", 2)
00215
             rebar.addProperty("App::PropertyDistance", "TopCover", "RebarDialog", QT_TRANSLATE_NOOP("App::Property"
          "Top cover of rebar")).TopCover = t_cover rebar.setEditorMode("TopCover", 2)
00216
       rebar.addProperty("App::PropertyDistance", "TrueSpacing", "RebarDialog", QT_TRANSLATE_NOOP("
App::Property", "Spacing between of rebars")).TrueSpacing = amount_spacing_value
rebar.addProperty("App::PropertyString", "Orientation", "RebarDialog", QT_TRANSLATE_NOOP("App::Property
00217
00218
       ", "Shape of rebar")).Orientation = orientation
            rebar.setEditorMode("Orientation", 2)
rebar.setEditorMode("TrueSpacing", 2)
00219
00220
00221
            if amount_spacing_check:
00222
                 rebar.AmountCheck = True
            else:
00224
                 rebar.AmountCheck = False
            rebar.TrueSpacing = amount_spacing_value
rebar.Label = "LShapeRebar"
00225
00226
00227
            FreeCAD.ActiveDocument.recompute()
00228
            return rebar
00229
```



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)

```
\verb|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 getupleOfCustomSpacing (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
        seg2_area = size - seg1_area - seg3_area - 2 * frontCover
if seg2_area < 0:</pre>
          seg3_area = amount3 * spacing3 - spacing3 / 2
00066
00067
              FreeCAD.Console.PrintError("Sum of length of segment 1 and segment 2 is greater than length of
00068
rebar expands.\n")
00070
          if spacing1 and spacing2 and spacing3 and amount1 and amount2 and amount3:
00071
00072
          if spacing1 and spacing2 and spacing3:
00073
                   amount2 = math.ceil(seg2_area / spacing2)
00075
                   spacing2 = seg2_area / amount2
00076
            elif amount1 and amount2 and amount3:
     spacing2 = math.floor(seg2_area / amount2)
CustomSpacing = str(amount1) + "@" + str(spacing1) + "+" + str(int(amount2)) + "@" + str(spacing2) + "+
" + str(amount3) + "@" + str(spacing3)
00078
00079
          return CustomSpacing
08000
```

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

6.5.1.3 def RebarDistribution.removeRebarDistribution (self)

Definition at line 108 of file RebarDistribution.py.

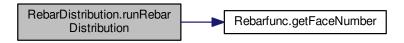
```
00108 def removeRebarDistribution(self):
00109     self.CustomSpacing = ""
00110     self.Rebar.CustomSpacing = ""
00111     FreeCAD.ActiveDocument.recompute()
00112
00113  #runRebarDistribution(App.ActiveDocument.Rebar)
00114
```

6.5.1.4 def RebarDistribution.runRebarDistribution (self)

Definition at line 98 of file RebarDistribution.py.

```
00098 def runRebarDistribution(self):
          frontCover = self.form.frontCover.text()
          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)
          dialog.setupUi()
00104
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.
```

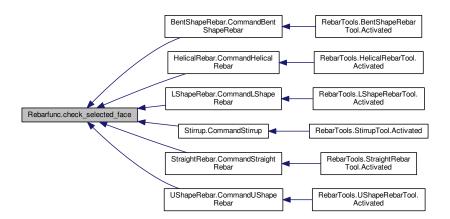
Definition at line 255 of file Rebarfunc.py.

```
00255 def check_selected_face():
          """ check_selected_face(): This function checks whether user have selected
              any face or not."""
00258
          selected_objs = FreeCADGui.Selection.getSelectionEx()
00259
          if not selected_objs:
              showWarning("Select any face of the structural element.")
00260
00261
              selected_obj = None
00262
          else:
00263
             selected_face_names = selected_objs[0].SubElementNames
00264
              if not selected_face_names:
00265
                  selected_obj = None
showWarning("Select any face of the structural element.")
00266
              elif "Face" in selected_face_names[0]:
00267
                 if len(selected_face_names) > 1:
00268
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
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)

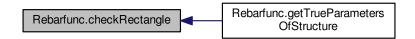
checkRectangle(edges=[]): This function checks whether the given form rectangle
 or not. It will return True when edges form rectangular shape or return False
 when edges not form a rectangular.

Definition at line 46 of file Rebarfunc.py.

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.



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

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:



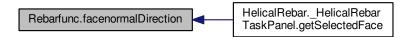
6.6.1.5 def Rebarfunc.facenormalDirection (structure = None, facename = None)

Definition at line 81 of file Rebarfunc.py.

```
00081 def facenormalDirection(structure = None, facename = None):
         if not structure and not facename:
00083
               selected_obj = FreeCADGui.Selection.getSelectionEx()[0]
               structure = selected_obj.Object
facename = selected_obj.SubElementNames[0]
00084
00085
00086
          face = structure.Shape.Faces[getFaceNumber(facename) - 1]
          normal = face.normalAt(0,0)
normal = face.Placement.Rotation.inverted().multVec(normal)
00087
88000
00089
           return normal
00090
00091 # -----
00092 \# Main functions which is use while creating any rebar.
00093 #
00094
```



Here is the caller graph for this function:



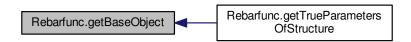
6.6.1.6 def Rebarfunc.getBaseObject (obj)

```
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"):
        return getBaseObject(obj.Base)
00069    else:
00070    return obj
```

Here is the caller graph for this function:

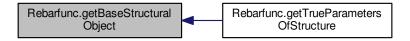


6.6.1.7 def Rebarfunc.getBaseStructuralObject (obj)

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

Definition at line 57 of file Rebarfunc.py.

Here is the caller graph for this function:



6.6.1.8 def Rebarfunc.getEdgesAngle (edge1, edge2)

```
\verb"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
```

Here is the caller graph for this function:

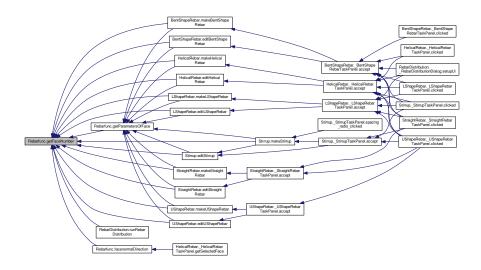


6.6.1.9 def Rebarfunc.getFaceNumber (s)

```
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.

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)]
```

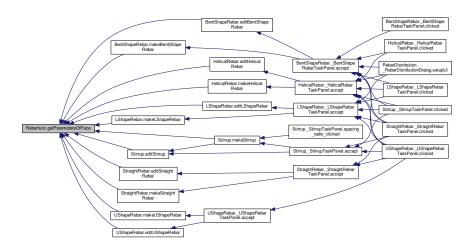
Definition at line 126 of file Rebarfunc.py.

```
00126 def getParametersOfFace(structure, facename, sketch = True):
          """ 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 eq.:
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, CenterOfMassX, CenterOfMassZ)]"""
00141
          face = structure.Shape.Faces[getFaceNumber(facename) - 1]
00142
          center of mass = face.CenterOfMass
          #center_of_mass = center_of_mass.sub(getBaseStructuralObject(structure).Placement.Base)
00143
          center_of_mass = center_of_mass.sub(structure.Placement.Base)
00145
          Edges = []
00146
          facePRM = []
00147
          \ensuremath{\sharp} 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:
                       # Checks whether similar edges is already present in Edges list
00155
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)
              # facePRM holds length of a edges.
00161
              facePRM = [(vec(edge)).Length for edge in Edges]
00162
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]
                   if round(Edges[1].tangentAt(0)[1]) in {1. -1}:
00168
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:
                      y = center_of_mass[2]
00180
              elif round(Edges[0].tangentAt(0)[2]) in {1,-1}:
00181
                  y = center_of_mass[2]
                   if round(Edges[1].tangentAt(0)[0]) in {1, -1}:
00183
00184
                       x = center_of_mass[0]
00185
                   else:
                     x = center_of_mass[1]
00186
                   facePRM.reverse()
00187
              facelength = facePRM[0]
facewidth = facePRM[1]
00188
00189
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)
                   normal = face.Placement.Rotation.inverted().multVec(normal)
00198
                   #print "x: ", boundbox.XLength
#print "y: ", boundbox.YLength
#print "z: ", boundbox.ZLength
00199
00200
00201
00202
                   # Set length and width of user selected face of structural element
00203
                   flag = True
00204
                   # FIXME: Improve below logic.
                  for i in range(len(normal)):
    if round(normal[i]) == 0:
00205
00206
                           if flag and i == 0:
00207
00208
                               x = center_of_mass[i]
00209
                               facelength = boundbox.XLength
00210
                               flag = False
                           elif flag and i == 1:
00211
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:
                                   y = center_of_mass[i]
facewidth = boundbox.ZLength
00219
00220
           #print [(facelength, facewidth), (x, y)]
# Return parameter of the face when rebar is not created from the sketch.
# For eg. non-planar rebars like stirrup etc.
00221
00222
00223
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 caller graph for this function:



6.6.1.11 def Rebarfunc.getSelectedFace (self)

Definition at line 278 of file Rebarfunc.py.

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

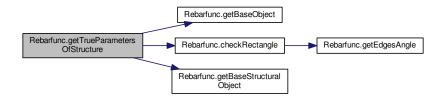
6.6.1.12 def Rebarfunc.getTrueParametersOfStructure (obj)

```
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,
          width and height of the structural element in the form of array like
00098
          [Length, Width, Height]"""
          baseObject = getBaseObject(obj)
00099
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
                          rep_vector = edge.Vertexes[1].Point.sub(edge.Vertexes[0].Point)
00108
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:
                              width = edge.Length
00113
00114
                  else:
00115
                      return None
00116
              else:
00117
                  return None
00118
             height = obj.Height.Value
00119
          else:
00120
              structuralBaseObject = getBaseStructuralObject(obj)
              length = structuralBaseObject.Length.Value
00121
              width = structuralBaseObject.Width.Value
00122
00123
              height = structuralBaseObject.Height.Value
00124
          return [length, width, height]
00125
```

Here is the call graph for this function:



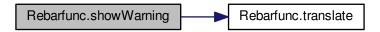
6.6.1.13 def Rebarfunc.showWarning (message)

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

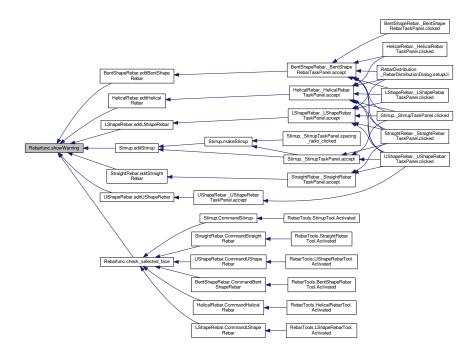
Definition at line 293 of file Rebarfunc.py.

```
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 tanslation handling
```

Here is the call graph for this function:



Here is the caller graph for this function:

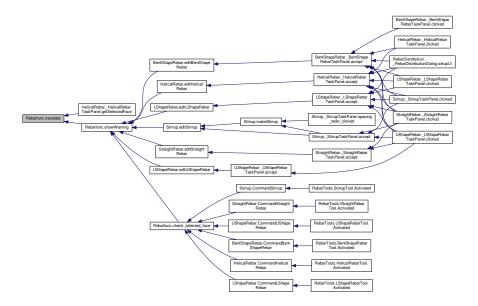


6.6.1.14 def Rebarfunc.translate (context, text, disambig = None)

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 Variable Documentation

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_Ebar_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, I_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, I_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:

```
Stirrup.CommandStirrup Rebarfunc.check_selected_face Rebarfunc.showWarning Rebarfunc.translate
```

Here is the caller graph for this function:



6.8.1.2 def Stirrup.editDialog (vobj)

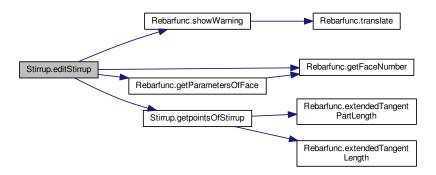
Definition at line 327 of file Stirrup.py.

```
00327 def editDialog(vobj):
          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))
          obj.form.t_sideCover.setText(str(vobj.Object.TopCover))
00333
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)
          obj.form.rounding.setValue(vobj.Object.Rounding)
00338
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)
              obj.form.spacing.setText(str(vobj.Object.TrueSpacing))
00347
          #obj.form.PickSelectedFace.setVisible(False)
00348
          FreeCADGui.Control.showDialog(obj)
00349
```

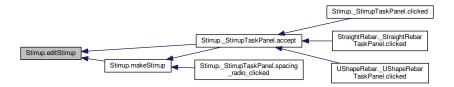
6.8.1.3 def Stirrup.editStirrup (Rebar, I_cover, r_cover, t_cover, b_cover, f_cover, bentAngle, bentFactor, diameter, rounding, amount spacing check, amount spacing value, structure = None, facename = None)

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:
               sketch.Support = [(structure, facename)]
           # Check if sketch support is empty.
00285
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
           # 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)
           # Get parameters of the face where sketch of rebar is drawn
FacePRM = getParametersOfFace(structure, facename, False)
00294
00295
           FaceNormal = face.normalAt(0, 0)
00296
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,
      bentFactor, diameter, rounding, FaceNormal)
Rebar.Base.Points = points
00300
00301
           FreeCAD.ActiveDocument.recompute()
00302
           Rebar.Direction = FaceNormal.negative()
00303
           Rebar.OffsetStart = f_cover
00304
           Rebar.OffsetEnd = f cover
           Rebar.BentAngle = bentAngle
00305
00306
           Rebar.BentFactor = bentFactor
00307
           Rebar.Rounding = rounding
00308
           Rebar.Diameter = diameter
           if amount_spacing_check:
00309
               Rebar.Amount = amount_spacing_value
FreeCAD.ActiveDocument.recompute()
00310
00311
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)
               FreeCAD.ActiveDocument.recompute()
00316
00317
               Rebar.AmountCheck = False
00318
           Rebar.FrontCover = f_cover
           Rebar.LeftCover = 1_cover
00319
00320
           Rebar.RightCover = r_cover
00321
           Rebar.TopCover = t_cover
           Rebar.BottomCover = b_cover
Rebar.TrueSpacing = amount_spacing_value
00322
00323
00324
           FreeCAD.ActiveDocument.recompute()
00325
           return Rebar
00326
```



Here is the caller graph for this function:



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

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

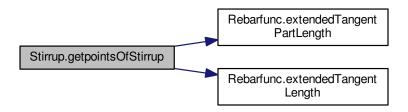
Definition at line 40 of file Stirrup.py.

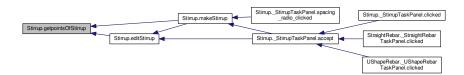
```
00040 def getpointsOfStirrup(FacePRM, l_cover, r_cover, t_cover, b_cover, bentAngle,
      bentFactor, diameter, rounding, facenormal):
""" getpointsOfStirrup(FacePRM, LeftCover, RightCover, TopCover, BottomCover, BentAngle, BentFactor,
00041
        Diameter, Rounding, FaceNormal):
00042
      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

```
x3 = x2 - normal_dis / 4
                x4 = x3 - normal_dis / 4
00062
00063
                x5 = x4 - normal_dis / 4
                x0 = x1 + normal_dis / 4
00064
                y0 = y1 + (tangent\_length + bentFactor * diameter) * math.sin(math.radians(angle))

z0 = z1 - (tangent\_length + bentFactor * diameter) * math.cos(math.radians(angle))
00065
00066
                x6 = x5 - normal_dis / 4
00068
                y6 = y5 + (tangent\_length + bentFactor * diameter) * math.sin(math.radians(90 - angle))
                z6 = z5 - (tangent_length + bentFactor * diameter) * math.cos(math.radians(90 - angle))
00069
00070
           elif round(facenormal[1]) in {1,-1}:
             x1 = FacePRM[1][0] - FacePRM[0][0] / 2 + 1_cover
00071
00072
                v1 = 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 + 1_cover
                z2 = FacePRM[1][2] - FacePRM[0][1] / 2 + b_cover
00075
00076
                x3 = FacePRM[1][0] + FacePRM[0][0] / 2 - r_cover
                z3 = FacePRM[1][2] - FacePRM[0][1] / 2 + b_cover
00077
00078
                x4 = FacePRM[1][0] + FacePRM[0][0] / 2 - r_cover
                z4 = FacePRM[1][2] + FacePRM[0][1] / 2 - t_cover
00079
                x5 = FacePRM[1][0] - FacePRM[0][0] / 2 + 1_cover - tangent_part_length
00080
                z5 = FacePRM[1][2] + FacePRM[0][1] / 2 - t_cover
00081
                side_length = abs(x5 - x4) - tangent_part_length
normal_dis = (diameter * (side_length + tangent_part_length)) / side_length
00082
00083
00084
                v2 = v1 - normal dis / 4
00085
                y3 = y2 - normal_dis / 4
                y4 = y3 - normal_dis / 4
00086
00087
                y5 = y4 - normal_dis / 4
00088
                y0 = y1 + normal_dis / 4
                x0 = x1 + (tangent\_length + bentFactor * diameter) * math.sin(math.radians(angle))

x0 = x1 - (tangent\_length + bentFactor * diameter) * math.cos(math.radians(angle))
00089
00090
                x6 = x5 + (tangent_length + bentFactor * diameter) * math.sin(math.radians(90 - angle))
00091
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}:
             x1 = FacePRM[1][0] - FacePRM[0][0] / 2 + 1_cover
y1 = FacePRM[1][1] + FacePRM[0][1] / 2 - t_cover + tangent_part_length
00095
00096
00097
                z1 = FacePRM[1][2]
                x2 = FacePRM[1][0] - FacePRM[0][0] / 2 + 1_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
                x4 = FacePRM[1][0] + FacePRM[0][0] / 2 - r_cover
00102
                x4 = FacePRM[1][1] + FacePRM[0][0] / 2 - t_cover
y4 = FacePRM[1][1] + FacePRM[0][1] / 2 - t_cover
x5 = FacePRM[1][0] - FacePRM[0][0] / 2 + 1_cover - tangent_part_length
00103
00104
                y5 = FacePRM[1][1] + FacePRM[0][1] / 2 - t_cover
00105
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
                z3 = z2 - normal_dis / 4
00109
00110
                z4 = z3 - normal_dis / 4
                z5 = z4 - normal_dis / 4
00111
00112
                z0 = z1 + normal_dis / 4
00113
                x0 = x1 + (tangent_length + bentFactor * diameter) * math.sin(math.radians(angle))
               y0 = y1 - (tangent_length + bentFactor * diameter) * math.cos(math.radians(angle)) x6 = x5 + (tangent_length + bentFactor * diameter) * math.sin(math.radians(90 - angle)) y6 = y5 - (tangent_length + bentFactor * diameter) * math.cos(math.radians(90 - angle))
00114
00115
00116
                z6 = z5 - normal_dis / 4
00118
           return [FreeCAD.Vector(x0, y0, z0), FreeCAD.Vector(x1, y1, z1),
                     FreeCAD.Vector(x2, y2, z2), FreeCAD.Vector(x3, y3, z3), FreeCAD.Vector(x4, y4, z4), FreeCAD.Vector(x5, y5, z5),
00119
00120
00121
                     FreeCAD. Vector (x6, y6, z6)]
00122
```





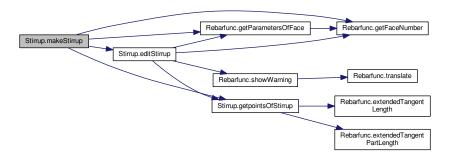
6.8.1.5 def Stirrup.makeStirrup (*I_cover, r_cover, t_cover, f_cover, bentAngle, bentFactor, diameter, rounding, amount spacing check, amount spacing value, structure = None, facename = None)*

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

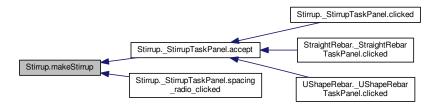
Definition at line 210 of file Stirrup.py.

```
00210
                amount spacing check, amount spacing value, structure = None, facename = None):
           """ makeStirrup(LeftCover, RightCover, TopCover, BottomCover, FrontCover, BentAngle,
BentFactor, Diameter, Rounding, AmountSpacingCheck, AmountSpacingValue, Structure, Facename):
Adds the Stirrup reinforcement bar to the selected structural object."""
00211
00212
00213
           Adds the Stirrup reinforcement bar to the selected structural object."
           if not structure and not facename:
    selected_obj = FreeCADGui.Selection.getSelectionEx()[0]
00214
00215
00216
                structure = selected_obj.Object
                facename = selected_obj.SubElementNames[0]
00217
00218
            face = structure.Shape.Faces[getFaceNumber(facename) - 1]
00219
            #StructurePRM = getTrueParametersOfStructure(structure)
00220
           FacePRM = getParametersOfFace(structure, facename, False)
00221
           FaceNormal = face.normalAt(0,0)
#FaceNormal = face.Placement.Rotation.inverted().multVec(FaceNormal)
00222
            if not FacePRM:
00223
00224
                FreeCAD.Console.PrintError("Cannot identified shape or from which base object sturctural element is
        derived\n")
00225
            # Calculate the coordinate values of Stirrup
00226
           points = getpointsOfStirrup(FacePRM, l_cover, r_cover, t_cover, b_cover, bentAngle,
00227
      bentFactor, diameter, rounding, FaceNormal)
00228
            import Draft
00229
            line = Draft.makeWire(points, closed = False, face = True, support = None)
            import Arch
00230
00231
           line.Support = [(structure, facename)]
00232
           if amount_spacing_check:
00233
                rebar = Arch.makeRebar(structure, line, diameter, amount_spacing_value, f_cover)
00234
00235
                size = (ArchCommands.projectToVector(structure.Shape.copy(), face.normalAt(0, 0))).Length
00236
                rebar = Arch.makeRebar(structure, line, diameter, \
00237
                    int((size - diameter) / amount_spacing_value), f_cover)
           rebar.Direction = FaceNormal.negative()
00238
           rebar.Rounding = rounding
00239
00240
            # Adds properties to the rebar object
           rebar.ViewObject.addProperty("App::PropertyString", "RebarShape", "RebarDialog",\
QT_TRANSLATE_NOOP("App::Property", "Shape of rebar")).RebarShape = "Stirrup"
00241
00242
           rebar.ViewObject.setEditorMode("RebarShape", 2)
rebar.addProperty("App::PropertyDistance", "LeftCover", "RebarDialog",\")
00243
00244
                QT_TRANSLATE_NOOP("App::Property", "Left Side cover of rebar")).LeftCover = 1_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 rebar.setEditorMode("RightCover", 2)
00248
00249
           rebar.addProperty("App::PropertyDistance", "TopCover", "RebarDialog",\
QT_TRANSLATE_NOOP("App::Property", "Top Side cover of rebar")).TopCover = t_cover
00250
00251
            rebar.setEditorMode("TopCover", 2)
00252
00253
           rebar.addProperty("App::PropertyDistance", "BottomCover", "RebarDialog",\
00254
                QT_TRANSLATE_NOOP("App::Property",
                                                          "Bottom Side cover of rebar")).BottomCover = b_cover
           rebar.setEditorMode("BottomCover", 2)
rebar.addProperty("App::PropertyDistance", "FrontCover", "RebarDialog", \
00255
00256
                QT_TRANSLATE_NOOP("App::Property", "Top cover of rebar")).FrontCover = f_cover
00257
00258
            rebar.setEditorMode("FrontCover", 2)
00259
           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)
            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)
           rebar.addProperty("App::PropertyBool", "AmountCheck", "RebarDialog",\
QT_TRANSLATE_NOOP("App::Property", "Amount radio button is checked")).AmountCheck
00265
00266
00267
            rebar.setEditorMode("AmountCheck", 2)
           rebar.addProperty("App::PropertyDistance", "TrueSpacing", "RebarDialog",\
QT_TRANSLATE_NOOP("App::Property", "Spacing between of rebars")).TrueSpacing = amount_spacing_value
00268
00269
            rebar.setEditorMode("TrueSpacing", 2)
00270
00271
           if amount_spacing_check:
00272
                 rebar.AmountCheck = True
00273
            else:
00274
                rebar.AmountCheck = False
           rebar.TrueSpacing = amount_spacing_value
rebar.Label = "Stirrup"
FreeCAD.ActiveDocument.recompute()
00275
00276
00277
00278
            return rebar
00279
```



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

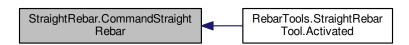
6.9.1.1 def StraightRebar.CommandStraightRebar ()

Definition at line 320 of file StraightRebar.py.

Here is the call graph for this function:



Here is the caller graph for this function:



6.9.1.2 def StraightRebar.editDialog (vobj)

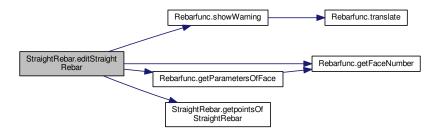
Definition at line 299 of file StraightRebar.py.

```
00299 def editDialog(vobi):
          FreeCADGui.Control.closeDialog()
          obj = _StraightRebarTaskPanel(vobj.Object)
00301
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)
          obj.form.spacing.setText(str(vobj.Object.TrueSpacing))
#obj.form.PickSelectedFace.setVisible(False)
00316
00317
00318
          FreeCADGui.Control.showDialog(obi)
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.
         if not sketch.Support:
00261
00262
              showWarning("You have checked remove external geometry of base sketchs when needed.\nTo
      unchecked Edit->Preferences->Arch.")
00263
00264
          # Assigned values
00265
          facename = sketch.Support[0][1][0]
          structure = sketch.Support[0][0]
00266
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
          points = getpointsOfStraightRebar(FacePRM, rt_cover, lb_cover, coverAlong,
00272
     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
          Rebar.OffsetEnd = f_cover
00278
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
          Rebar.LeftBottomCover = lb_cover
00290
00291
          Rebar.CoverAlong = coverAlong[0]
00292
          Rebar.Cover = coverAlong[1]
00293
          Rebar.TrueSpacing = amount_spacing_value
00294
          Rebar.Diameter = diameter
          Rebar.Orientation = orientation
00295
00296
          FreeCAD.ActiveDocument.recompute()
00297
          return Rebar
00298
```



Here is the caller graph for this function:



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

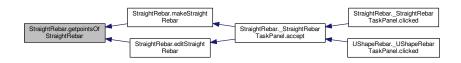
```
getpointsOfStraightRebar(FacePRM, RightTopcover, LeftBottomcover, CoverAlong, Orientation):
Return points of the Straight rebar in the form of array for sketch.

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'
              Case II: When Orientation is 'Vertical':
00046
                   We have two option in CoverAlong i.e. 'Left Side' or 'Right Side'
00047
00048
00049
              if orientation == "Horizontal":
00050
                    if coverAlong[0] == "Bottom Side":
                   x1 = FacePRM[1][0] - FacePRM[0][0] / 2 + lb_cover
y1 = FacePRM[1][1] - FacePRM[0][1] / 2 + coverAlong[1]
x2 = FacePRM[1][0] - FacePRM[0][0] / 2 + FacePRM[0][0] - rt_cover
y2 = FacePRM[1][1] - FacePRM[0][1] / 2 + coverAlong[1]
elif coverAlong[0] == "Top Side":
00051
00052
00053
00054
00055
00056
                         cover = FacePRM[0][1] - coverAlong[1]
                         x1 = FacePRM[1][0] - FacePRM[0][0] / 2 + lb_cover
y1 = FacePRM[1][1] - FacePRM[0][1] / 2 + cover
x2 = FacePRM[1][0] - FacePRM[0][0] / 2 + FacePRM[0][0] - rt_cover
00057
00058
00059
                         y2 = FacePRM[1][1] - FacePRM[0][1] / 2 + cover
00060
00061
              elif orientation == "Vertical":
00062
                    if coverAlong[0] == "Left Side":
```

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



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)

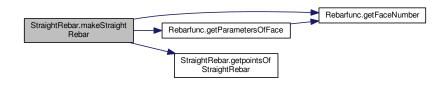
```
Adds the straight reinforcement bar to the selected structural object.
```

```
Case I: When orientation of straight rebar is 'Horizontal':
    makeStraightRebar(FrontCover, CoverAlong, RightCover, LeftCover, Diameter, AmountSpacingCheck, Am
```

Definition at line 185 of file StraightRebar.py.

```
00185 def makeStraightRebar(f_cover, coverAlong, rt_cover, lb_cover, diameter,
      amount_spacing_check, amount_spacing_value, orientation = "Horizontal", structure = None, facename = None):
00186
               Adds the straight reinforcement bar to the selected structural object.
00187
00188
          Case I: When orientation of straight rebar is 'Horizontal':
               makeStraightRebar(FrontCover, CoverAlong, RightCover, LeftCover, Diameter, AmountSpacingCheck,
00189
       AmountSpacingValue, Orientation = "Horizontal",
00190
               Structure, Facename)
               Note: Type of CoverAlong argument is a tuple. Syntax: (<Along>, <Value>). Here we have horizontal
00191
       orientation so we can pass Top Side  \hspace{1cm} \text{and Bottom Side to $<$Along>$ arguments.} 
00192
00193
               For eg. ("Top Side", 20) and ("Bottom Side", 20)
00194
00195
          Case II: When orientation of straight rebar is 'Vertical':
       makeStraightRebar(FrontCover, CoverAlong, TopCover, BottomCover, Diameter, AmountSpacingCheck,
AmountSpacingValue, Orientation = "Horizontal",
00196
00197
               Structure, Facename)
00198
               Note: Type of CoverAlong argument is a tuple. Syntax: (<Along>, <Value>). Here we have vertical
       orientation so we can pass Left Side
00199
               and Right Side to <Along> arguments.
00200
               For eg. ("Left Side", 20) and ("Right Side", 20)
00201
          if not structure and not facename:
00202
00203
               selected_obj = FreeCADGui.Selection.getSelectionEx()[0]
```

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





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 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 ()

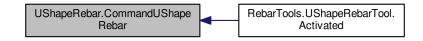
Definition at line 313 of file UShapeRebar.py.

```
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.

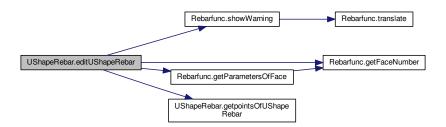
```
00291 def editDialog(vobj):
00292
          FreeCADGui.Control.closeDialog()
00293
          obj = _UShapeRebarTaskPanel(vobj.Object)
          obj.form.frontCover.setText(str(vobj.Object.FrontCover))
00294
00295
          obj.form.r_sideCover.setText(str(vobj.Object.RightCover))
00296
          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
          if structure and facename:
00241
00242
             sketch.Support = [(structure, facename)]
00243
          # Check if sketch support is empty.
         if not sketch.Support:
00244
00245
              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)
          # Get points of U-Shape rebar
00254
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
          {\tt 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()
          Rebar.OffsetStart = f_cover
00268
00269
          Rebar.OffsetEnd = f_cover
00270
          if amount_spacing_check:
    Rebar.Amount = amount_spacing_value
00271
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 = 1_cover
00283
          Rebar.BottomCover = b cover
          Rebar.TopCover = t_cover
Rebar.Rounding = rounding
00284
00285
00286
          Rebar.TrueSpacing = amount_spacing_value
00287
          Rebar.Orientation = orientation
00288
          FreeCAD.ActiveDocument.recompute()
          return Rebar
00289
00290
```

Here is the call graph for this function:



Here is the caller graph for this function:

```
UShapeRebar. UShapeRebar TaskPanel.accept UShapeRebar TaskPanel.clicked
```

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

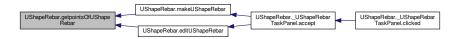
```
getpointsOfUShapeRebar(FacePRM, RightCover, LeftCover, BottomCover, TopCover, Orientation):
Return points of the UShape rebar in the form of array for sketch.
It takes four different orientations input i.e. 'Bottom', 'Top', 'Left', 'Right'.
```

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):
           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'.
00042
00043
00044
00045
           if orientation == "Bottom":
00046
               x1 = FacePRM[1][0] - FacePRM[0][0] / 2 + 1_cover
00047
               y1 = FacePRM[1][1] + FacePRM[0][1] / 2 - t_cover
               x2 = FacePRM[1][0] - FacePRM[0][0] / 2 + 1_cover
00048
00049
               y2 = FacePRM[1][1] - FacePRM[0][1] / 2 + b_cover
               x3 = FacePRM[1][0] - FacePRM[0][0] / 2 + FacePRM[0][0] - r_cover
00050
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":
               x1 = FacePRM[1][0] - FacePRM[0][0] / 2 + 1_cover
00055
               y1 = FacePRM[1][1] - FacePRM[0][1] / 2 + b_cover
00056
               x2 = FacePRM[1][0] - FacePRM[0][0] / 2 + 1_cover
00057
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
               y3 = FacePRM[1][1] + FacePRM[0][1] / 2 - t_cover
00060
          x4 = FacePRM[1][0] - FacePRM[0][0] / 2 + FacePRM[0][0] - r_cover
y4 = FacePRM[1][1] - FacePRM[0][1] / 2 + b_cover
elif orientation == "Left":
00061
00062
00063
              x1 = FacePRM[1][0] - FacePRM[0][0] / 2 + FacePRM[0][0] - r_cover
00064
00065
               y1 = FacePRM[1][1] + FacePRM[0][1] / 2 - t_cover
00066
               x2 = FacePRM[1][0] - FacePRM[0][0] / 2 + 1_cover
               y2 = FacePRM[1][1] + FacePRM[0][1]
00067
                                                       / 2 - t_cover
00068
               x3 = FacePRM[1][0] - FacePRM[0][0] / 2 + 1_cover
               y3 = FacePRM[1][1] - FacePRM[0][1] / 2 + b_cover
00069
               x4 = FacePRM[1][0] - FacePRM[0][0] / 2 + FacePRM[0][0] - r_cover
00070
00071
               y4 = FacePRM[1][1] - FacePRM[0][1] / 2 + b_cover
00072
           elif orientation == "Right":
              x1 = FacePRM[1][0] - FacePRM[0][0] / 2 + 1_cover
y1 = FacePRM[1][1] + FacePRM[0][1] / 2 - t_cover
00073
00074
00075
               x^2 = FacePRM[1][0] - FacePRM[0][0] / 2 + FacePRM[0][0] - r_cover
               y2 = FacePRM[1][1] + FacePRM[0][1] / 2 - t_cover
00077
               x3 = FacePRM[1][0] - FacePRM[0][0] / 2 + FacePRM[0][0] - r_cover
               y3 = FacePRM[1][1] - FacePRM[0][1] / 2 + b_cover
x4 = FacePRM[1][0] - FacePRM[0][0] / 2 + l_cover
00078
00079
               y4 = FacePRM[1][1] - FacePRM[0][1] / 2 + b_cover
08000
           00081
00082
```

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, AmountSpacingChe 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'.

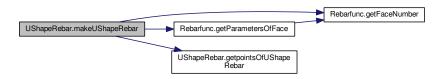
Definition at line 177 of file UShapeRebar.py.

```
00177 def makeUShapeRebar(f_cover, b_cover, r_cover, l_cover, diameter, t_cover, rounding,
       amount_spacing_check, amount_spacing_value, orientation = "Bottom", structure = None, facename = None):

""" makeUShapeRebar(FrontCover, BottomCover, RightCover, LeftCover, Diameter, Topcover, Rounding,
        AmountSpacingCheck, AmountSpacingValue,
00179
           Orientation, Structure, Facename): Adds the U-Shape reinforcement bar to the selected structural
        object.
00180
            It takes four different types of orientations as input i.e 'Bottom', 'Top', 'Right', 'Left'.
00181
00182
            if not structure and not facename:
00183
                selected_obj = FreeCADGui.Selection.getSelectionEx()[0]
00184
                structure = selected_obj.Object
00185
                facename = selected_obj.SubElementNames[0]
00186
            face = structure.Shape.Faces[getFaceNumber(facename) - 1]
00187
            #StructurePRM = getTrueParametersOfStructure(structure)
00188
            FacePRM = getParametersOfFace(structure, facename)
            if not FacePRM:
00189
00190
                FreeCAD.Console.PrintError("Cannot identified shape or from which base object sturctural element is
        derived\n")
00191
00192
            # Get points of U-Shape rebar
00193
            points = getpointsOfUShapeRebar(FacePRM, r_cover, l_cover, b_cover, t_cover,
       orientation)
            import Part
00194
            import Arch
00195
            sketch = FreeCAD.activeDocument().addObject('Sketcher::SketchObject', 'Sketch')
00196
            sketch.MapMode = "FlatFace"
sketch.Support = [(structure, facename)]
00197
00198
00199
            FreeCAD.ActiveDocument.recompute()
00200
            sketch.addGeometry(Part.LineSegment(points[0], points[1]), False)
00201
            {\tt sketch.addGeometry\,(Part.LineSegment\,(points[1],\ points[2]),\ False)}
00202
            import Sketcher
00203
            sketch.addGeometry(Part.LineSegment(points[2], points[3]), False)
00204
            if amount_spacing_check:
00205
                rebar = Arch.makeRebar(structure, sketch, diameter, amount_spacing_value, f_cover)
                FreeCAD.ActiveDocument.recompute()
00206
00207
            else:
                size = (ArchCommands.projectToVector(structure.Shape.copy(), face.normalAt(0, 0))).Length
00208
00209
                rebar = Arch.makeRebar(structure, sketch, diameter, int((size - diameter) / amount_spacing_value),
       f_cover)
00210
            rebar.Rounding = rounding
00211
            # Adds properties to the rebar object
      rebar.ViewObject.addProperty("App::PropertyString", "RebarShape", "RebarDialog", QT_TRANSLATE_NOOP("
App::Property", "Shape of rebar")).RebarShape = "UShapeRebar"
rebar.ViewObject.setEditorMode("RebarShape", 2)
rebar.addProperty("App::PropertyDistance", "FrontCover", "RebarDialog", QT_TRANSLATE_NOOP("
00212
00213
00214
       App::Property",
                          "Front cover of rebar")).FrontCover = f_cover
00215
            rebar.setEditorMode("FrontCover", 2)
00216
            rebar.addProperty("App::PropertyDistance", "RightCover", "RebarDialog", QT_TRANSLATE_NOOP("
       App::Property", "Right Side cover of rebar")).RightCover = r_cover
  rebar.setEditorMode("RightCover", 2)
00217
            rebar.addProperty("App::PropertyDistance", "LeftCover", "RebarDialog", QT_TRANSLATE_NOOP("App::Property
00218
        ", "Left Side cover of rebar")).LeftCover = 1_cover
00219
            rebar.setEditorMode("LeftCover", 2)
            rebar.addProperty("App::PropertyDistance", "BottomCover", "RebarDialog", QT_TRANSLATE_NOOP("
00220
      App::Property", "Bottom cover of rebar")).BottomCover = b_cover
  rebar.setEditorMode("BottomCover", 2)
00221
            rebar.addProperty("App::PropertyBool", "AmountCheck", "RebarDialog", QT_TRANSLATE_NOOP("App::Property",
00222
        "Amount radio button is checked")).AmountCheck
00223
            rebar.setEditorMode("AmountCheck", 2)
         rebar.addProperty("App::PropertyDistance", "TopCover", "RebarDialog", QT_TRANSLATE_NOOP("App::Property"
"Top cover of rebar")).TopCover = t_cover
rebar.setEditorMode("TopCover", 2)
00224
00225
           rebar.addProperty("App::PropertyDistance", "TrueSpacing", "RebarDialog", QT_TRANSLATE_NOOP("::Property", "Spacing between of rebars")).TrueSpacing = amount_spacing_value
00226
       App::Property",
00227
            rebar.setEditorMode("TrueSpacing", 2)
            rebar.addProperty("App::PropertyString", "Orientation", "RebarDialog", QT_TRANSLATE_NOOP("App::Property
00228
          "Shape of rebar")).Orientation = orientation
00229
            rebar.setEditorMode("Orientation", 2)
00230
            if amount_spacing_check:
00231
                rebar.AmountCheck = True
00232
            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:

BentShapeRebar._BentShape RebarTaskPanel

- + SelectedObj
- + FaceName
- + form
- + Rebar
- + __init__()
- + getOrientation()
- + getStandardButtons()
- + clicked()
- + accept()
- + amount_radio_clicked()
- + spacing_radio_clicked()

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

- SelectedObi
- 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.

```
def __init__(self, Rebar = None):
    if not Rebar:
00106
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]
             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
             \verb|self.form.amount_radio.clicked.connect(self.amount_radio_clicked)|\\
00118
             self.form.spacing_radio.clicked.connect(self.spacing_radio_clicked)
             self.form.customSpacing.clicked.connect(lambda: runRebarDistribution(self))
00119
00120
             self.form.removeCustomSpacing.clicked.connect(lambda:
     removeRebarDistribution(self))
00121
             \verb|self.form.PickSelectedFace.clicked.connect(lambda: \verb|getSelectedFace(self)||)|
00122
             \verb|self.form.orientation.currentIndexChanged.connect(self.getOrientation)|\\
00123
             self.form.image.setPixmap(QtGui.QPixmap(os.path.split(os.path.abspath(_
     /icons/BentShapeRebar.svg"))
00124
             self.form.toolButton.setIcon(self.form.toolButton.style().standardIcon(
     QtGui.QStyle.SP_DialogHelpButton))
             (os.path.abspath(\__file\__)) \ [0] \ + \ "/icons/BentShapeRebarDetailed.svg"))
00126
             self.Rebar = Rebar
00127
```

7.1.3 Member Function Documentation

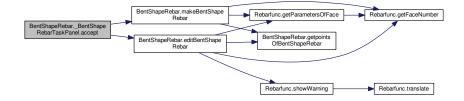
7.1.3.1 def BentShapeRebar_BentShapeRebarTaskPanel.accept (self, signal = None)

Definition at line 146 of file BentShapeRebar.py.

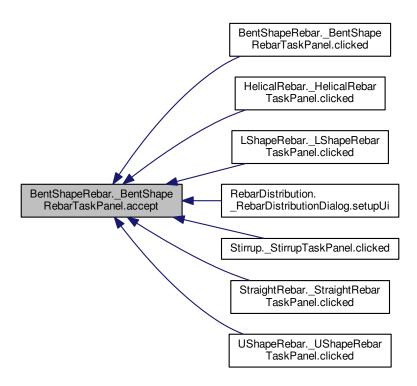
```
00146
          def accept(self, signal = None):
             f_cover = self.form.frontCover.text()
f_cover = FreeCAD.Units.Quantity(f_cover).Value
00147
00148
00149
              b_cover = self.form.bottomCover.text()
              b_cover = FreeCAD.Units.Quantity(b_cover).Value
00150
00151
              l_cover = self.form.l_sideCover.text()
00152
              1_cover = FreeCAD.Units.Quantity(1_cover).Value
00153
              r_cover = self.form.r_sideCover.text()
              r_cover = FreeCAD.Units.Quantity(r_cover).Value
00154
              t_cover = self.form.topCover.text()
00155
              t_cover = FreeCAD.Units.Quantity(t_cover).Value
00156
00157
              bentLength = self.form.bentLength.text()
00158
              bentLength = FreeCAD.Units.Quantity(bentLength).Value
```

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

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)
```

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.

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

7.1.3.5 def BentShapeRebar_BentShapeRebarTaskPanel.getStandardButtons (self)

Definition at line 139 of file BentShapeRebar.py.

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:

HelicalRebar._HelicalRebar TaskPanel + form + Rebar + SelectedObj

+ ___init___()

+ FaceName

- + getStandardButtons()
- + clicked()
- + getSelectedFace()
- + accept()

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:
[1][0])
00115
                                                  normal = facenormalDirection(Rebar.Base.Support[0][0], Rebar.Base.Support[0]
                                        if not round(normal.z) in {1, -1}:
                                                  self.form.topCoverLabel.setText(translate("RebarAddon", "Left Cover"))
 00116
 00117
                                                    self.form.bottomCoverLabel.setText(translate("RebarAddon", "Right Cover"))
00118
                                        self.form.PickSelectedFace.clicked.connect(self.getSelectedFace)
                                        \verb|self.form.image.setPixmap(QtGui.QPixmap(os.path.split(os.path.abspath(\_file\_))[0]| + \verb|self.form.image.setPixmap(QtGui.QPixmap(os.path.split(os.path.abspath(\_file\_))[0]| + \verb|self.form.image.setPixmap(QtGui.QPixmap(os.path.split(os.path.abspath(\_file\_))[0]| + \verb|self.form.image.setPixmap(QtGui.QPixmap(os.path.split(os.path.abspath(\_file\_))[0]| + \verb|self.form.image.setPixmap(QtGui.QPixmap(os.path.split(os.path.abspath(\_file\_))[0]| + \verb|self.form.image.setPixmap(os.path.split(os.path.abspath(\_file\_))[0]| + \verb|self.form.image.setPixmap(os.path.split(os.path.split(os.path.abspath(\_file\_))[0]| + \verb|self.form.image.setPixmap(os.path.split(os.path.split(os.path.abspath(\_file\_))[0]| + \verb|self.form.image.setPixmap(os.path.split(os.path.split(os.path.split(os.path.split(os.path.split(os.path.split(os.path.split(os.path.split(os.path.split(os.path.split(os.path.split(os.path.split(os.path.split(os.path.split(os.path.split(os.path.split(os.path.split(os.path.split(os.path.split(os.path.split(os.path.split(os.path.split(os.path.split(os.path.split(os.path.split(os.path.split(os.path.split(os.path.split(os.path.split(os.path.split(os.path.split(os.path.split(os.path.split(os.path.split(os.path.split(os.path.split(os.path.split(os.path.split(os.path.split(os.path.split(os.path.split(os.path.split(os.path.split(os.path.split(os.path.split(os.path.split(os.path.split(os.path.split(os.path.split(os.path.split(os.path.split(os.path.split(os.path.split(os.path.split(os.path.split(os.path.split(os.path.split(os.path.split(os.path.split(os.path.split(os.path.split(os.path.split(os.path.split(os.path.split(os.path.split(os.path.split(os.path.split(os.path.split(os.path.split(os.path.split(os.path.split(os.path.split(os.path.split(os.path.split(os.path.split(os.path.split(os.path.split(os.path.split(os.path.split(os.path.split(os.path.split(os.path.split(os.path.split(os.path.split(os.path.split(os.path.split(os.path.split(os.path.split(os.path.split(os.path.split(os.path.split(os.path.split(os.path.split(os.path.split(os.path.split(os.p
00119
                /icons/HelicalRebar.svg"))
00120
                                       self.form.toolButton.clicked.connect(lambda: showPopUpImageDialog(os.path.split
                (os.path.abspath(__file__))[0] + "/icons/HelicalRebarDetailed.svg"))
 00121
                                        self.form.toolButton.setIcon(self.form.toolButton.style().standardIcon(
               QtGui.QStyle.SP_DialogHelpButton))
                             self.Rebar = Rebar
self.SelectedObj = None
 00122
00123
                                      self.FaceName = None
 00125
```

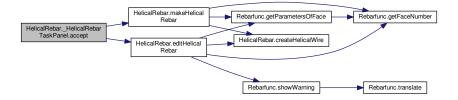
7.2.3 Member Function Documentation

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

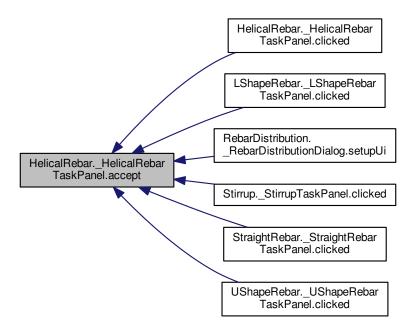
Definition at line 144 of file HelicalRebar.py.

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

Here is the call graph for this function:



Here is the caller graph for this function:

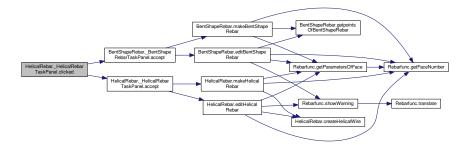


7.2.3.2 def HelicalRebar._HelicalRebarTaskPanel.clicked (self, button)

Definition at line 129 of file HelicalRebar.py.

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

Here is the call graph for this function:

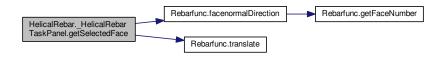


7.2.3.3 def HelicalRebar._HelicalRebarTaskPanel.getSelectedFace (self)

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
                     self.form.topCoverLabel.setText(translate("RebarAddon", "Top Cover"))
self.form.bottomCoverLabel.setText(translate("RebarAddon", "Bottom Cover"))
00140
00141
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.

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:

LShapeRebar._LShapeRebar TaskPanel

- + CustomSpacing
- + SelectedObj
- + FaceName
- + form
- + Rebar
- + __init__()
- + getOrientation()
- + getStandardButtons()
- + clicked()
- + accept()
- + amount_radio_clicked()
- + spacing_radio_clicked()

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
                               if not Rebar:
08000
                                        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]
                                         self.SelectedObj = Rebar.Base.Support[0][0]
00085
                              self.form = FreeCADGui.PySideUic.loadUi(os.path.splitext(__file__)[0] + ".ui")
00087
                              self.form.setWindowTitle(QtGui.QApplication.translate("RebarAddon", "L-Shape Rebar", None))
88000
                              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:
            removeRebarDistribution(self))
00093
                            self.form.PickSelectedFace.clicked.connect(lambda: getSelectedFace(self))
00094
                              \verb|self.form.orientation.currentIndexChanged.connect(self.getOrientation)|\\
00095
                                \tt self.form.image.setPixmap(QtGui.QPixmap(os.path.split(os.path.abspath(\_file\_))[0] + "linearized form.image.setPixmap(QtGui.QPixmap(os.path.split(os.path.abspath(\_file\_))[0] + "linearized form.image.setPixmap(QtGui.QPixmap(os.path.split(os.path.abspath(\_file\_))[0] + "linearized form.image.setPixmap(QtGui.QPixmap(os.path.split(os.path.abspath(\_file\_))[0] + "linearized form.image.setPixmap(os.path.split(os.path.abspath(\_file\_))[0] + "linearized form.split(os.path.abspath(\_file\_))[0] + "linearized form.split(os.path.ab
             /icons/LShapeRebarBR.svg"))
00096
                               self.form.toolButton.setIcon(self.form.toolButton.style().standardIcon(
             QtGui.QStyle.SP_DialogHelpButton))
00097
                                self.form.toolButton.clicked.connect(lambda: showPopUpImageDialog(os.path.split
              (os.path.abspath(__file__))[0] + "/icons/LShapeRebarDetailed.svg"))
00098
                                self.Rebar = Rebar
00099
```

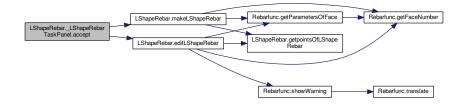
7.3.3 Member Function Documentation

7.3.3.1 def LShapeRebar_LShapeRebarTaskPanel.accept (self, signal = None)

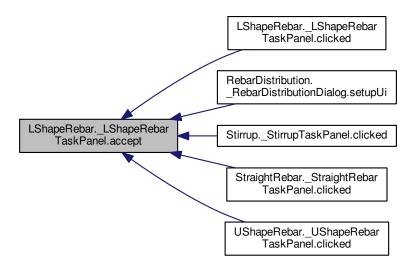
Definition at line 118 of file LShapeRebar.py.

```
def accept(self, signal = None):
               f_cover = self.form.frontCover.text()
f_cover = FreeCAD.Units.Quantity(f_cover).Value
00119
00120
00121
               b_cover = self.form.bottomCover.text()
               b_cover = FreeCAD.Units.Quantity(b_cover).Value
00122
               1_cover = self.form.l_sideCover.text()
00123
               1_cover = FreeCAD.Units.Quantity(1_cover).Value
00124
00125
               r_cover = self.form.r_sideCover.text()
00126
               r_cover = FreeCAD.Units.Quantity(r_cover).Value
00127
               t_cover = self.form.topCover.text()
               t_cover = FreeCAD.Units.Quantity(t_cover).Value
00128
00129
               diameter = self.form.diameter.text()
               diameter = FreeCAD.Units.Quantity(diameter).Value
00130
00131
               rounding = self.form.rounding.value()
00132
               orientation = self.form.orientation.currentText()
               amount_check = self.form.amount_radio.isChecked()
spacing_check = self.form.spacing_radio.isChecked()
00133
00134
               if not self.Rebar:
    if amount_check:
00135
00136
00137
                         amount = self.form.amount.value()
00138
                         rebar = makeLShapeRebar(f_cover, b_cover, l_cover, r_cover, diameter,
      t_cover, rounding, True, amount, orientation, self.SelectedObj, self.
      FaceName)
00139
                    elif spacing_check:
                         spacing = self.form.spacing.text()
spacing = FreeCAD.Units.Quantity(spacing).Value
00140
00141
00142
                         rebar = makeLShapeRebar(f_cover, b_cover, l_cover, r_cover, diameter,
      t_cover, rounding, False, spacing, orientation, self.SelectedObj, self.
      FaceName)
00143
               else:
00144
                    if amount check:
                        amount = self.form.amount.value()
rebar = editLShapeRebar(self.Rebar, f_cover, b_cover, l_cover, r_cover,
00145
00146
       diameter, t_cover, rounding, True, amount, orientation, self.SelectedObj, self.
00147
                    elif spacing_check:
                        spacing = self.form.spacing.text()
spacing = FreeCAD.Units.Quantity(spacing).Value
00148
00149
00150
                         rebar = editLShapeRebar(self.Rebar, f_cover, b_cover, l_cover, r_cover,
       diameter, t_cover, rounding, False, spacing, orientation, self.SelectedObj, self.
      FaceName)
00151
               if self.CustomSpacing:
                    rebar.CustomSpacing = self.CustomSpacing
00152
00153
                    FreeCAD.ActiveDocument.recompute()
00154
               self.Rebar = rebar
00155
               if signal == int(QtGui.QDialogButtonBox.Apply):
00156
00157
               else:
00158
                    FreeCADGui.Control.closeDialog(self)
00159
```

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.

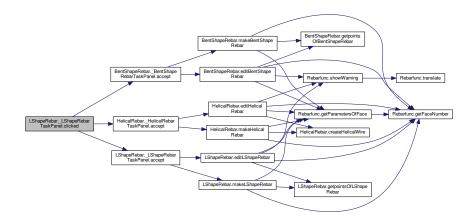
```
00160 def amount_radio_clicked(self):
00161 self.form.spacing.setEnabled(False)
00162 self.form.amount.setEnabled(True)
00163
```

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)
```

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

7.3.3.5 def LShapeRebar_LShapeRebarTaskPanel.getStandardButtons (self)

Definition at line 111 of file LShapeRebar.py.

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:

RebarDistribution. _RebarDistributionDialog + FrontCover + ExpandingLength + form + CustomSpacing + __init__() + accept() + setupUi()

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.

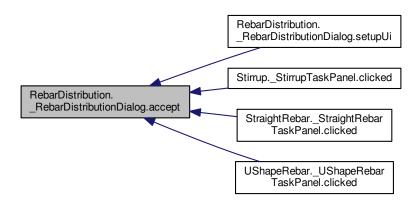
7.4.3 Member Function Documentation

7.4.3.1 def RebarDistribution._RebarDistributionDialog.accept (self)

Definition at line 46 of file RebarDistribution.py.

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

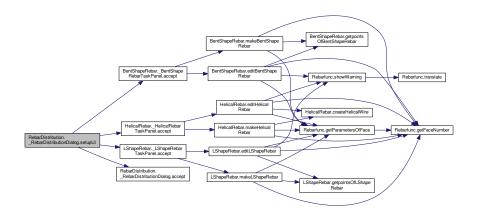
Here is the caller graph for this function:



7.4.3.2 def RebarDistribution._RebarDistributionDialog.setupUi (self)

Definition at line 58 of file RebarDistribution.py.

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:

Stirrup. StirrupTaskPanel

- + CustomSpacing
- + SelectedObj
- + FaceName
- + form
- + Rebar
- + __init__() + getStandardButtons()
- + clicked()
- + accept()
- + amount_radio_clicked()
- + spacing_radio_clicked()

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:
                   selected_obj = FreeCADGui.Selection.getSelectionEx()[0]
00128
                   self.SelectedObj = selected_obj.Object
00129
                   self.FaceName = selected_obj.SubElementNames[0]
00130
              else:
                  self.FaceName = Rebar.Base.Support[0][1][0]
00131
                   self.SelectedObj = Rebar.Base.Support[0][0]
00132
              self.form = FreeCADGui.PySideUic.loadUi(os.path.splitext(__file__)[0] + ".ui")
00133
              self.form.setWindowTitle(QtGui.QApplication.translate("RebarAddon", "Stirrup Rebar", None)) self.form.bentAngle.addItems(["135", "90"])
00135
00136
               self.form.amount_radio.clicked.connect(self.amount_radio_clicked)
              self.form.spacing_radio.clicked.connect(self.spacing_radio_clicked)
self.form.image.setPixmap(QtGui.QPixmap(os.path.split(os.path.abspath(__file__))[0]+"
00137
00138
      /icons/Stirrup.svg"))
00139
              self.form.customSpacing.clicked.connect(lambda: runRebarDistribution(self))
00140
               self.form.removeCustomSpacing.clicked.connect(lambda:
      removeRebarDistribution(self))
00141
              self.form.PickSelectedFace.clicked.connect(lambda: getSelectedFace(self))
00142
               self.form.toolButton.setIcon(self.form.toolButton.style().standardIcon(
      OtGui.OStyle.SP DialogHelpButton))
00143
               self.form.toolButton.clicked.connect(lambda: showPopUpImageDialog(os.path.split
      (os.path.abspath(__file__))[0] + "/icons/StirrupDetailed.svg"))
00144
              self.Rebar = Rebar
00145
```

7.5.3 Member Function Documentation

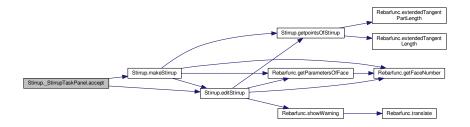
7.5.3.1 def Stirrup._StirrupTaskPanel.accept (self, signal = None)

Definition at line 153 of file Stirrup.py.

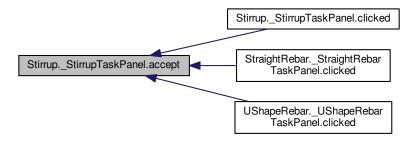
```
00153
         def accept(self, signal = None):
00154
             l_cover = self.form.l_sideCover.text()
             1_cover = FreeCAD.Units.Quantity(1_cover).Value
00155
00156
             r_cover = self.form.r_sideCover.text()
00157
             r_cover = FreeCAD.Units.Quantity(r_cover).Value
00158
             t_cover = self.form.t_sideCover.text()
             t_cover = FreeCAD.Units.Quantity(t_cover).Value
00159
             b_cover = self.form.b_sideCover.text()
00160
             b_cover = FreeCAD.Units.Quantity(b_cover).Value
00161
00162
             f_cover = self.form.frontCover.text()
00163
             f_cover = FreeCAD.Units.Quantity(f_cover).Value
```

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

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.

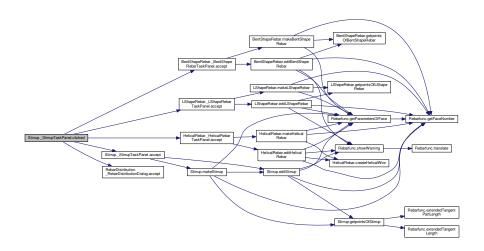
```
00200 def amount_radio_clicked(self):
00201 self.form.spacing.setEnabled(False)
00202 self.form.amount.setEnabled(True)
00203
```

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)
```

Here is the call graph for this function:



7.5.3.4 def Stirrup._StirrupTaskPanel.getStandardButtons (self)

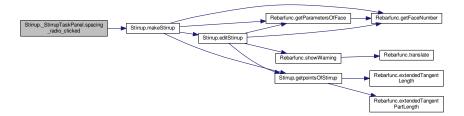
Definition at line 146 of file Stirrup.py.

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:

StraightRebar__StraightRebar TaskPanel + CustomSpacing + SelectedObj + FaceName + form + Rebar + __init__() + changeOrientation() + changeCoverAlong() + getStandardButtons() + clicked() + accept() + amount_radio_clicked() + spacing_radio_clicked()

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
                                if not Rebar:
00079
                                         selected_obj = FreeCADGui.Selection.getSelectionEx()[0]
08000
                                          self.SelectedObj = selected_obj.Object
00081
                                         self.FaceName = selected_obj.SubElementNames[0]
00082
00083
                                         self.FaceName = Rebar.Base.Support[0][1][0]
00084
                                         self.SelectedObj = Rebar.Base.Support[0][0]
                                self.form = FreeCADGui.PySideUic.loadUi(os.path.splitext(__file__)[0] + ".ui")
                                self.form.setWindowTitle(QtGui.QApplication.translate("RebarAddon", "Straight Rebar", None))
self.form.orientation.addItems(["Horizontal", "Vertical"])
self.form.coverAlong.addItems(["Bottom Side", "Top Side"])
00086
00087
00088
                                self.form.amount_radio.clicked.connect(self.amount_radio_clicked)
00089
                                self.form.spacing_radio.clicked.connect(self.spacing_radio_clicked)
00090
                                self.form.customSpacing.clicked.connect(lambda: runRebarDistribution(self))
                                 self.form.removeCustomSpacing.clicked.connect(lambda:
             removeRebarDistribution(self))
00093
                                self.form.PickSelectedFace.setCheckable(True)
00094
                                self.form.PickSelectedFace.toggle()
00095
                                self.form.PickSelectedFace.clicked.connect(lambda: getSelectedFace(self))
                                self.form.image.setPixmap(QtGui.QPixmap(os.path.split(os.path.abspath(__file__))[0] + "
00096
             /icons/StraightRebarH.svg"))
00097
                              self.form.orientation.currentIndexChanged.connect(self.changeOrientation)
00098
                                self.form.coverAlong.currentIndexChanged.connect(self.changeCoverAlong)
00099
                                \verb|self.form.toolButton.setIcon(self.form.toolButton.style().standardIcon(self.form.toolButton.style().standardIcon(self.form.toolButton.style().standardIcon(self.form.toolButton.style().standardIcon(self.form.toolButton.style().standardIcon(self.form.toolButton.style().standardIcon(self.form.toolButton.style().standardIcon(self.form.toolButton.style().standardIcon(self.form.toolButton.style().standardIcon(self.form.toolButton.style().standardIcon(self.form.toolButton.style().standardIcon(self.form.toolButton.style().standardIcon(self.form.toolButton.style().standardIcon(self.form.toolButton.style().standardIcon(self.form.toolButton.style().standardIcon(self.form.toolButton.style().standardIcon(self.form.toolButton.style().standardIcon(self.form.toolButton.style().standardIcon(self.form.toolButton.style().standardIcon(self.form.toolButton.style().standardIcon(self.form.toolButton.style().standardIcon(self.form.toolButton.style().standardIcon(self.form.toolButton.style().standardIcon(self.form.toolButton.style().standardIcon(self.form.toolButton.style().standardIcon(self.form.toolButton.style().standardIcon(self.form.toolButton.style().standardIcon(self.form.toolButton.style().standardIcon(self.form.toolButton.style().standardIcon(self.form.toolButton.style().standardIcon(self.form.toolButton.style().standardIcon(self.form.toolButton.style().standardIcon(self.form.toolButton.style().standardIcon(self.form.toolButton.style().standardIcon(self.form.toolButton.style().standardIcon(self.form.toolButton.style().standardIcon(self.form.toolButton.style().standardIcon(self.form.style().standardIcon(self.form.toolButton.style().standardIcon(self.form.toolButton.style().standardIcon(self.form.toolButton.style().standardIcon(self.form.toolButton.style().standardIcon(self.form.toolButton.style().standardIcon(self.form.toolButton.style().standardIcon(self.form.toolButton.style().standardIcon(self.form.toolButton.style().standardIcon(self.form.toolButton.style().standardIcon(self.form.toolButton.style().standardIcon(self.for
             OtGui.OStvle.SP DialogHelpButton))
00100
                                self.form.toolButton.clicked.connect(lambda: showPopUpImageDialog(os.path.split
              (os.path.abspath(__file__))[0] + "/icons/StraightRebarDetailed.svg"))
00101
                                self.Rebar = Rebar
00102
```

7.6.3 Member Function Documentation

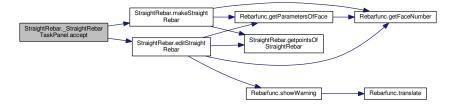
7.6.3.1 def StraightRebar._StraightRebarTaskPanel.accept (self, signal = None)

Definition at line 136 of file StraightRebar.py.

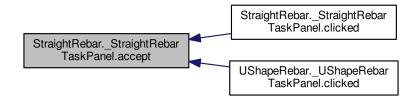
```
00136
          def accept(self, signal = None):
00137
              f_cover = self.form.frontCover.text()
00138
              f_cover = FreeCAD.Units.Quantity(f_cover).Value
00139
              cover = self.form.bottomCover.text()
              cover = FreeCAD.Units.Quantity(cover).Value
00140
00141
              lb_cover = self.form.l_sideCover.text()
              lb_cover = FreeCAD.Units.Quantity(lb_cover).Value
00142
              rt_cover = self.form.r_sideCover.text()
00143
              rt_cover = FreeCAD.Units.Quantity(rt_cover).Value
00144
00145
              orientation = self.form.orientation.currentText()
00146
              coverAlong = self.form.coverAlong.currentText()
              diameter = self.form.diameter.text()
diameter = FreeCAD.Units.Quantity(diameter).Value
00147
00148
              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()
                      rebar = makeStraightRebar(f_cover, (coverAlong, cover), rt_cover, lb_cover
00154
      , diameter, 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 = makeStraightRebar(f_cover, (coverAlong, cover), rt_cover, lb_cover
       diameter, False, spacing, orientation, self.SelectedObj, self.
      FaceName)
00159
              else:
00160
                  if amount_check:
00161
                      amount = self.form.amount.value()
00162
                       rebar = editStraightRebar(self.Rebar, f_cover, (coverAlong, cover),
      rt_cover, lb_cover, diameter, True, amount, orientation, self.SelectedObj, self.
      FaceName)
00163
                  elif spacing_check:
                      spacing = self.form.spacing.text()
spacing = FreeCAD.Units.Quantity(spacing).Value
00164
00165
00166
                       rebar = editStraightRebar(self.Rebar, f_cover, (coverAlong, cover),
      rt_cover, lb_cover, diameter, False, spacing, orientation, self.SelectedObj, self.
      FaceName)
00167
              if self.CustomSpacing:
                  rebar.CustomSpacing = self.CustomSpacing
00168
                  FreeCAD.ActiveDocument.recompute()
00170
              self.Rebar = rebar
              if signal == int(QtGui.QDialogButtonBox.Apply):
00171
00172
00173
              else:
00174
                  FreeCADGui.Control.closeDialog(self)
00175
```

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):
               coverAlong = self.form.coverAlong.currentText()
if coverAlong == "Bottom Side":
00119
00120
00121
                   self.form.bottomCoverLabel.setText("Bottom Cover")
               elif coverAlong == "Top Side":
00122
00123
                   self.form.bottomCoverLabel.setText("Top Cover")
00124
               elif coverAlong == "Right Side":
                   self.form.bottomCoverLabel.setText("Right Cover")
00125
00126
               else:
                   self.form.bottomCoverLabel.setText("Left Cover")
00127
00128
```

7.6.3.4 def StraightRebar._StraightRebarTaskPanel.changeOrientation (self)

Definition at line 103 of file StraightRebar.py.

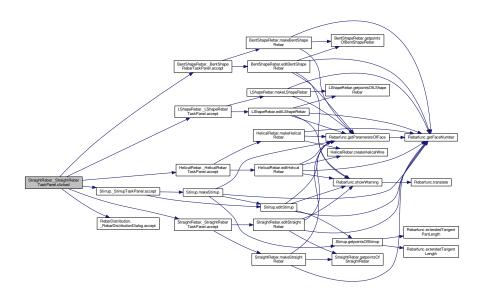
```
00103
           def changeOrientation(self):
               orientation = self.form.orientation.currentText()
if orientation == "Horizontal":
00104
00105
00106
                    self.form.image.setPixmap(QtGui.QPixmap(os.path.split(os.path.abspath(__file__))[0] + "
      /icons/StraightRebarH.svg"))
00107
                    self.form.r_sideCoverLabel.setText("Right Side Cover")
                    self.form.l_sideCoverLabel.setText("Left Side Cover")
00108
00109
                    self.form.coverAlong.clear()
                    self.form.coverAlong.addItems(["Bottom Side", "Top Side"])
00110
00111
               else:
00112
                    self.form.image.setPixmap(QtGui.QPixmap(os.path.split(os.path.abspath(__file__))[0] + "
      /icons/StraightRebarV.svg"))
                    self.form.r_sideCoverLabel.setText("Top Side Cover")
self.form.l_sideCoverLabel.setText("Bottom Side Cover")
00113
00114
                    self.form.coverAlong.clear()
00115
                    self.form.coverAlong.addItems(["Right Side", "Left Side"])
00116
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)
00135
```

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(
QtGui.QDialogButtonBox.Cancel)
00131
```

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:

UShapeRebar._UShapeRebar TaskPanel

- + CustomSpacing
- + SelectedÓbj
- + FaceName
- + form
- + Rebar
- + __init__()
- + getOrientation()
- + getStandardButtons()
- + clicked()
- + accept()
- + amount_radio_clicked()
- + spacing_radio_clicked()

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.

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

7.7.3 Member Function Documentation

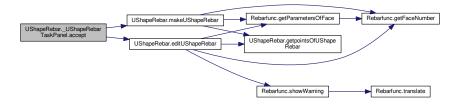
7.7.3.1 def UShapeRebar_UShapeRebarTaskPanel.accept (self, signal = None)

Definition at line 126 of file UShapeRebar.py.

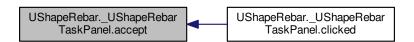
```
00126
          def accept(self, signal = None):
00127
              f_cover = self.form.frontCover.text()
              f_cover = FreeCAD.Units.Quantity(f_cover).Value
00129
              b_cover = self.form.bottomCover.text()
             b_cover = FreeCAD.Units.Quantity(b_cover).Value
00130
00131
             r_cover = self.form.r_sideCover.text()
             r_cover = FreeCAD.Units.Quantity(r_cover).Value
00132
00133
              l_cover = self.form.l_sideCover.text()
             1_cover = FreeCAD.Units.Quantity(l_cover).Value
00134
00135
              t_cover = self.form.topCover.text()
              t_cover = FreeCAD.Units.Quantity(t_cover).Value
00136
00137
             diameter = self.form.diameter.text()
              diameter = FreeCAD.Units.Quantity(diameter).Value
00138
00139
             rounding = self.form.rounding.value()
             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:
                 if amount check:
00144
00145
                      amount = self.form.amount.value()
                      rebar = makeUShapeRebar(f_cover, b_cover, r_cover, l_cover, diameter,
00146
      t_cover, rounding, True, amount, orientation, self.SelectedObj, self.
00147
                  elif spacing_check:
                      spacing = self.form.spacing.text()
00148
                      spacing = FreeCAD.Units.Quantity(spacing).Value
00149
                      rebar = makeUShapeRebar(f_cover, b_cover, r_cover, l_cover, diameter,
00150
      t_cover, rounding, False, spacing, orientation, self.SelectedObj, self.
      FaceName)
00151
              else:
00152
                  if amount check:
                      amount = self.form.amount.value()
00153
                      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:
                             spacing = self.form.spacing.text()
spacing = FreeCAD.Units.Quantity(spacing).Value
00156
00157
         rebar = editUShapeRebar(self.Rebar, f_cover, b_cover, r_cover, l_cover, diameter, t_cover, rounding, False, spacing, orientation, self.SelectedObj, self.
00158
        FaceName)
00159
00160
                       rebar.CustomSpacing = self.CustomSpacing
00161
                       FreeCAD.ActiveDocument.recompute()
                  self.Rebar = rebar
if signal == int(QtGui.QDialogButtonBox.Apply):
00162
00163
00164
00165
00166
                       FreeCADGui.Control.closeDialog(self)
00167
```

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.

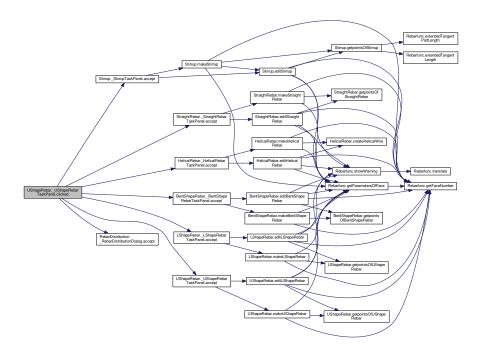
```
00168 def amount_radio_clicked(self):
00169 self.form.spacing.setEnabled(False)
00170 self.form.amount.setEnabled(True)
00171
```

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)
```

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] + "
      /icons/UShapeRebarBottom.svg"))
    elif orientation == "Top":
00112
00113
                   self.form.image.setPixmap(QtGui.QPixmap(os.path.split(os.path.abspath(__file__))[0] + "
      /icons/UShapeRebarTop.svg"))
    elif orientation == "Right":
00114
00115
                   self.form.image.setPixmap(QtGui.QPixmap(os.path.split(os.path.abspath(__file__))[0] + "
       /icons/UShapeRebarRight.svg"))
00116
               else:
                   self.form.image.setPixmap(QtGui.QPixmap(os.path.split(os.path.abspath(__file__))[0] + "
00117
       /icons/UShapeRebarLeft.svg"))
00118
```

7.7.3.5 def UShapeRebar._UShapeRebarTaskPanel.getStandardButtons (self)

Definition at line 119 of file UShapeRebar.py.

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:

RebarTools.BentShapeRebarTool

- + GetResources()
- + IsActive()
- + Activated()

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:

```
RebarTools BentShapeRebar Tool Activated

BentShapeRebar CommandBent ShapeRebar Tool Activated

Rebarfunc.check_selected_face

Rebarfunc.showWarning
```

7.8.2.2 def RebarTools.BentShapeRebarTool.GetResources (self)

Definition at line 106 of file RebarTools.py.

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

96 Class Documentation

7.9 RebarTools.HelicalRebarTool Class Reference

Collaboration diagram for RebarTools.HelicalRebarTool:

+ GetResources() + IsActive() + Activated()

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()
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:

```
RebarTools.HelicalRebarTool.
Activated Rebartunc.showWarning Rebarfunc.showWarning Rebarfunc.showWarning Rebarfunc.showWarning
```

7.9.2.2 def RebarTools.HelicalRebarTool.GetResources (self)

Definition at line 124 of file RebarTools.py.

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:

RebarTools.LShapeRebarTool + GetResources() + IsActive() + Activated()

Public Member Functions

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

98 Class Documentation

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 CommandUShaepRebar() function
00084 LShapeRebar.CommandLShapeRebar()
```

Here is the call graph for this function:

```
RebarTools.LShapeRebarTool.
Activated

LShapeRebar CommandLShape
Rebarfunc.check_selected_face

Rebarfunc.showWarning

Rebarfunc.translate
```

7.10.2.2 def RebarTools.LShapeRebarTool.GetResources (self)

Definition at line 70 of file RebarTools.py.

7.10.2.3 def RebarTools.LShapeRebarTool.IsActive (self)

Definition at line 75 of file RebarTools.py.

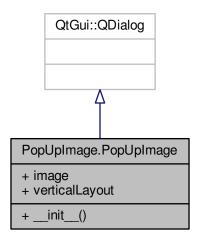
```
00075 def IsActive(self):
00076 if FreeCADGui.ActiveDocument:
00077 return True
00078 else:
00079 return False
```

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

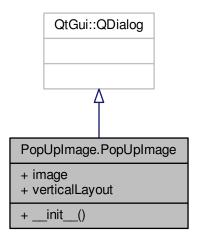
RebarTools.py

7.11 PopUpImage.PopUpImage Class Reference

Inheritance diagram for PopUpImage.PopUpImage:



Collaboration diagram for PopUpImage.PopUpImage:



Public Member Functions

def __init__ (self, img)

100 Class Documentation

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.

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:

+ GetResources() + IsActive() + Activated()

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:

```
RebarTools.StirrupTool.Activated Stirrup.CommandStirrup Rebarfunc.check_selected_face Rebarfunc.showWarning Rebarfunc.translate
```

7.12.2.2 def RebarTools.StirrupTool.GetResources (self)

Definition at line 88 of file RebarTools.py.

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
```

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

RebarTools.py

102 Class Documentation

7.13 RebarTools.StraightRebarTool Class Reference

Collaboration diagram for RebarTools.StraightRebarTool:

RebarTools.StraightRebarTool

+ GetResources()
+ IsActive()
+ Activated()

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.

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
```

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

RebarTools.py

7.14 RebarTools.UShapeRebarTool Class Reference

Collaboration diagram for RebarTools.UShapeRebarTool:

RebarTools.UShapeRebarTool + GetResources() + IsActive() + Activated()

Public Member Functions

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

104 Class Documentation

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 CommandUShaepRebar() function
00066        UShapeRebar.CommandUShapeRebar()
```

Here is the call graph for this function:

```
RebarTools.UShapeRebarTool.
Activated Rebar RebarTool.
Activated RebarTool.
```

7.14.2.2 def RebarTools.UShapeRebarTool.GetResources (self)

Definition at line 52 of file RebarTools.py.

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, I_cover, r_cover, b_cover, t_cover, bent
 Length, bentAngle, orientation)
- def BentShapeRebar.makeBentShapeRebar (f_cover, b_cover, I_cover, r_cover, diameter, t_cover, bent
 Length, 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 -*-
00003 # *
00004 # *
             Copyright (c) 2017 - Amritpal Singh <amrit3701@gmail.com>
00005 # *
00006 # *
            This program is free software; you can redistribute it and/or modify
            it under the terms of the GNU Lesser General Public License (LGPL)
00007 # *
            as published by the Free Software Foundation; either version 2 of
00008 # *
            the License, or (at your option) any later version.
00009 # *
00010 # *
            for detail see the LICENCE text file.
00011 # *
00012 # *
            This program is distributed in the hope that it will be useful,
            but WITHOUT ANY WARRANTY; without even the implied warranty of
00013 # *
            MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
00014 # *
            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
             Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307
00019 # *
00020 # *
00021 # *
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):
          Return points of the LShape rebar in the form of array for sketch.
00042
          It takes four different orientations input i.e. 'Bottom', 'Top', 'Left', 'Right'.
00043
00045
          if orientation == "Bottom":
              x1 = FacePRM[1][0] - FacePRM[0][0] / 2 + 1_cover
y1 = FacePRM[1][1] + FacePRM[0][1] / 2 - t_cover
00046
00047
00048
              x2 = x1 + bentLength
00049
              y2 = y1
dis = (FacePRM[0][1] - t_cover - b_cover) * math.tan(math.radians(bentAngle - 90))
00050
00051
              y3 = FacePRM[1][1] - FacePRM[0][1] / 2 + b_cover
x4 = FacePRM[1][0] + FacePRM[0][0] / 2 - r_cover - bentLength - dis
00052
00053
00054
               y4 = y3
              x5 = x4 + dis
00055
00056
              y5 = y2
00057
              x6 = x5 + bentLength
00058
              y6 = y5
00059
          elif orientation == "Top":
             x1 = FacePRM[1][0] - FacePRM[0][0] / 2 + 1_cover
y1 = FacePRM[1][1] - FacePRM[0][1] / 2 + b_cover
00060
00061
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
              y3 = FacePRM[1][1] + FacePRM[0][1] / 2 - t_cover
x4 = FacePRM[1][0] + FacePRM[0][0] / 2 - r_cover - bentLength - dis
00066
00067
00068
              v4 = v3
              x5 = x4 + dis
00069
00070
              y5 = y2
00071
               x6 = x5 + bentLength
              y6 = y5
00072
00073
          elif orientation == "Left":
00074
             x1 = FacePRM[1][0] + FacePRM[0][0] / 2 - r_cover
              y1 = FacePRM[1][1] + FacePRM[0][1] / 2 - t_cover
00076
              x2 = x1
00077
               y2 = y1 - bentLength
               dis = (FacePRM[0][0] - r_cover - l_cover) * math.tan(math.radians(bentAngle - 90))
00078
00079
              x3 = FacePRM[1][0] - FacePRM[0][0] / 2 + 1_cover
              y3 = y2 - dis
00080
00081
              x4 = x3
00082
              y4 = FacePRM[1][1] - FacePRM[0][1] / 2 + b_cover + bentLength + dis
```

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

```
diameter = FreeCAD.Units.Quantity(diameter).Value
               rounding = self.form.rounding.value()
00162
00163
               orientation = self.form.orientation.currentText()
               amount_check = self.form.amount_radio.isChecked()
spacing_check = self.form.spacing_radio.isChecked()
00164
00165
00166
               if not self.Rebar:
                   if amount_check:
00167
00168
                       amount = self.form.amount.value()
00169
                       rebar = makeBentShapeRebar(f_cover, b_cover, l_cover, r_cover, diameter,
      t_cover, bentLength, bentAngle, rounding, True, amount, orientation, self.
      SelectedObj, self.FaceName)
00170
                   elif spacing_check:
                       spacing = self.form.spacing.text()
spacing = FreeCAD.Units.Quantity(spacing).Value
00171
00172
00173
                       rebar = makeBentShapeRebar(f_cover, b_cover, l_cover, r_cover, diameter,
      t_cover, bentLength, bentAngle, rounding, False, spacing, orientation, self.
      SelectedObj, self.FaceName)
00174
              else:
00175
                  if amount check:
00176
                       amount = self.form.amount.value()
                       rebar = editBentShapeRebar(self.Rebar, f_cover, b_cover, l_cover,
00177
      r_cover, diameter, t_cover, bentLength, bentAngle, rounding, True, amount, orientation, self.
      SelectedObj, self.FaceName)
00178
                   elif spacing check:
                       spacing = self.form.spacing.text()
spacing = FreeCAD.Units.Quantity(spacing).Value
00179
00180
                       rebar = editBentShapeRebar(self.Rebar, f_cover, b_cover, l_cover,
00181
      r_cover, diameter, t_cover, bentLength, bentAngle, rounding, False, spacing, orientation, self.
      SelectedObj, self.FaceName)
00182
              if self.CustomSpacing:
00183
                   rebar.CustomSpacing = self.CustomSpacing
00184
                   FreeCAD.ActiveDocument.recompute()
00185
               self.Rebar = rebar
00186
               if signal == int(QtGui.QDialogButtonBox.Apply):
00187
00188
               else:
00189
                   FreeCADGui.Control.closeDialog(self)
00190
00191
          def amount radio clicked(self):
00192
               self.form.spacing.setEnabled(False)
00193
               self.form.amount.setEnabled(True)
00194
          def spacing radio clicked(self):
00195
00196
               self.form.amount.setEnabled(False)
00197
               self.form.spacing.setEnabled(True)
00198
00199
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):
    """ makeBentShapeRebar(FrontCover, BottomCover, LeftCover, RightCover, Diameter, TopCover, BentLength,
00201
       BentAngle, Rounding,
00202
          AmountSpacingCheck, AmountSpacingValue, Orientation, Structure, Facename): Adds the Bent-Shape
       reinforcement bar to the
00203
          selected structural object.
           It takes four different orientations input i.e. 'Bottom', 'Top', 'Left', 'Right'.
00204
00205
00206
           if not structure and not facename:
               selected_obj = FreeCADGui.Selection.getSelectionEx()[0]
00207
               structure = selected_obj.Object
facename = selected_obj.SubElementNames[0]
00208
00209
00210
           face = structure.Shape.Faces[getFaceNumber(facename) - 1]
00211
           #StructurePRM = getTrueParametersOfStructure(structure)
           FacePRM = getParametersOfFace(structure, facename)
00212
           if not FacePRM:
00213
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
      , bentLength, bentAngle, orientation)
           import Part
00218
00219
           import Arch
00220
           sketch = FreeCAD.activeDocument().addObject('Sketcher::SketchObject', 'Sketch')
          sketch.MapMode = "FlatFace"
sketch.Support = [(structure, facename)]
00221
00222
00223
           FreeCAD.ActiveDocument.recompute()
00224
           sketch.addGeometry(Part.LineSegment(points[0], points[1]), False)
00225
           sketch.addGeometry(Part.LineSegment(points[1], points[2]), False)
00226
           sketch.addGeometry(Part.LineSegment(points[2], points[3]), False)
00227
          sketch.addGeometry(Part.LineSegment(points[3], points[4]), False)
00228
          sketch.addGeometry(Part.LineSegment(points[4], points[5]), False)
00229
           import Sketcher
00230
             amount_spacing_check:
00231
               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),
      f_cover)
           rebar.Rounding = rounding
00236
00237
           # Adds properties to the rebar object
      00238
           rebar.ViewObject.setEditorMode("RebarShape", 2)
rebar.addProperty("App::PropertyDistance", "FrontCover", "RebarDialog", QT_TRANSLATE_NOOP("
00239
00240
      App::Property", "Front cover of rebar")).FrontCover = f_cover
  rebar.setEditorMode("FrontCover", 2)
00241
           rebar.addProperty("App::PropertyDistance", "LeftCover", "RebarDialog", QT_TRANSLATE_NOOP("App::Property
00242
          "Left Side cover of rebar")).LeftCover = 1_cover
           rebar.setEditorMode("LeftCover", 2)
00243
00244
           rebar.addProperty("App::PropertyDistance", "RightCover", "RebarDialog", QT_TRANSLATE_NOOP("
      App::Property", "Right Side cover of rebar")).RightCover = r_cover rebar.setEditorMode("RightCover", 2) rebar.addProperty("App::PropertyDistance", "BottomCover", "RebarDialog", QT_TRANSLATE_NOOP("App::Property", "Bottom cover of rebar")).BottomCover = b_cover
00245
00246
00247
           rebar.setEditorMode("BottomCover", 2)
           rebar.addProperty("App::PropertyBool", "AmountCheck", "RebarDialog", QT_TRANSLATE_NOOP("App::Property",
00248
        "Amount radio button is checked")).AmountCheck rebar.setEditorMode("AmountCheck", 2)
00249
        rebar.addProperty("App::PropertyDistance", "TopCover", "RebarDialog", QT_TRANSLATE_NOOP("App::Property"
"Top cover of rebar")).TopCover = t_cover
00250
           rebar.setEditorMode("TopCover", 2)
00251
           rebar.addProperty("App::PropertyDistance", "TrueSpacing", "RebarDialog", QT_TRANSLATE_NOOP("
00252
      App::Property", "Spacing between of rebars")).TrueSpacing = amount_spacing_value rebar.addProperty("App::PropertyString", "Orientation", "RebarDialog", QT_TRANSLATE_NOOP("App::Property", "Shape of rebar")).Orientation = orientation
00253
00254
           rebar.setEditorMode("Orientation", 2)
00255
           rebar.setEditorMode("TrueSpacing", 2)
           rebar.addProperty("App::PropertyDistance", "BentLength", "RebarDialog", QT_TRANSLATE_NOOP("
00256
      App::Property", "BentLength cover of rebar")).BentLength = bentLength
           rebar.setEditorMode("BentLength", 2)
00257
          rebar.addProperty("App::PropertyDistance", "BentAngle", "RebarDialog", QT_TRANSLATE_NOOP("App::Property "Bent Angle of rebar")).BentAngle = bentAngle
00258
           rebar.setEditorMode("BentAngle", 2)
00259
00260
00261
           if amount_spacing_check:
00262
               rebar.AmountCheck = True
00263
           else:
00264
               rebar.AmountCheck = False
00265
               rebar.TrueSpacing = amount_spacing_value
           rebar.Label = "BentShapeRebar"
00266
00267
           FreeCAD.ActiveDocument.recompute()
00268
00269
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:
               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]
           structure = sketch.Support[0][0]
00280
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
           points = getpointsOfBentShapeRebar(FacePRM, l_cover, r_cover, b_cover, t_cover
00286
       , 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
00296
           sketch.movePoint(2, 1, points[2], 0)
00297
           FreeCAD.ActiveDocument.recompute()
00298
           sketch.movePoint(2, 2, points[3], 0)
00299
           FreeCAD.ActiveDocument.recompute()
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
```

```
sketch.movePoint(4, 1, points[4], 0)
00306
          FreeCAD.ActiveDocument.recompute()
00307
          sketch.movePoint(4, 2, points[5], 0)
00308
          {\tt FreeCAD.ActiveDocument.recompute()}
00309
00310
          Rebar.OffsetStart = f cover
00311
          Rebar.OffsetEnd = f_cover
00312
          if amount_spacing_check:
              Rebar.Amount = amount_spacing_value
00313
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
          Rebar.FrontCover = f_cover
Rebar.LeftCover = l_cover
00322
00323
00324
          Rebar.RightCover = r_cover
00325
          Rebar.BottomCover = b_cover
00326
          Rebar.TopCover = t_cover
00327
          Rebar.BentLength = bentLength
          Rebar.BentAngle = bentAngle
Rebar.Rounding = rounding
00328
00329
00330
          Rebar.TrueSpacing = amount_spacing_value
00331
          Rebar.Orientation = orientation
00332
          FreeCAD.ActiveDocument.recompute()
00333
          return Rebar
00334
00335 def editDialog(vobi):
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
00351
              obj.form.amount_radio.setChecked(False)
00352
              obj.form.spacing_radio.setChecked(True)
00353
              obj.form.amount.setDisabled(True)
00354
              obi.form.spacing.setEnabled(True)
              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()
          if selected_obj:
00362
              FreeCADGui.Control.showDialog(_BentShapeRebarTaskPanel())
```

8.3 HelicalRebar.py File Reference

Classes

• class HelicalRebar._HelicalRebarTaskPanel

Namespaces

HelicalRebar

8.4 HelicalRebar.py 111

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
           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 Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307
00019 # *
00020 # *
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 PvSide.OtCore import OT 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,
     diameter, size, direction):
    """ getpointsOfHelicalRebar(FacePRM, s_cover, b_cover, t_cover):
00040
          Return points of the LShape rebar in the form of array for sketch.""" dx = s_cover + diameter / 2
00041
00042
          dz = float(pitch) / edges
00043
00044
          R = diameter / 2 - dx
00045
          R = FacePRM[0][0] / 2 - s_cover
          points = []
00046
00047
          if direction[2] in {-1,1}:
00048
              z = 0
00049
             1 = 0
00050
              if direction[2] == 1:
```

```
zz = FacePRM[1][2] - t_cover
              elif direction[2] =
00052
00053
                 zz = FacePRM[1][2] + b_cover
00054
              count = 0
              flag = False
00055
00056
              while (round(z) < abs(size - b_cover - t_cover)):</pre>
                 for i in range(0, int(edges) + 1):
00058
                      if not i and flag:
00059
                         continue
00060
                      if not flag:
                         z -= dz
00061
                          flag = True
00062
00063
                      iAngle = i * 360 / edges
                      y = FacePRM[1][0] + R * math.cos(math.radians(iAngle))
y = FacePRM[1][1] + R * math.sin(math.radians(iAngle))
00064
00065
00066
                      points.append(FreeCAD.Vector(x, y, zz))
00067
                      count += 1
00068
                      if direction[2] == 1:
00069
                         zz -= dz
                      elif direction[2] == -1:
00070
00071
                         zz += dz
00072
                      z += dz
00073
         return points
00074
00075 def createHelicalWire (FacePRM, s_cover, b_cover, t_cover, pitch, size, direction, helix =
          """ createHelicalWire(FacePRM, SideCover, BottomCover, TopCover, Pitch, Size, Direction, Helix = None):
00076
00077
          It creates a helical wire."""
00078
          import Part
00079
          if not helix:
08000
             helix = FreeCAD.ActiveDocument.addObject("Part::Helix","Helix")
00081
          helix.Pitch = pitch
          helix.Radius = FacePRM[0][0] / 2 - s_cover
00082
00083
          helix.Angle = 0
00084
          helix.LocalCoord = 0
          helix.Height = size - b_cover - t_cover
00085
00086
          if round(direction.x) == 1:
              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.Rotation = FreeCAD.Rotation(FreeCAD.Vector(0, -1, 0), -90)
00091
00092
          elif round(direction.v) == 1:
              helix.Placement.Base = FreeCAD.Vector(FacePRM[1][0], FacePRM[1][1] - b_cover, FacePRM[1][2])
00093
              helix.Placement.Rotation = FreeCAD.Rotation(FreeCAD.Vector(1, 0, 0), 90)
00094
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)
              helix.Placement.Rotation = FreeCAD.Rotation(FreeCAD.Vector(0, 0, 1), 0)
00100
          elif round(direction.z) == -1:
00101
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
00107 class HelicalRebarTaskPanel:
00108
         def __init__(self, Rebar = None):
00109
              self.form = FreeCADGui.PySideUic.loadUi(os.path.splitext(__file_
                                                                               _)[0] + ".ui")
              self.form.setWindowTitle(QtGui.QApplication.translate("Arch", "Helical Rebar", None))
00110
00111
              if not Rebar:
00112
                 normal = facenormalDirection()
00113
              else:
                  normal = facenormalDirection(Rebar.Base.Support[0][0], Rebar.Base.Support[0]
00114
     [1][0])
00115
              if not round(normal.z) in {1, -1}:
                 self.form.topCoverLabel.setText(translate("RebarAddon", "Left Cover"))
00116
00117
                  self.form.bottomCoverLabel.setText(translate("RebarAddon", "Right Cover"))
00118
              self.form.PickSelectedFace.clicked.connect(self.getSelectedFace)
              self.form.image.setPixmap(QtGui.QPixmap(os.path.split(os.path.abspath(__file__))[0] + "
00119
     /icons/HelicalRebar.svg"))
00120
              self.form.toolButton.clicked.connect(lambda: showPopUpImageDialog(os.path.split
     (os.path.abspath(__file__))[0] + "/icons/HelicalRebarDetailed.svg"))
              self.form.toolButton.setIcon(self.form.toolButton.style().standardIcon(
00121
     QtGui.QStyle.SP_DialogHelpButton))
00122
             self.Rebar = Rebar
00123
              self.SelectedObj = None
00124
              self.FaceName = None
00125
         def getStandardButtons(self):
00126
00127
              return int(QtGui.QDialogButtonBox.Ok) | int(QtGui.QDialogButtonBox.Apply) | int(
     OtGui.ODialogButtonBox.Cancel)
00128
00129
          def clicked(self, button):
              if button == int(OtGui.ODialogButtonBox.Apply):
00130
                 self.accept (button)
00131
```

8.4 HelicalRebar.py 113

```
00132
00133
           def getSelectedFace(self):
00134
                getSelectedFace(self)
00135
               normal = facenormalDirection()
00136
               if not round(normal.z) in {1, -1}:
                    self.form.topCoverLabel.setText(translate("RebarAddon", "Left Cover"))
00137
00138
                    self.form.bottomCoverLabel.setText(translate("RebarAddon", "Right Cover"))
00139
                    self.form.topCoverLabel.setText(translate("RebarAddon", "Top Cover"))
00140
00141
                    self.form.bottomCoverLabel.setText(translate("RebarAddon", "Bottom Cover"))
00142
00143
00144
           def accept(self, signal = None):
               b_cover = self.form.bottomCover.text()
00145
00146
               b_cover = FreeCAD.Units.Quantity(b_cover).Value
00147
               s_cover = self.form.sideCover.text()
               s_cover = FreeCAD.Units.Quantity(s_cover).Value
00148
               t_cover = self.form.topCover.text()
00149
               t_cover = FreeCAD.Units.Quantity(t_cover).Value
00150
00151
               pitch = self.form.pitch.text()
00152
               pitch = FreeCAD.Units.Quantity(pitch).Value
               diameter = self.form.diameter.text()
diameter = FreeCAD.Units.Quantity(diameter).Value
00153
00154
00155
               if not self.Rebar:
                   rebar = makeHelicalRebar(s_cover, b_cover, diameter, t_cover, pitch, self.
00156
      SelectedObj, self.FaceName)
00157
               else:
00158
                    rebar = editHelicalRebar(self.Rebar, s_cover, b_cover, diameter, t_cover,
      pitch, self.SelectedObj, self.FaceName)
00159
               self.Rebar = rebar
00160
               if signal == int(QtGui.QDialogButtonBox.Apply):
00161
00162
00163
                    FreeCADGui.Control.closeDialog(self)
00164
00165 def makeHelicalRebar(s_cover, b_cover, diameter, t_cover, pitch, structure = None, facename
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]
               structure = selected_obj.Object
facename = selected_obj.SubElementNames[0]
00170
00171
00172
           face = structure.Shape.Faces[getFaceNumber(facename) - 1]
00173
           #StructurePRM = getTrueParametersOfStructure(structure)
00174
           FacePRM = getParametersOfFace(structure, facename, False)
           if not FacePRM:
00175
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
           helix = createHelicalWire(FacePRM, s_cover, b_cover, t_cover, pitch, size, normal)
00182
00183
           helix.Support = [(structure, facename)]
           rebar = Arch.makeRebar(structure, helix, diameter, 1, 0)
00184
00185
           rebar.OffsetStart = 0
00186
           rebar.OffsetEnd = 0
           FreeCAD.ActiveDocument.recompute()
00187
          # Adds properties to the rebar object
rebar.ViewObject.addProperty("App::PropertyString", "RebarShape", "RebarDialog", QT_TRANSLATE_NOOP("
o::Property", "Shape of rebar")).RebarShape = "HelicalRebar"
rebar.ViewObject.setEditorMode("RebarShape", 2)
rebar.addProperty("App::PropertyDistance", "SideCover", "RebarDialog", QT_TRANSLATE_NOOP("App::Property
00188
00189
      App::Property",
00190
00191
      ", "Front cover of rebar")).SideCover = s_cover
          rebar.setEditorMode("SideCover", 2)
00192
           rebar.addProperty("App::PropertyDistance", "Pitch", "RebarDialog", QT_TRANSLATE_NOOP("App::Property", "
00193
      Left Side cover of rebar")).Pitch = pitch
           rebar.setEditorMode("Pitch", 2)
           rebar.addProperty("App::PropertyDistance", "BottomCover", "RebarDialog", QT_TRANSLATE_NOOP("
00195
      App::Property", "Bottom cover of rebar")).BottomCover = b_cover
        rebar.setEditorMode("BottomCover", 2)
rebar.addProperty("App::PropertyDistance", "TopCover", "RebarDialog", QT_TRANSLATE_NOOP("App::Property"
"Top cover of rebar")).TopCover = t_cover
rebar.setEditorMode("TopCover", 2)
00196
00197
00198
00199
           rebar.Label = "HelicalRebar"
00200
           FreeCAD.ActiveDocument.recompute()
00201
           return rebar
00202
00203 def editHelicalRebar(Rebar, s_cover, b_cover, diameter, t_cover, pitch, structure = None,
      facename = None):
          sketch = Rebar.Base
00204
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
00211
          # Assigned values
00212
          facename = sketch.Support[0][1][0]
          structure = sketch.Support[0][0]
00213
          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)
          size = (ArchCommands.projectToVector(structure.Shape.copy(), face.normalAt(0, 0))).Length
normal = face.normalAt(0,0)
00218
00219
          #normal = face.Placement.Rotation.inverted().multVec(normal)
00220
          helix = createHelicalWire(FacePRM, s_cover, b_cover, t_cover, pitch, size, normal,
00221
     Rebar.Base)
00222
          FreeCAD.ActiveDocument.recompute()
00223
          Rebar.Diameter = diameter
          Rebar.SideCover = s_cover
00224
          Rebar.BottomCover = b_cover
00225
00226
          Rebar.TopCover = t_cover
00227
          Rebar.Pitch = pitch
00228
          FreeCAD.ActiveDocument.recompute()
00229
          return Rebar
00230
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
          \verb"obj.form.diameter.setText" (\verb"str" (vobj.Object.Diameter")")
00237
          obi.form.topCover.setText(str(vobi.Object.TopCover))
00238
          obj.form.pitch.setText(str(vobj.Object.Pitch))
00239
          FreeCADGui.Control.showDialog(obj)
00240
00241 def CommandHelicalRebar():
00242
          selected_obj = check_selected_face()
00243
          if selected obj:
              FreeCADGui.Control.showDialog(_HelicalRebarTaskPanel())
```

8.5 LShapeRebar.py File Reference

Classes

class LShapeRebar._LShapeRebarTaskPanel

Namespaces

LShapeRebar

Functions

- def LShapeRebar.getpointsOfLShapeRebar (FacePRM, I_cover, r_cover, b_cover, t_cover, orientation)
- def LShapeRebar.makeLShapeRebar (f_cover, b_cover, I_cover, r_cover, diameter, t_cover, rounding, amount_spacing_check, amount_spacing_value, orientation="Bottom Left", structure=None, face-name=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 115

8.6 LShapeRebar.py

```
00001 # -*- coding: utf-8 -*-
00004 # *
              Copyright (c) 2017 - Amritpal Singh <amrit3701@gmail.com>
00005 # *
00006 # *
             This program is free software; you can redistribute it and/or modify
             it under the terms of the GNU Lesser General Public License (LGPL)
00007 # *
             as published by the Free Software Foundation; either version 2 of
00008 # *
             the License, or (at your option) any later version.
00009 # *
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
             MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
00014 # *
             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 # *
00021 # *
00023
00024 __title__ = "LShapeRebar"
00025 __author__ = "Amritpal Singh"
00026 __url__ = "https://www.freecadweb.org"
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
{\tt 00040~def~getpointsOfLShapeRebar(FacePRM,~l\_cover,~r\_cover,~b\_cover,~t\_cover,~orientation):}
00041
            """ 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'
00042
00043
00044
           if orientation == "Bottom Left":
00046
               x1 = FacePRM[1][0] - FacePRM[0][0] / 2 + 1_cover
                y1 = FacePRM[1][1] + FacePRM[0][1] / 2 - t_cover
00047
               x2 = FacePRM[1][0] - FacePRM[0][0] / 2 + 1_cover

y2 = FacePRM[1][1] - FacePRM[0][1] / 2 + b_cover

x3 = FacePRM[1][0] - FacePRM[0][0] / 2 + FacePRM[0][0] - r_cover

y3 = FacePRM[1][1] - FacePRM[0][1] / 2 + b_cover
00048
00049
00050
00051
           elif orientation == "Bottom Right":
00052
             x1 = FacePRM[1][0] - FacePRM[0][0] / 2 + FacePRM[0][0] - r_cover
y1 = FacePRM[1][1] + FacePRM[0][1] / 2 - t_cover
00053
00054
               x2 = FacePRM[1][0] - FacePRM[0][0] / 2 + FacePRM[0][0] - r_cover

y2 = FacePRM[1][1] - FacePRM[0][1] / 2 + b_cover

x3 = FacePRM[1][0] - FacePRM[0][0] / 2 + 1_cover

y3 = FacePRM[1][1] - FacePRM[0][1] / 2 + b_cover
00055
00056
00057
00058
00059
           elif orientation == "Top Left":
             x1 = FacePRM[1][0] - FacePRM[0][0] / 2 + 1_cover
y1 = FacePRM[1][1] - FacePRM[0][1] / 2 + b_cover
00060
00061
                x2 = FacePRM[1][0] - FacePRM[0][0] / 2 + 1_cover
00062
               y2 = FacePRM[1][1] + FacePRM[0][1] / 2 - t_cover
00063
               x3 = FacePRM[1][0] - FacePRM[0][0] / 2 + FacePRM[0][0] - r_cover
00064
00065
                y3 = FacePRM[1][1] + FacePRM[0][1] / 2 - t_cover
           elif orientation == "Top Right":
00066
             x1 = FacePRM[1][0] - FacePRM[0][0] / 2 + FacePRM[0][0] - r_cover
y1 = FacePRM[1][1] - FacePRM[0][1] / 2 + b_cover
00067
00068
                x^2 = FacePRM[1][0] - FacePRM[0][0] / 2 + FacePRM[0][0] - r_cover
00069
               y2 = FacePRM[1][1] + FacePRM[0][1] / 2 - t_cover
               x3 = FacePRM[1][0] - FacePRM[0][0] / 2 + 1_cover
y3 = FacePRM[1][1] + FacePRM[0][1] / 2 - t_cover
00071
00072
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
                if not Rebar:
00079
08000
                    selected_obj = FreeCADGui.Selection.getSelectionEx()[0]
00081
                    self.SelectedObj = selected_obj.Object
                    self.FaceName = selected_obj.SubElementNames[0]
                else:
```

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

8.6 LShapeRebar.py 117

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

```
rebar.TrueSpacing = amount_spacing_value
          rebar.Label = "LShapeRebar"
00226
00227
          FreeCAD.ActiveDocument.recompute()
00228
          return rebar
00229
00230 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):
00231
          sketch = Rebar.Base
00232
          if structure and facename:
00233
              sketch.Support = [(structure, facename)]
          # Check if sketch support is empty.
00234
00235
          if not sketch.Support:
              showWarning("You have checked remove external geometry of base sketchs when needed.\nTo
00236
       unchecked Edit->Preferences->Arch.")
00237
00238
          # Assigned values
00239
          facename = sketch.Support[0][1][0]
          structure = sketch.Support[0][0]
00240
00241
          face = structure.Shape.Faces[getFaceNumber(facename) - 1]
00242
          #StructurePRM = getTrueParametersOfStructure(structure)
00243
          # Get parameters of the face where sketch of rebar is drawn
00244
          FacePRM = getParametersOfFace(structure, facename)
          # Get points of L-Shape rebar
00245
          points = getpointsOfLShapeRebar(FacePRM, l_cover, r_cover, b_cover, t_cover,
00246
     orientation)
00247
          sketch.movePoint(0, 1, points[0], 0)
00248
          FreeCAD.ActiveDocument.recompute()
00249
          sketch.movePoint(0, 2, points[1], 0)
00250
          FreeCAD.ActiveDocument.recompute()
00251
          sketch.movePoint(1, 1, points[1], 0)
          FreeCAD.ActiveDocument.recompute()
00252
00253
          sketch.movePoint(1, 2, points[2], 0)
00254
          FreeCAD.ActiveDocument.recompute()
00255
          Rebar.OffsetStart = f_cover
          Rebar.OffsetEnd = f_cover
00256
00257
          if amount_spacing_check:
00258
              Rebar.Amount = amount spacing value
              FreeCAD.ActiveDocument.recompute()
00260
              Rebar.AmountCheck = True
00261
00262
              size = (ArchCommands.projectToVector(structure.Shape.copy(), face.normalAt(0, 0))).Length
00263
              Rebar.Amount = int((size - diameter) / amount_spacing_value)
              FreeCAD.ActiveDocument.recompute()
00264
00265
              Rebar.AmountCheck = False
00266
          Rebar.Diameter = diameter
00267
          Rebar.FrontCover = f_cover
00268
          Rebar.LeftCover = 1_cover
          Rebar.RightCover = r_cover
00269
          Rebar.BottomCover = b_cover
00270
00271
          Rebar.TopCover = t cover
          Rebar.Rounding = rounding
00272
00273
          Rebar.TrueSpacing = amount_spacing_value
00274
          Rebar.Orientation = orientation
00275
          FreeCAD.ActiveDocument.recompute()
00276
          return Rebar
00277
00278 def editDialog(vobj):
          FreeCADGui.Control.closeDialog()
00279
00280
          obj = _LShapeRebarTaskPanel(vobj.Object)
00281
          obj.form.frontCover.setText(str(vobj.Object.FrontCover))
          obj.form.l_sideCover.setText(str(vobj.Object.LeftCover))
00282
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
          \verb|obj.form.orientation.setCurrentIndex(obj.form.orientation.findText(str(vobj.Object.Orientation))||
00289
          if vobi.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
00300 def CommandLShapeRebar():
00301
          selected_obj = check_selected_face()
00302
          if selected obj:
00303
              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 -*-
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 # *
00021 #
00023
00024 __title__ = "PopUpImage"
00025 __author__ = "Amritpal Singh"
00026 __url__ = "https://www.freecadweb.org"
00027
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)
            self.setWindowTitle(QtGui.QApplication.translate("RebarTool", "Detailed description", None)) self.verticalLayout = QtGui.QVBoxLayout(self)
00039
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."""
00045
00046
         dialog = PopUpImage(img)
         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/r9b517K.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) **Stra
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 R
00014 ![screenshot](https://www.freecadweb.org/wiki/images/3/35/Footing_UShapeRebar.png)
00016 -
          ![icon](https://www.freecadweb.org/wiki/images/thumb/3/38/Arch_Rebar_LShape.png/32px-Arch_Rebar_LShape.png) **LShape R
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) **Be
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) **Stirru
00023 ![screenshot] (https://www.freecadweb.org/wiki/images/9/9b/Stirrup.png)
00024
          ![icon](https://www.freecadweb.org/wiki/images/thumb/c/c9/Arch_Rebar_Helical.png/32px-Arch_Rebar_Helical.png) **Helical.png
\tt 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/ZOGCOoe.png)] (https://www.youtube.com/watch?v=BYOOjEKmx5E&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 \rightarrow Rebar tools \rightarrow Different rebars'').
00044 ## How it works
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
         ""makeStraightRebar'" function) of the rebar object and it will take ""Rebar" object as input which is created ""makeStraightRebar" function. In '"StraightRebar.py'", "'_StraightRebarTaskPanel'" class is present. This class loads UI(present in ""StriaghtRebar.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
          excuted.
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.getupleOfCustomSpacing (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 -*-
00004 # *
            Copyright (c) 2017 - Amritpal Singh <amrit3701@gmail.com>
00005 # *
            This program is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License (LGPL)
00006 # *
00007 # *
            as published by the Free Software Foundation; either version 2 of
00008 # *
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, but WITHOUT ANY WARRANTY; without even the implied warranty of
00013 # *
00014 # *
            MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
00015 # *
            GNU Library General Public License for more details.
00016 # *
00017 # *
            You should have received a copy of the GNU Library General Public
            License along with this program; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307
00018 # *
00019 # *
00020 # *
00023
00024 __title__ = "DialogDistribution"
00025 __author__ = "Amritpal Singh"
00026 __url__ = "https://www.freecadweb.org"
00028 from PySide import QtCore, QtGui
00029 from Rebarfunc import \star
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):
               self.FrontCover = frontCover
```

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

8.13 Rebarfunc.py File Reference

8.14 Rebarfunc.py 123

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 -*-
00003 # *
00004 # *
           Copyright (c) 2017 - Amritpal Singh <amrit3701@gmail.com>
00005 # *
00006 # *
           This program is free software; you can redistribute it and/or modify
           it under the terms of the GNU Lesser General Public License (LGPL)
00007 # *
00008 # *
           as published by the Free Software Foundation; either version 2 of
00009 # *
           the License, or (at your option) any later version.
           for detail see the LICENCE text file.
00010 # *
00011 # *
           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
           License along with this program; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307
00018 # *
00019 #
00020 #
00021 # *
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)
          vec2 = vec(edge2)
00041
00042
          angle = vec1.getAngle(vec2)
          angle = math.degrees(angle)
00043
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
               when edges not form a rectangular."""
00049
           angles = [round(getEdgesAngle(edges[0], edges[1])), round(
00050
      getEdgesAngle(edges[0], edges[2])),
00051
                   round(getEdgesAngle(edges[0], edges[3]))]
00052
           if angles.count(90) == 2 and (angles.count(180) == 1 or angles.count(0) == 1):
00053
              return True
00054
           else:
00055
               return False
00056
00057 def getBaseStructuralObject(obj):
           """ getBaseStructuralObject(obj): This function will return last base structural object."""
00058
00059
00060
           if not obj.Base:
00061
              return obi
00062
          else:
00063
               return getBaseStructuralObject(obj.Base)
00064
00065 def getBaseObject(obj):
00066    """ getBaseObject(obj): This function will return last base object."""
00067    if hasattr(obj, "Base"):
              return getBaseObject(obj.Base)
00069
           else:
00070
              return obj
00071
00072 def getFaceNumber(s):
00073 """ getFaceNumber(facename): This will return a face number from face name.
00074
           For eg.:
           Input: "Face12"
00075
00076
              Output: 12"""
          head = s.rstrip('0123456789')
tail = s[len(head):]
00077
00078
00079
          return int(tail)
00080
00081 def facenormalDirection(structure = None, facename = None):
00082
          if not structure and not facename:
00083
               selected_obj = FreeCADGui.Selection.getSelectionEx()[0]
               structure = selected_obj.Object
facename = selected_obj.SubElementNames[0]
00084
00085
00086
          face = structure.Shape.Faces[getFaceNumber(facename) - 1]
          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
00095 def getTrueParametersOfStructure(obj):
00096 """ getTrueParametersOfStructure(obj): This function return actual length,
           width and height of the structural element in the form of array like [Length, Width, Height]"""
00097
00098
           baseObject = getBaseObject(obj)
00099
00100
           # If selected_obj is not derived from any base object
00101
           if baseObject:
00102
               # If selected_obj is derived from SketchObject
               if baseObject.isDerivedFrom("Sketcher::SketchObject"):
00103
00104
                   edges = baseObject.Shape.Edges
                    if checkRectangle(edges):
00105
00106
                        for edge in edges:
00107
                             # Representation vector of edge
                             rep_vector = edge.Vertexes[1].Point.sub(edge.Vertexes[0].Point)
rep_vector_angle = round(math.degrees(rep_vector.getAngle(FreeCAD.Vector(1,0,0))))
00108
00109
                             if rep_vector_angle in {0, 180}:
00110
00111
                                 length = edge.Length
00112
                             else:
00113
                                 width = edge.Length
00114
00115
                      return None
00116
               else:
                   return None
00117
```

8.14 Rebarfunc.py 125

```
00118
              height = obj.Height.Value
00119
00120
               structuralBaseObject = getBaseStructuralObject(obj)
00121
               length = structuralBaseObject.Length.Value
               width = structuralBaseObject.Width.Value
00122
               height = structuralBaseObject.Height.Value
00123
00124
          return [length, width, height]
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
00132
          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
00133
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 #center_of_mass = center_of_mass.sub(getBaseStructuralObject(structure).Placement.Base)
00143
           center_of_mass = center_of_mass.sub(structure.Placement.Base)
00144
          Edges = []
00145
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:
                   if not Edges:
00152
00153
                      Edges.append(edge)
00154
                   else:
00155
                       # Checks whether similar edges is already present in Edges list
00156
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:
                   Edges.append(edge)
00160
00161
               # facePRM holds length of a edges.
               facePRM = [(vec(edge)).Length for edge in Edges]
00162
00163
               # Find the orientation of the face. Also eliminating normal axes
00164
               # to the edge/face.
00165
               \# When edge is parallel to x-axis
               if round(Edges[0].tangentAt(0)[0]) in {1,-1}:
00166
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]
               elif round(Edges[0].tangentAt(0)[2]) in {1,-1}:
00181
00182
                   y = center_of_mass[2]
00183
                   if round(Edges[1].tangentAt(0)[0]) in {1, -1}:
00184
                       x = center_of_mass[0]
                   else:
00185
00186
                      x = center_of_mass[1]
00187
                   facePRM.reverse()
               facelength = facePRM[0]
facewidth = facePRM[1]
00188
00189
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)};
                   normal = face.normalAt(0,0)
normal = face.Placement.Rotation.inverted().multVec(normal)
00197
00198
                   #print "x: ", boundbox.XLength
#print "y: ", boundbox.YLength
#print "z: ", boundbox.ZLength
00199
00200
00201
00202
                   \# Set length and width of user selected face of structural element
00203
                   flag = True
00204
                   # FIXME: Improve below logic.
```

```
for i in range(len(normal)):
                       if round(normal[i]) == 0:
   if flag and i == 0:
00206
00207
00208
                               x = center_of_mass[i]
00209
                                facelength = boundbox.XLength
00210
                           flag = False
elif flag and i == 1:
00211
00212
                                x = center_of_mass[i]
00213
                                facelength = boundbox.YLength
                                flag = False
00214
                           if i == 1:
    y = center_of_mass[i]
00215
00216
00217
                                facewidth = boundbox.YLength
00218
                            elif i == 2:
00219
                               y = center_of_mass[i]
00220
                                facewidth = boundbox.ZLength
          #print [(facelength, facewidth), (x, y)] # Return parameter of the face when rebar is not created from the sketch.
00221
00222
          # For eg. non-planar rebars like stirrup etc.
          if not sketch:
00224
00225
             center_of_mass = face.CenterOfMass
00226
              return [(facelength, facewidth), center_of_mass]
          \mbox{\tt\#TODO:} Add support when bounding box have depth. Here bounding box looks
00227
          # like cuboid. If we given curved face.
00228
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):
           """ extendedTangentPartLength(rounding, diameter, angle): Get a extended
          length of rounding on corners."""
00237
00238
          radius = rounding * diameter
          x1 = radius / math.tan(math.radians(angle))
x2 = radius / math.cos(math.radians(90 - angle)) - radius
00239
00240
00241
          return x1 + x2
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.""
          radius = rounding * diameter
00246
          x1 = radius / math.sin(math.radians(angle))
x2 = radius * math.tan(math.radians(90 - angle))
00247
00248
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.")
              selected_obj = None
00262
          else:
00263
              selected_face_names = selected_objs[0].SubElementNames
00264
              if not selected_face_names:
                   selected_obj = None
showWarning("Select any face of the structural element.")
00265
00266
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
                  elif len(selected_face_names) == 1:
00271
00272
                       selected_obj = selected_objs[0]
00273
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()
          if selected_objs:
00280
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]
                       self.form.PickSelectedFaceLabel.setText("Selected face is " + self.FaceName)
00285
00286
                   else:
00287
                       showWarning("Select any face of the structural element.")
00288
00289
                   showWarning("Select only one face of the structural element.")
00290
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 = OtGui.OMessageBox()
           msg.setIcon(QtGui.QMessageBox.Warning)
00297
           msg.setText(translate("RebarAddon", message))
00298
00299
           msg.setStandardButtons(QtGui.QMessageBox.Ok)
00300
           msq.exec_()
00301
00302 # Qt tanslation handling
00303 def translate(context, text, disambig=None):
00304
          return OtCore.OCoreApplication.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 -*-
00004 # *
            Copyright (c) 2017 - Amritpal Singh <amrit3701@gmail.com>
00005 # *
            This program is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License (LGPL)
00006 # *
00007 # *
00008 # *
            as published by the Free Software Foundation; either version 2 of
            the License, or (at your option) any later version.
00009 # *
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 MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
00014 # *
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 # *
00021 #
```

```
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]+'
     /icons/dropdown_list/StraightRebar.svg',
         'MenuText': QT_TRANSLATE_NOOP("RebarAddon", "Straight Rebar"),
'ToolTip': QT_TRANSLATE_NOOP("RebarAddon", "Creates a Striaght bar reinforcement from the
00036
00037
       selected face of the Structural element.")}
00038
00039
         def IsActive(self):
00040
           if FreeCADGui.ActiveDocument:
00041
                 return True
00042
              else:
00043
                 return False
00044
00045
         def Activated(self):
00046
            import StraightRebar
00047
              # Call to CommandStraightRebar() function
00048
              StraightRebar.CommandStraightRebar()
00049
00050 class UShapeRebarTool:
00051
         def GetResources(self):
              return {'Pixmap' : os.path.split(os.path.abspath(__file__))[0]+'
00053
     /icons/dropdown_list/UShapeRebar.svg',
                    'MenuText': QT_TRANSLATE_NOOP("RebarAddon", "U-Shape Rebar"),
'ToolTip': QT_TRANSLATE_NOOP("RebarAddon", "Creates a U-Shape bar reinforcement from the
00054
00055
      selected face of the Structural element.")}
00056
00057
         def IsActive(self):
00058
            if FreeCADGui.ActiveDocument:
00059
                 return True
             else:
00060
00061
                 return False
00062
         def Activated(self):
00063
00064
              import UShapeRebar
00065
              # Call to CommandUShaepRebar() function
00066
              {\tt UShapeRebar.CommandUShapeRebar()}\\
00067
00068 class LShapeRebarTool:
00069
00070
          def GetResources(self):
00071
             return {'Pixmap' : os.path.split(os.path.abspath(__file__))[0]+'
     00072
00073
       selected face of the Structural element.")}
00074
00075
         def IsActive(self):
00076
             if FreeCADGui.ActiveDocument:
00077
                 return True
00078
              else:
00079
                 return False
08000
00081
         def Activated(self):
00082
             import LShapeRebar
00083
              # Call to CommandUShaepRebar() function
00084
              LShapeRebar.CommandLShapeRebar()
00085
00086 class StirrupTool:
00087
00088
          def GetResources(self):
     00089
00090
       selected face of the Structural element.")}
00092
00093
         def IsActive(self):
00094
             if FreeCADGui.ActiveDocument:
00095
                 return True
00096
              else:
00097
                 return False
00098
00099
         def Activated(self):
00100
              import Stirrup
              # Call to CommandStirrup() function
00101
```

```
Stirrup.CommandStirrup()
00103
00104 class BentShapeRebarTool:
00105
          def GetResources(self):
00106
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"),
                     'MenuText': QI_IRANSLAIE_NOOF( RebarAddon", "Creates a BentShape bar reinforcement from the
00109
       selected face of the Structural element.")}
00110
00111
          def IsActive(self):
00112
              if FreeCADGui.ActiveDocument:
                   return True
00113
00114
00115
                  return False
00116
         def Activated(self):
00117
              import BentShapeRebar
00119
               # Call to CommandBentShaepRebar() function
00120
               BentShapeRebar.CommandBentShapeRebar()
00121
00122 class HelicalRebarTool:
00123
          def GetResources(self):
00124
               return {'Pixmap' : os.path.split(os.path.abspath(__file__))[0]+'
      /icons/dropdown_list/HelixShapeRebar.svg'
                      'MenuText': QT_TRANSLATE_NOOP("RebarAddon", "Helical Rebar"),
'ToolTip': QT_TRANSLATE_NOOP("RebarAddon", "Creates a Helical bar reinforcement from the
00126
00127
       selected face of the Structural element.")}
00128
          def IsActive(self):
00130
            if FreeCADGui.ActiveDocument:
00131
                   return True
00132
00133
                   return False
00134
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
00148 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, I_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, I_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 -*-
00003 # *
00004 # *
            Copyright (c) 2017 - Amritpal Singh <amrit3701@gmail.com>
00005 # *
            This program is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License (LGPL)
00006 # *
00007 # *
            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 # *
            You should have received a copy of the GNU Library General Public License along with this program; if not, write to the Free Software
00017 # *
00018 # *
00019 # *
            Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307
00020 # *
00021 #
00023
00024 __title__ = "StirrupRebar"
00025 __uthor__ = "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,
     bentFactor, diameter, rounding, facenormal):

""" getpointsOfStirrup(FacePRM, LeftCover, RightCover, TopCover, BottomCover, BentAngle, BentFactor,
00041
      Diameter, Rounding, FaceNormal):
00042
         Return the coordinates points of the Stirrup in the form of array."""
00043
          angle = 180 - bentAngle
          tangent_part_length = extendedTangentPartLength(rounding, diameter, angle)
00044
          tangent_length = extendedTangentLength(rounding, diameter, angle)
00045
00046
          if round(facenormal[0]) in {1,-1}:
00047
              x1 = FacePRM[1][0]
              y1 = FacePRM[1][1] - FacePRM[0][0] / 2 + 1_cover
00048
             z1 = FacePRM[1][2] + FacePRM[0][1] / 2 - t_cover + tangent_part_length
00049
              y2 = FacePRM[1][1] - FacePRM[0][0] / 2 + 1_cover
00050
00051
              z2 = FacePRM[1][2] - FacePRM[0][1] / 2 + b_cover
              y3 = FacePRM[1][1] + FacePRM[0][0] / 2 - r_cover
00052
              z3 = FacePRM[1][2] - FacePRM[0][1] / 2 + b_cover
00053
              y4 = FacePRM[1][1] + FacePRM[0][0] / 2 - r_cover
00054
              z4 = FacePRM[1][2] + FacePRM[0][1] / 2 - t_cover
00055
             y5 = FacePRM[1][1] - FacePRM[0][0] / 2 + 1_cover - tangent_part_length
00056
00057
              z5 = FacePRM[1][2] + FacePRM[0][1] / 2 - t_cover
             side_length = abs(y5 - y4) - tangent_part_length
normal_dis = (diameter * (side_length + tangent_part_length)) / side_length
00058
00059
             x2 = x1 - normal_dis / 4
00060
             x3 = x2 - normal_dis / 4
00061
00062
             x4 = x3 - normal_dis / 4
             x5 = x4 - normal_dis / 4
00063
00064
              x0 = x1 + normal_dis / 4
00065
             y0 = y1 + (tangent_length + bentFactor * diameter) * math.sin(math.radians(angle))
              z0 = z1 - (tangent_length + bentFactor * diameter) * math.cos(math.radians(angle))
00066
             x6 = x5 - normal_dis / 4
00067
             y6 = y5 + (tangent_length + bentFactor * diameter) * math.sin(math.radians(90 - angle))
00068
              z6 = z5 - (tangent_length + bentFactor * diameter) * math.cos(math.radians(90 - angle))
00069
```

8.18 Stirrup.py 131

```
elif round(facenormal[1]) in {1,-1}:
              x1 = FacePRM[1][0] - FacePRM[0][0] / 2 + 1_cover
00071
00072
               y1 = FacePRM[1][1]
00073
               \mbox{z1 = FacePRM[1][2] + FacePRM[0][1] / 2 - t\_cover + tangent\_part\_length} \label{eq:z1}
               x2 = FacePRM[1][0] - FacePRM[0][0] / 2 + 1_cover
00074
               z2 = FacePRM[1][2] - FacePRM[0][1] / 2 + b_cover
00075
               x3 = FacePRM[1][0] + FacePRM[0][0] / 2 - r_cover
00076
00077
               z3 = FacePRM[1][2] - FacePRM[0][1] / 2 + b_cover
               x4 = FacePRM[1][0] + FacePRM[0][0] / 2 - r_cover
z4 = FacePRM[1][2] + FacePRM[0][1] / 2 - t_cover
00078
00079
               x5 = FacePRM[1][0] - FacePRM[0][0] / 2 + 1_cover - tangent_part_length
00080
               z5 = FacePRM[1][2] + FacePRM[0][1] / 2 - t_cover side_length = abs(x5 - x4) - tangent_part_length normal_dis = (diameter * (side_length + tangent_part_length)) / side_length
00081
00082
00083
00084
               y2 = y1 - normal_dis / 4
00085
               y3 = y2 - normal_dis / 4
               y4 = y3 - normal_dis / 4
00086
               y5 = y4 - normal_dis / 4
00087
00088
               y0 = y1 + normal_dis / 4
               x0 = x1 + (tangent_length + bentFactor * diameter) * math.sin(math.radians(angle))
00089
               z0 = z1 - (tangent_length + bentFactor * diameter) * math.cos(math.radians(angle))
x6 = x5 + (tangent_length + bentFactor * diameter) * math.sin(math.radians(90 - angle))
00090
00091
               y6 = y5 - normal_dis / 4
z6 = z5 - (tangent_length + bentFactor * diameter) * math.cos(math.radians(90 - angle))
00092
00093
00094
          elif round(facenormal[2]) in {1,-1}:
             x1 = FacePRM[1][0] - FacePRM[0][0] / 2 + 1_cover
00095
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 + 1_cover
               y2 = FacePRM[1][1] - FacePRM[0][1] / 2 + b_cover
00099
00100
               x3 = FacePRM[1][0] + FacePRM[0][0] / 2 - r_cover
00101
               y3 = FacePRM[1][1] - FacePRM[0][1] / 2 + b_cover
               x4 = FacePRM[1][0] + FacePRM[0][0] / 2 - r_cover
00102
               y4 = FacePRM[1][1] + FacePRM[0][1] / 2 - t_cover
00103
               x5 = FacePRM[1][0] - FacePRM[0][0] / 2 + 1_cover - tangent_part_length
y5 = FacePRM[1][1] + FacePRM[0][1] / 2 - t_cover
00104
00105
               side_length = abs(x5 - x4) - tangent_part_length
normal_dis = (diameter * (side_length + tangent_part_length)) / side_length
00106
00107
00108
               z2 = z1 - normal_dis / 4
               z3 = z2 - normal_dis / 4
00109
00110
               z4 = z3 - normal_dis / 4
               z5 = z4 - normal_dis / 4
00111
               z0 = z1 + normal dis / 4
00112
               x0 = x1 + (tangent_length + bentFactor * diameter) * math.sin(math.radians(angle))
00113
               y0 = y1 - (tangent_length + bentFactor * diameter) * math.cos(math.radians(angle))
00114
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))
               z6 = z5 - normal_dis / 4
00117
          return [FreeCAD.Vector(x0, y0, z0), FreeCAD.Vector(x1, y1, z1),
00118
                    FreeCAD.Vector(x2, y2, z2), FreeCAD.Vector(x3, y3, z3), FreeCAD.Vector(x4, y4, z4), FreeCAD.Vector(x5, y5, z5),
00119
00120
00121
                    FreeCAD. Vector (x6, y6, z6)]
00122
00123 class _StirrupTaskPanel:
          def __init__(self, Rebar = None):
    self.CustomSpacing = None
00124
00125
               if not Rebar:
00126
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]
               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.addItems(["135", "90"])
00136
               \verb|self.form.amount_radio.clicked.connect(self.amount_radio_clicked)|\\
               self.form.spacing radio.clicked.connect(self.spacing radio clicked)
00137
00138
               self.form.image.setPixmap(OtGui.OPixmap(os.path.split(os.path.abspath( file )))[0]+"
      /icons/Stirrup.svg"))
00139
              self.form.customSpacing.clicked.connect(lambda: runRebarDistribution(self))
00140
               self.form.removeCustomSpacing.clicked.connect(lambda:
      removeRebarDistribution(self))
00141
              self.form.PickSelectedFace.clicked.connect(lambda: getSelectedFace(self))
               self.form.toolButton.setIcon(self.form.toolButton.style().standardIcon(
00142
      QtGui.QStyle.SP_DialogHelpButton))
00143
               self.form.toolButton.clicked.connect(lambda: showPopUpImageDialog(os.path.split
      (os.path.abspath(__file__))[0] + "/icons/StirrupDetailed.svg"))
00144
               self.Rebar = Rebar
00145
           def getStandardButtons(self):
00146
00147
               return int(QtGui.QDialogButtonBox.Ok) | int(QtGui.QDialogButtonBox.Apply) | int(
      OtGui.ODialogButtonBox.Cancel)
00148
           def clicked(self, button):
00149
               if button == int(OtGui.ODialogButtonBox.Apply):
00150
00151
                    self.accept (button)
```

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

8.18 Stirrup.py 133

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

```
FreeCAD.ActiveDocument.recompute()
               Rebar.AmountCheck = True
00313
00314
               size = (ArchCommands.projectToVector(structure.Shape.copy(), face.normalAt(0, 0))).Length
00315
               Rebar.Amount = int((size - diameter) / amount_spacing_value)
               FreeCAD.ActiveDocument.recompute()
00316
               Rebar.AmountCheck = False
00317
00318
          Rebar.FrontCover = f_cover
00319
          Rebar.LeftCover = 1_cover
00320
          Rebar.RightCover = r_cover
          Rebar.TopCover = t_cover
00321
00322
          Rebar.BottomCover = b_cover
          Rebar.TrueSpacing = amount_spacing_value
00323
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))
          obj.form.t_sideCover.setText(str(vobj.Object.TopCover))
obj.form.b_sideCover.setText(str(vobj.Object.BottomCover))
00333
00334
00335
          obj.form.diameter.setText(str(vobj.Object.Diameter))
          obj.form.bentAngle.setCurrentIndex(obj.form.bentAngle.findText(str(vobj.Object.BentAngle)))
00336
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)
          obj.form.spacing.setText(str(vobj.Object.TrueSpacing))
#obj.form.PickSelectedFace.setVisible(False)
00346
00347
          FreeCADGui.Control.showDialog(obj)
00349
00350 def CommandStirrup():
00351
          selected_obj = check_selected_face()
          if selected_obj:
00352
              FreeCADGui.Control.showDialog(_StirrupTaskPanel())
00353
```

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.editDialog (vobj)
- def StraightRebar.CommandStraightRebar ()

8.20 StraightRebar.py 135

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 -*-
00003 # *
00004 # *
                Copyright (c) 2017 - Amritpal Singh <amrit3701@gmail.com>
00005 # *
                This program is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License (LGPL)
00006 # *
00007 # *
                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.
00015 # *
                GNU Library General Public License for more details.
00016 # *
                You should have received a copy of the GNU Library General Public License along with this program; if not, write to the Free Software
00017 # *
00018 # *
00019 # *
                Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307
00020 #
00021 #
00023
00024 __title__ = "StraightRebar"
00025 __uthor_ = "Amritpal Singh"
00026 __url__ = "https://www.freecadweb.org"
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
             Case I: When Orientation is 'Horizontal':
00044
00045
                  We have two option in CoverAlong i.e. 'Bottom Side' or 'Top Side'
              Case II: When Orientation is 'Vertical':
00046
              We have two option in CoverAlong i.e. 'Left Side' or 'Right Side'
00047
00048
             if orientation == "Horizontal":
    if coverAlong[0] == "Bottom Side":
00049
00050
                        x1 = FacePRM[1][0] - FacePRM[0][0] / 2 + lb_cover
00051
             x1 = FacePRM[1][0] - FacePRM[0][0] / 2 + lb_cover
y1 = FacePRM[1][1] - FacePRM[0][1] / 2 + coverAlong[1]
x2 = FacePRM[1][0] - FacePRM[0][0] / 2 + FacePRM[0][0] - rt_cover
y2 = FacePRM[1][1] - FacePRM[0][1] / 2 + coverAlong[1]
elif coverAlong[0] == "Top Side":
    cover = FacePRM[0][1] - coverAlong[1]
    x1 = FacePRM[1][0] - FacePRM[0][0] / 2 + lb_cover
    y1 = FacePRM[1][1] - FacePRM[0][1] / 2 + cover
    x2 = FacePRM[1][0] - FacePRM[0][0] / 2 + FacePRM[0][0] - rt_cover
    y2 = FacePRM[1][1] - FacePRM[0][1] / 2 + cover
elif orientation == "Vertical":
    if coverAlong[0] == "Left Side":
    x1 = FacePRM[1][0] - FacePRM[0][0] / 2 + coverAlong[1]
00052
00053
00054
00055
00056
00057
00058
00059
00060
00061
00062
                        x1 = FacePRM[1][0] - FacePRM[0][0] / 2 + coverAlong[1]
y1 = FacePRM[1][1] - FacePRM[0][1] / 2 + lb_cover
x2 = FacePRM[1][0] - FacePRM[0][0] / 2 + coverAlong[1]
00063
00064
00065
                   y2 = FacePRM[1][1] - FacePRM[0][1] / 2 + FacePRM[0][1] - rt_cover elif coverAlong[0] == "Right Side":
00066
00067
                        cover = FacePRM[0][0] - coverAlong[1]

x1 = FacePRM[1][0] - FacePRM[0][0] / 2 + cover

y1 = FacePRM[1][1] - FacePRM[0][1] / 2 + lb_cover
00068
00069
00070
                         x2 = FacePRM[1][0] - FacePRM[0][0] / 2 + cover
```

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

8.20 StraightRebar.py 137

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

```
00225
               size = (ArchCommands.projectToVector(structure.Shape.copy(), face.normalAt(0, 0))).Length
               rebar = Arch.makeRebar(structure, sketch, diameter, int((size - diameter) / amount_spacing_value),
      f_cover)
00227
           # Adds properties to the rebar object
           rebar.ViewObject.addProperty("App::PropertyString", "RebarShape", "RebarDialog", QT_TRANSLATE_NOOP(":Property", "Shape of rebar")).RebarShape = "StraightRebar"
00228
      App::Property",
           rebar.ViewObject.setEditorMode("RebarShape", 2)
rebar.addProperty("App::PropertyDistance", "FrontCover", "RebarDialog", QT_TRANSLATE_NOOP("
00229
00230
      App::Property", "Front cover of rebar")).FrontCover = f_cover
00231
           rebar.setEditorMode("FrontCover", 2)
      rebar.addProperty("App::PropertyDistance", "RightTopCover", "RebarDialog", QT_TRANSLATE_NOOP("App::Property", "Right/Top Side cover of rebar")).RightTopCover = rt_cover
00232
           rebar.setEditorMode("RightTopCover", 2)
00233
           rebar.addProperty("App::PropertyDistance", "LeftBottomCover", "RebarDialog", QT_TRANSLATE_NOOP("
00234
      App::Property", "Left/Bottom Side cover of rebar")).LeftBottomCover = lb_cover
           rebar.setEditorMode("LeftBottomCover", 2)
rebar.addProperty("App::PropertyString", "CoverAlong", "RebarDialog", QT_TRANSLATE_NOOP("App::Property"
00235
00236
        "Cover along")).CoverAlong = coverAlong[0]
           rebar.setEditorMode("CoverAlong", 2)
00237
           rebar.addProperty("App::PropertyDistance", "Cover", "RebarDialog", QT_TRANSLATE_NOOP("App::Property", "
00238
      Cover of rebar along user selected side")).Cover = coverAlong[1]
00239
           rebar.setEditorMode("Cover", 2)
           rebar.addProperty("App::PropertyBool", "AmountCheck", "RebarDialog", QT_TRANSLATE_NOOP("App::Property",
00240
        "Amount radio button is checked")) AmountCheck
00241
           rebar.setEditorMode("AmountCheck", 2)
           rebar.addProperty("App::PropertyDistance", "TrueSpacing", "RebarDialog", QT_TRANSLATE_NOOP("
00242
      App::Property", "Spacing between of rebars")).TrueSpacing = amount_spacing_value
00243
           rebar.setEditorMode("TrueSpacing", 2)
00244
           rebar.addProperty("App::PropertyString", "Orientation", "RebarDialog", QT_TRANSLATE_NOOP("App::Property
          "Shape of rebar")).Orientation = orientation
00245
           rebar.setEditorMode("Orientation", 2)
00246
           if amount_spacing_check:
               rebar.AmountCheck = True
00247
00248
           else:
           rebar.AmountCheck = False
  rebar.TrueSpacing = amount_spacing_value
rebar.Label = "StraightRebar"
00249
00250
00251
00252
           FreeCAD.ActiveDocument.recompute()
00253
00254
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:
               {\tt showWarning("You\ have\ checked\ remove\ external\ geometry\ of\ base\ sketchs\ when\ needed.} \setminus {\tt nTo}
00262
       unchecked Edit->Preferences->Arch.")
00263
00264
           # Assigned values
00265
           facename = sketch.Support[0][1][0]
00266
           structure = sketch.Support[0][0]
           face = structure.Shape.Faces[getFaceNumber(facename) - 1]
00267
           #StructurePRM = getTrueParametersOfStructure(structure)
00268
           # 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,
      orientation)
00273
           sketch.movePoint(0, 1, points[0], 0)
FreeCAD.ActiveDocument.recompute()
00274
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:
    Rebar.Amount = amount_spacing_value
00280
               FreeCAD.ActiveDocument.recompute()
00281
00282
               Rebar.AmountCheck = True
00283
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
           Rebar.RightTopCover = rt_cover
00289
           Rebar.LeftBottomCover = lb_cover
00290
00291
           Rebar.CoverAlong = coverAlong[0]
           Rebar.Cover = coverAlong[1]
00292
00293
           Rebar.TrueSpacing = amount_spacing_value
00294
           Rebar.Diameter = diameter
00295
           Rebar.Orientation = orientation
00296
           FreeCAD.ActiveDocument.recompute()
00297
           return Rebar
00298
```

```
00299 def editDialog(vobj):
00300
           FreeCADGui.Control.closeDialog()
00301
           obj = _StraightRebarTaskPanel(vobj.Object)
00302
           \verb"obj.form.frontCover.setText(str(vobj.Object.FrontCover)")"
00303
           obj.form.r_sideCover.setText(str(vobj.Object.RightTopCover))
          obj.form.l_sideCover.setText(str(vobj.Object.LeftBottomCover))
obj.form.bottomCover.setText(str(vobj.Object.Cover))
00304
00305
00306
           obj.form.diameter.setText(str(vobj.Object.Diameter))
00307
           \verb|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)
           obj.form.spacing.setText(str(vobj.Object.TrueSpacing))
#obj.form.PickSelectedFace.setVisible(False)
00316
00317
00318
          FreeCADGui.Control.showDialog(obj)
00319
00320 def CommandStraightRebar():
00321
          selected_obj = check_selected_face()
          if selected_obj:
00322
00323
               FreeCADGui.Control.showDialog(_StraightRebarTaskPanel())
```

8.21 UShapeRebar.py File Reference

Classes

class UShapeRebar. UShapeRebarTaskPanel

Namespaces

UShapeRebar

Functions

- def UShapeRebar.getpointsOfUShapeRebar (FacePRM, r cover, I 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 -*-
00003 # *
00004 # *
              Copyright (c) 2017 - Amritpal Singh <amrit3701@gmail.com>
00005 # *
00006 # *
              This program is free software; you can redistribute it and/or modify
              it under the terms of the GNU Lesser General Public License (LGPL)
00007 # *
              as published by the Free Software Foundation; either version 2 of
00008 # *
              the License, or (at your option) any later version.
00009 # *
00010 # *
              for detail see the LICENCE text file.
00011 # *
00012 # *
              This program is distributed in the hope that it will be useful,
              but WITHOUT ANY WARRANTY; without even the implied warranty of
00013 # *
              MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
00014 # *
              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 # *
00021 # *
00023
00024 __title__ = "UShapeRebar"
00025 __author__ = "Amritpal Singh"
00026 __url__ = "https://www.freecadweb.org"
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
00040 def 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'.
00041
00042
00043
00044
           if orientation == "Bottom":
00045
                x1 = FacePRM[1][0] - FacePRM[0][0] / 2 + 1_cover
00047
                 y1 = FacePRM[1][1] + FacePRM[0][1] / 2 - t_cover
                 x^2 = FacePRM[1][0] - FacePRM[0][0] / 2 + 1_cover
00048
00049
                y2 = FacePRM[1][1] - FacePRM[0][1] / 2 + b_cover
                x3 = FacePRM[1][0] - FacePRM[0][0] / 2 + FacePRM[0][0] - r_cover
00050
                y3 = FacePRM[1][1] - FacePRM[0][1] / 2 + b_cover
00051
                x4 = FacePRM[1][0] - FacePRM[0][0] / 2 + FacePRM[0][0] - r_cover
00052
                y4 = FacePRM[1][1] + FacePRM[0][1] / 2 - t_cover
00053
00054
         elif orientation == "Top":
           x1 = FacePRM[1][0] - FacePRM[0][0] / 2 + 1_cover
y1 = FacePRM[1][1] - FacePRM[0][1] / 2 + b_cover
x2 = FacePRM[1][0] - FacePRM[0][0] / 2 + 1_cover
00055
00056
00057
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
         AS - FACEFAM[1][0] - FACEFAM[0][0] / 2 + FACEPRM[0][0] - r_cover

y3 = FACEPRM[1][1] + FACEPRM[0][1] / 2 - t_cover

x4 = FACEPRM[1][0] - FACEPRM[0][0] / 2 + FACEPRM[0][0] - r_cover

y4 = FACEPRM[1][1] - FACEPRM[0][1] / 2 + b_cover

elif orientation == "Left":
00060
00061
00062
00063
             x1 = FacePRM[1][0] - FacePRM[0][0] / 2 + FacePRM[0][0] - r_cover
00064
                y1 = FacePRM[1][1] + FacePRM[0][1] / 2 - t_cover
00065
00066
                x2 = FacePRM[1][0] - FacePRM[0][0] / 2 + 1_cover
                y2 = FacePRM[1][1] + FacePRM[0][1] / 2 - t_cover
00067
               x3 = FacePRM[1][0] - FacePRM[0][0] / 2 + 1_cover
y3 = FacePRM[1][1] - FacePRM[0][1] / 2 + b_cover
x4 = FacePRM[1][0] - FacePRM[0][0] / 2 + FacePRM[0][0] - r_cover
00068
00069
00070
                y4 = FacePRM[1][1] - FacePRM[0][1] / 2 + b_cover
00072
         elif orientation == "Right":
           x1 = FacePRM[1][0] - FacePRM[0][0] / 2 + 1_cover
y1 = FacePRM[1][1] + FacePRM[0][1] / 2 - t_cover
00073
00074
                x2 = FacePRM[1][0] - FacePRM[0][0] / 2 + FacePRM[0][0] - r_cover
00075
00076
                y2 = FacePRM[1][1] + FacePRM[0][1] / 2 - t_cover
                x3 = FacePRM[1][0] - FacePRM[0][0] / 2 + FacePRM[0][0] - r_cover
           y3 = FacePRM[1][0] - FacePRM[0][1] / 2 + b_cover

y4 = FacePRM[1][0] - FacePRM[0][0] / 2 + 1_cover

y4 = FacePRM[1][1] - FacePRM[0][1] / 2 + b_cover

return [FreeCAD.Vector(x1, y1, 0), FreeCAD.Vector(x2, y2, 0),\

FreeCAD.Vector(x3, y3, 0), FreeCAD.Vector(x4, y4, 0)]
00078
00079
08000
00081
00082
00084 class _UShapeRebarTaskPanel:
```

8.22 UShapeRebar.py 141

```
00085
                _init__(self, Rebar = None):
              self.CustomSpacing = None
00086
              if not Rebar:
00087
00088
                  selected_obj = FreeCADGui.Selection.getSelectionEx()[0]
                  self.SelectedObj = selected_obj.Object
00089
00090
                  self.FaceName = selected_obj.SubElementNames[0]
00091
00092
                  self.FaceName = Rebar.Base.Support[0][1][0]
                   self.SelectedObj = Rebar.Base.Support[0][0]
00093
00094
              self.form = FreeCADGui.PySideUic.loadUi(os.path.splitext(__file__)[0] + ".ui")
              self.form.setWindowTitle(QtGui.QApplication.translate("RebarAddon", "U-Shape Rebar", None))
self.form.orientation.addItems(["Bottom", "Top", "Right", "Left"])
00095
00096
00097
              self.form.amount_radio.clicked.connect(self.amount_radio_clicked)
              self.form.spacing_radio.clicked.connect(self.spacing_radio_clicked)
00098
00099
              self.form.customSpacing.clicked.connect(lambda: runRebarDistribution(self))
00100
              self.form.removeCustomSpacing.clicked.connect(lambda:
      removeRebarDistribution(self))
00101
              self.form.PickSelectedFace.clicked.connect(lambda: getSelectedFace(self))
00102
              self.form.orientation.currentIndexChanged.connect(self.getOrientation)
00103
              self.form.image.setPixmap(QtGui.QPixmap(os.path.split(os.path.abspath(__file__))[0] + "
      /icons/UShapeRebarBottom.svg"))
00104
              QtGui.QStyle.SP_DialogHelpButton))
              {\tt self.form.toolButton.clicked.connect(lambda: showPopUpImageDialog(os.path.split))} \\
00105
      (os.path.abspath(_file__))[0] + "/icons/UShapeRebarDetailed.svg"))
    self.Rebar = Rebar
00106
00107
00108
          def getOrientation(self):
              orientation = self.form.orientation.currentText()
if orientation == "Bottom":
00109
00110
                  self.form.image.setPixmap(QtGui.QPixmap(os.path.split(os.path.abspath(__file__))[0] + "
00111
      /icons/UShapeRebarBottom.svg"))
00112
             elif orientation == "Top":
                  self.form.image.setPixmap(QtGui.QPixmap(os.path.split(os.path.abspath(__file__))[0] + "
00113
      /icons/UShapeRebarTop.svg"))
    elif orientation == "Right":
00114
                  self.form.image.setpixmap(QtGui.QPixmap(os.path.split(os.path.abspath(__file__))[0] + "
00115
      /icons/UShapeRebarRight.svg"))
00116
              else:
                  self.form.image.setPixmap(QtGui.QPixmap(os.path.split(os.path.abspath(__file__))[0] + "
00117
      /icons/UShapeRebarLeft.svg"))
00118
          def getStandardButtons(self):
00119
00120
              return int(QtGui.QDialogButtonBox.Ok) | int(QtGui.QDialogButtonBox.Apply) | int(
      OtGui.ODialogButtonBox.Cancel)
00121
00122
          def clicked(self, button):
00123
              if button == int(QtGui.QDialogButtonBox.Apply):
00124
                  self.accept(button)
00125
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()
              b_cover = FreeCAD.Units.Quantity(b_cover).Value
00130
00131
              r cover = self.form.r sideCover.text()
              r_cover = FreeCAD.Units.Quantity(r_cover).Value
              1_cover = self.form.l_sideCover.text()
00133
00134
              1_cover = FreeCAD.Units.Quantity(l_cover).Value
00135
              t_cover = self.form.topCover.text()
              t_cover = FreeCAD.Units.Quantity(t_cover).Value
00136
00137
              diameter = self.form.diameter.text()
00138
              diameter = FreeCAD.Units.Quantity(diameter).Value
              rounding = self.form.rounding.value()
00139
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()
                       rebar = makeUShapeRebar(f_cover, b_cover, r_cover, l_cover, diameter,
00146
      t_cover, rounding, True, amount, orientation, self.SelectedObj, self.
      FaceName)
00147
                   elif spacing_check:
                       spacing = self.form.spacing.text()
spacing = FreeCAD.Units.Quantity(spacing).Value
00148
00149
00150
                       rebar = makeUShapeRebar(f_cover, b_cover, r_cover, l_cover, diameter,
      t_cover, rounding, False, spacing, orientation, self.SelectedObj, self.
      FaceName)
00151
              else:
00152
                  if amount check:
00153
                       amount = self.form.amount.value()
                       rebar = editUShapeRebar(self.Rebar, f_cover, b_cover, r_cover, l_cover,
       diameter, t_cover, rounding, True, amount, orientation, self.SelectedObj, self.
      FaceName)
00155
                  elif spacing_check:
                       spacing = self.form.spacing.text()
00156
```

```
00157
                                     spacing = FreeCAD.Units.Quantity(spacing).Value
                                     rebar = editUShapeRebar(self.Rebar, f_cover, b_cover, r_cover, l_cover,
00158
            diameter, t_cover, rounding, False, spacing, orientation, self.SelectedObj, self.
          FaceName)
00159
                       if self.CustomSpacing:
                              rebar.CustomSpacing = self.CustomSpacing
00160
                              FreeCAD.ActiveDocument.recompute()
00161
00162
                        self.Rebar = rebar
00163
                        if signal == int(QtGui.QDialogButtonBox.Apply):
00164
                       else:
00165
                              FreeCADGui.Control.closeDialog(self)
00166
00167
00168
                 def amount_radio_clicked(self):
00169
                        self.form.spacing.setEnabled(False)
00170
                        self.form.amount.setEnabled(True)
00171
00172
                 def spacing radio clicked(self):
                       self.form.amount.setEnabled(False)
00174
                       self.form.spacing.setEnabled(True)
00175
00176
00177 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):
00178
                       makeUShapeRebar(FrontCover, BottomCover, RightCover, LeftCover, Diameter, Topcover, Rounding,
            AmountSpacingCheck, AmountSpacingValue,
00179
                 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'.
00180
00181
00182
                 if not structure and not facename:
00183
                       selected_obj = FreeCADGui.Selection.getSelectionEx()[0]
                        structure = selected_obj.Object
00184
00185
                        facename = selected_obj.SubElementNames[0]
00186
                 face = structure.Shape.Faces[getFaceNumber(facename) - 1]
00187
                 #StructurePRM = getTrueParametersOfStructure(structure)
                 FacePRM = getParametersOfFace(structure, facename)
00188
                 if not FacePRM:
00189
00190
                       FreeCAD.Console.PrintError("Cannot identified shape or from which base object sturctural element is
            derived\n")
00191
00192
                 # Get points of U-Shape rebar
                 points = getpointsOfUShapeRebar(FacePRM, r_cover, l_cover, b_cover, t_cover,
00193
          orientation)
00194
                 import Part
00195
                 import Arch
00196
                 sketch = FreeCAD.activeDocument().addObject('Sketcher::SketchObject', 'Sketch')
                 sketch.MapMode = "FlatFace"
sketch.Support = [(structure, facename)]
00197
00198
00199
                 FreeCAD.ActiveDocument.recompute()
                 sketch.addGeometry(Part.LineSegment(points[0], points[1]), False)
                 sketch.addGeometry(Part.LineSegment(points[1], points[2]), False)
00201
                 import Sketcher
00202
00203
                 sketch.addGeometry(Part.LineSegment(points[2], points[3]), False)
00204
                 if amount_spacing_check:
00205
                       rebar = Arch.makeRebar(structure, sketch, diameter, amount spacing value, f cover)
00206
                        FreeCAD.ActiveDocument.recompute()
00207
                 else:
00208
                       \verb|size| = (ArchCommands.projectToVector(structure.Shape.copy(), face.normalAt(0, 0))). Length | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 0) | (0, 
00209
                       rebar = Arch.makeRebar(structure, sketch, diameter, int((size - diameter) / amount_spacing_value),
          f cover)
00210
                 rebar.Rounding = rounding
00211
                 # Adds properties to the rebar object
          rebar.ViewObject.addProperty("App::PropertyString", "RebarShape", "RebarDialog", QT_TRANSLATE_NOOP("
App::Property", "Shape of rebar")).RebarShape = "UShapeRebar"
    rebar.ViewObject.setEditorMode("RebarShape", 2)
    rebar.addProperty("App::PropertyDistance", "FrontCover", "RebarDialog", QT_TRANSLATE_NOOP("
App::Property", "Front cover of rebar")).FrontCover = f_cover
00212
00213
00214
                 rebar.setEditorMode("FrontCover", 2)
00215
00216
                 rebar.addProperty("App::PropertyDistance", "RightCover", "RebarDialog", QT_TRANSLATE_NOOP("
                                     "Right Side cover of rebar")).RightCover = r_cover
          App::Property",
00217
                 rebar.setEditorMode("RightCover", 2)
               rebar.addProperty("App::PropertyDistance", "LeftCover", "RebarDialog", QT_TRANSLATE_NOOP("App::Property "Left Side cover of rebar")).LeftCover = l_cover
00218
00219
                 rebar.setEditorMode("LeftCover", 2)
                 rebar.addProperty("App::PropertyDistance", "BottomCover", "RebarDialog", QT_TRANSLATE_NOOP("
          App::Property", "Bottom cover of rebar")).BottomCover = b_cover
                 rebar.setEditorMode("BottomCover", 2) rebar.addProperty("App::PropertyBool", "AmountCheck", "RebarDialog", QT_TRANSLATE_NOOP("App::Property",
00221
00222
             "Amount radio button is checked")). AmountCheck
                rebar.setEditorMode("AmountCheck", 2)
00223
00224
                 rebar.addProperty("App::PropertyDistance", "TopCover", "RebarDialog", QT_TRANSLATE_NOOP("App::Property"
              "Top cover of rebar")).TopCover = t_cover rebar.setEditorMode("TopCover", 2)
00225
          rebar.addProperty("App::PropertyDistance", "TrueSpacing", "RebarDialog", QT_TRANSLATE_NOOP("
App::Property", "Spacing between of rebars")).TrueSpacing = amount_spacing_value
    rebar.setEditorMode("TrueSpacing", 2)
00226
```

8.22 UShapeRebar.py 143

```
00228
          rebar.addProperty("App::PropertyString", "Orientation", "RebarDialog", QT_TRANSLATE_NOOP("App::Property
      ", "Shape of rebar")).Orientation = orientation
00229
          rebar.setEditorMode("Orientation", 2)
00230
          if amount_spacing_check:
00231
             rebar.AmountCheck = True
00232
          else:
             rebar.AmountCheck = False
              rebar.TrueSpacing = amount_spacing_value
00234
00235
          rebar.Label = "UShapeRebar"
00236
          FreeCAD.ActiveDocument.recompute()
00237
         return rebar
00238
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.
          if not sketch.Support:
00244
00245
              showWarning("You have checked remove external geometry of base sketchs when needed.\nTo
       unchecked Edit->Preferences->Arch.")
00246
              return
00247
          # Assigned values
          facename = sketch.Support[0][1][0]
00248
00249
          structure = sketch.Support[0][0]
          face = structure.Shape.Faces[getFaceNumber(facename) - 1]
00250
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()
          sketch.movePoint(1, 1, points[1], 0)
FreeCAD.ActiveDocument.recompute()
00260
00261
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
              FreeCAD.ActiveDocument.recompute()
00272
00273
              Rebar.AmountCheck = True
00274
          else:
00275
              00276
              Rebar.Amount = int((size - diameter) / amount_spacing_value)
00277
              FreeCAD.ActiveDocument.recompute()
00278
              Rebar.AmountCheck = False
00279
          Rebar.Diameter = diameter
          Rebar.FrontCover = f_cover
00280
          Rebar.RightCover = r_cover
00281
00282
          Rebar.LeftCover = 1_cover
00283
          Rebar.BottomCover = b_cover
          Rebar.TopCover = t_cover
Rebar.Rounding = rounding
00284
00285
00286
          Rebar.TrueSpacing = amount_spacing_value
          Rebar.Orientation = orientation
00287
00288
          FreeCAD.ActiveDocument.recompute()
00289
          return Rebar
00290
00291 def editDialog(vobi):
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
             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
          #obi.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())
```