

---

# **Mechatronic**

***Release 0.2.1***

**David Muñoz Bernal**

**Dec 02, 2020**



# CONTENTS

<b>1</b>	<b>Table Of Contents</b>	<b>3</b>
1.1	Introduccion . . . . .	3
1.2	Install . . . . .	4
1.3	Tutorial . . . . .	4
1.4	Wiki . . . . .	14
<b>2</b>	<b>Indices and tables</b>	<b>145</b>
	<b>Python Module Index</b>	<b>147</b>
	<b>Index</b>	<b>149</b>



On this page you can find all the information about the Mechatronic Workbench. You have some tutorials where you can learn how to use it, as well as the documentation of all the 3D models and the functions that have been designed.



## TABLE OF CONTENTS

### 1.1 Introduccion

#### 1.1.1 Mechatronic

Mechatronic is a Workbench for FreeCAD that allows the modification of parametric models that includes and simplify the assembly and composition. It also provides a library of functions with which to generate new parametric models to be included in the Workbench.

#### 1.1.2 How it works

Mechatronic is designed for two different users:

**1. Basic User:** User without CAD or Programming knowledge who needs to make a design. For this user is the graphic part. The steps to follow would be:

1. Select the part
2. Enter the values you want to use
3. You already have the part you want.

---

**Note:** See the *Tutorial* section for more details

---

Additionally, this user may want to combine parts with each other. For this purpose, there is the *Assembly* module<sup>1</sup> that allows the placement of some pieces with respect to others

**2. Advanced user:** User with programming knowledge who wants to design parameterizable 3D models. This user has the *Functions Library*<sup>2</sup> to make 3D models in a simple way. You can consult the class design (*UML*) if you wish to better understand the operation

---

<sup>1</sup> The Assembly module will be upgraded

<sup>2</sup> The functions library is in process

### 1.1.3 History

Mechatronic Workbench started with the Filter Stage project. In this project, we designed a support for the sample holder of a microscope to the URJC. In order to be able to modify the design and adapt it to the required dimensions, we chose to make a parametric design. Parametric design requires the use of a programming language to describe the model, in this case we use Python.

Based on this parameterizable design, a final degree work was conceived to create a Workbench where the Filter Stage parameters could be modified from the FreeCAD interface.

It was decided to improve this first Workbench by adding designs commonly used in mechatronic systems and functions to facilitate the placement of these designs

## 1.2 Install

To install the Mechatronic Workbench you need [FreeCAD](#).

---

**Note:** Works on FreeCAD 0.19

---

After installing the program, there is two ways to install the Workbench:

1. **Use the addon manager.** In FreeCAD, go to the menu *Tools/Addon manager*. In the Addon Manager choose *Configuration* from the upper right side. In the new window add <https://github.com/davidmubernal/Mechatronic> to the Custom Repositories
2. Download Mechatronic Workbench from the following [file](#). Take the *Mechatronic* folder from the *Mechatronic.zip* file and put it in the *Mod* folder inside the *FreeCAD* installation folder. The default path of *FreeCAD* is:

S.O.	Folder
WINDOWS	<i>C:/Program Files/FreeCAD 0.19/Mod</i>
MAC	<i>/Applications/FreeCAD 0.19/Mod</i>

After completing these steps, the Workbench will be listed in the FreeCAD Workbench

## 1.3 Tutorial

---

**Note:** Work in progress

---



### 1.3.1 YouTube Tutorials in Spanish

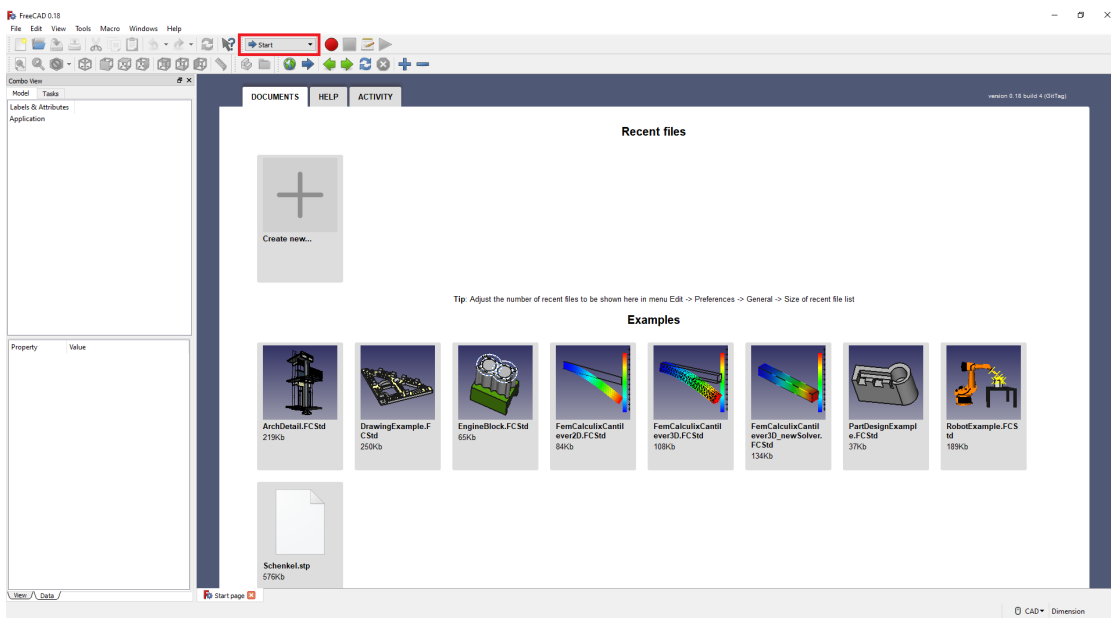
There are diferents tutorials availables in [Youtube](#).

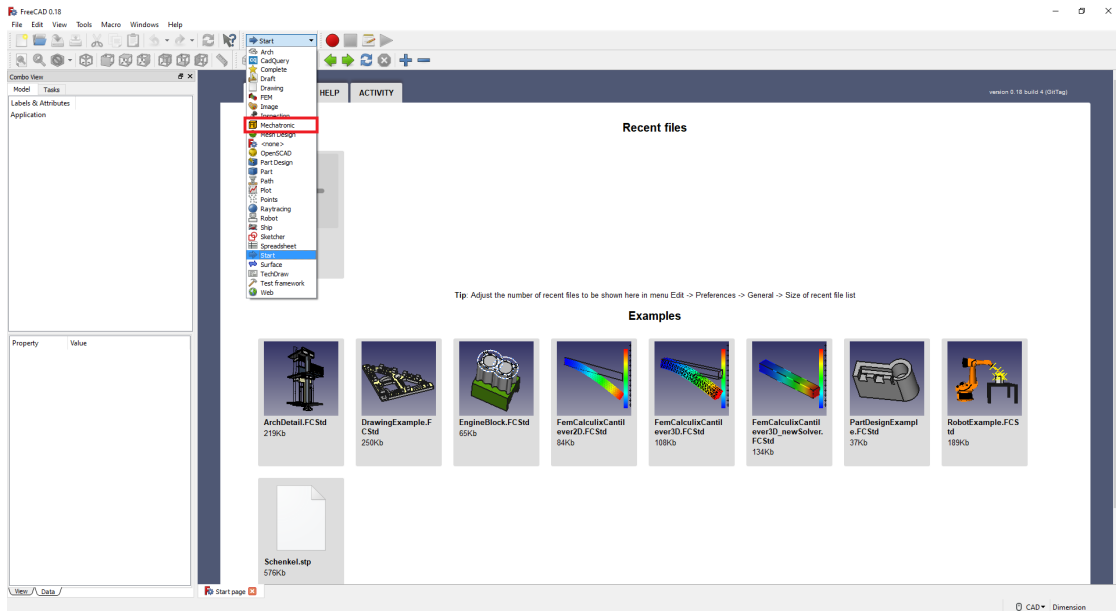
**Warning:** We have update the workbench but we have not yet updated the followings tutorials

### 1.3.2 Tutorial CAD 1 - Create part

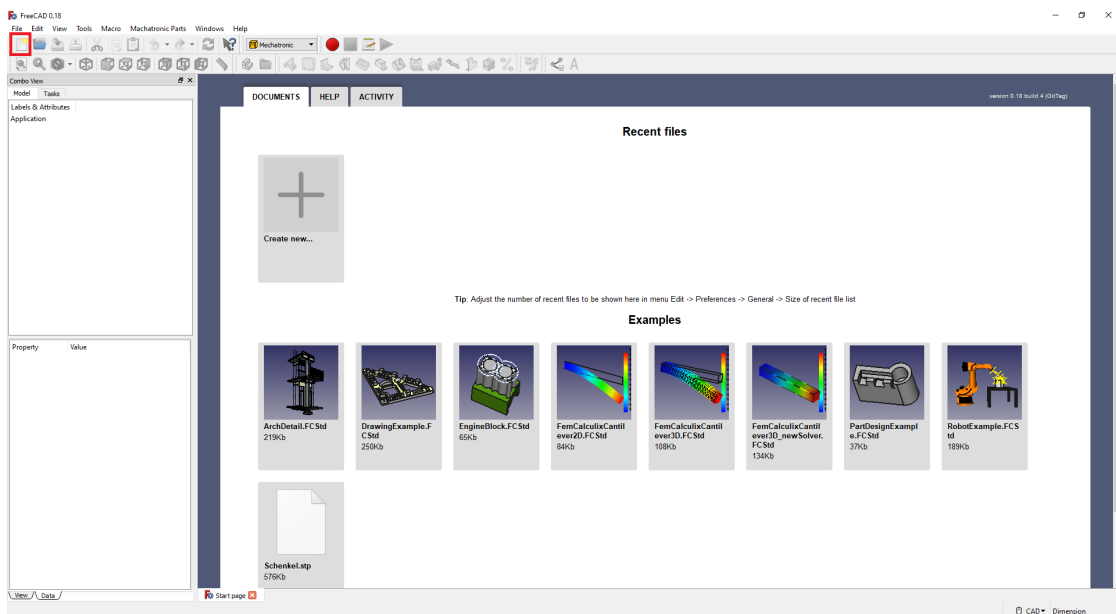
1- After opening FreeCAD, the first step is to select the Mechatronic workbench. To do this, click from the drop-down menu on the top bar and select Mechatronic.

**Note:** If Mechatronic does not appear, check that you have followed the installation steps correctly

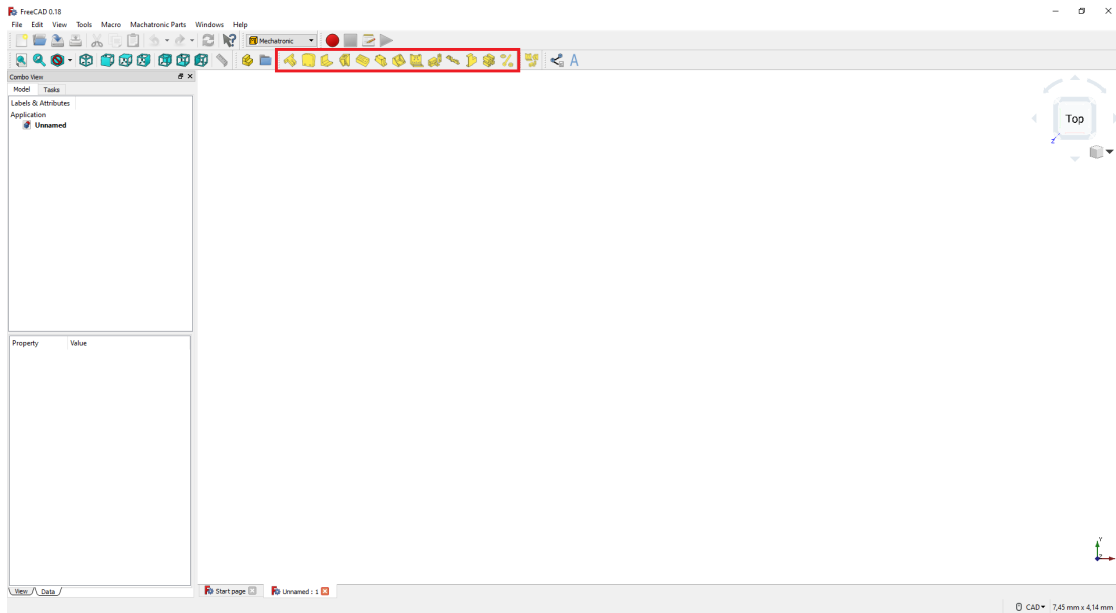




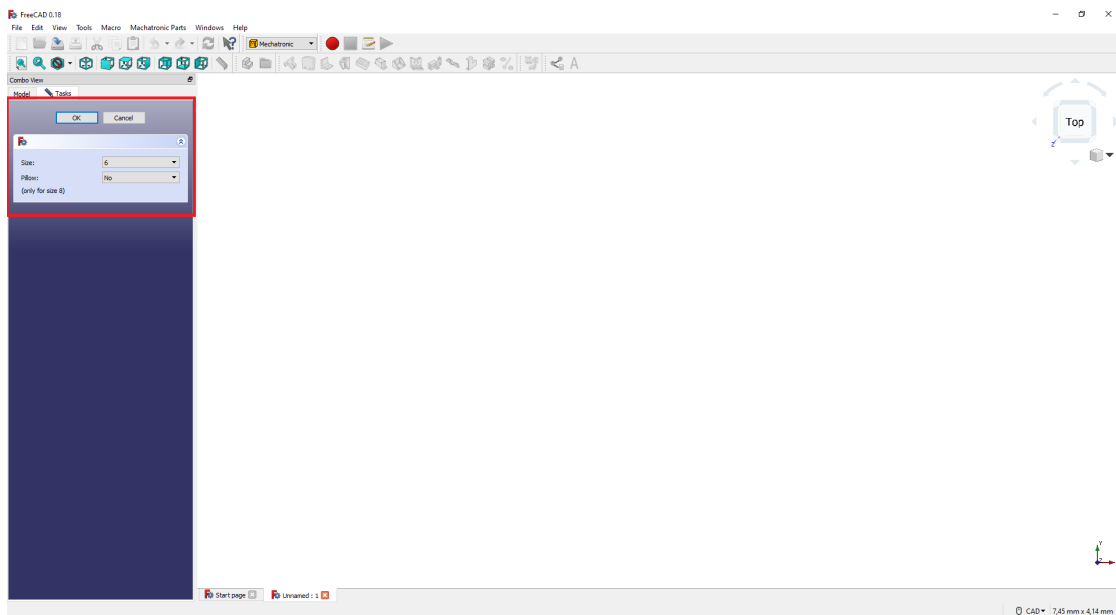
2- Once you have selected the Mechatronic workbench, open a new document, if you have not done so before.



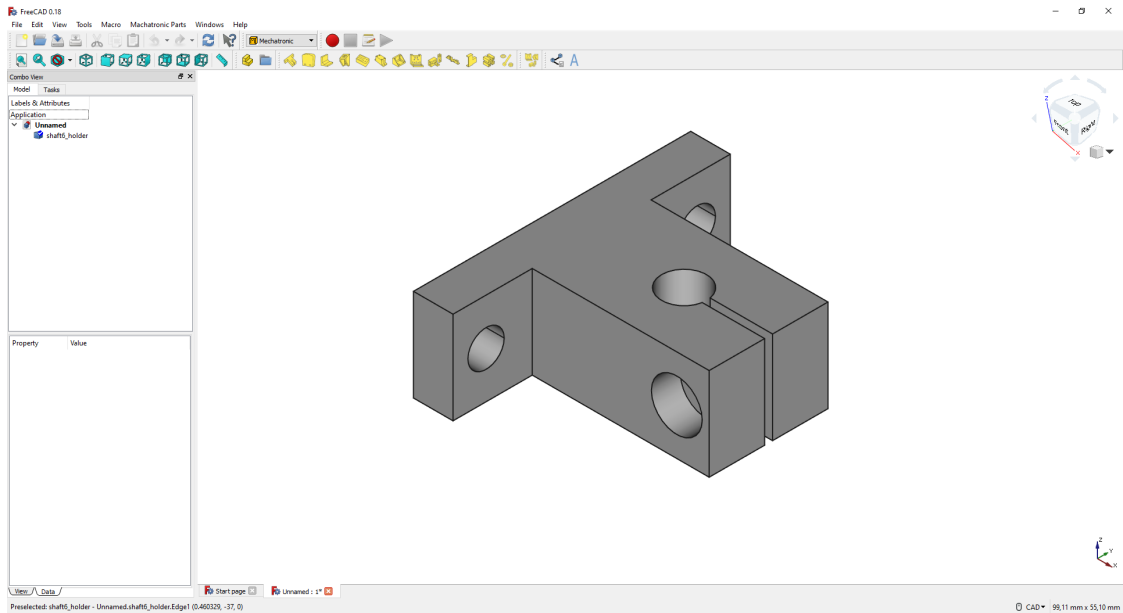
3- When you have selected the Mechatronic tool bank, a set of icons should appear at the top. Select one of the models to make your first part. You can also select the *Mechatronic Parts* menu to view the models available.



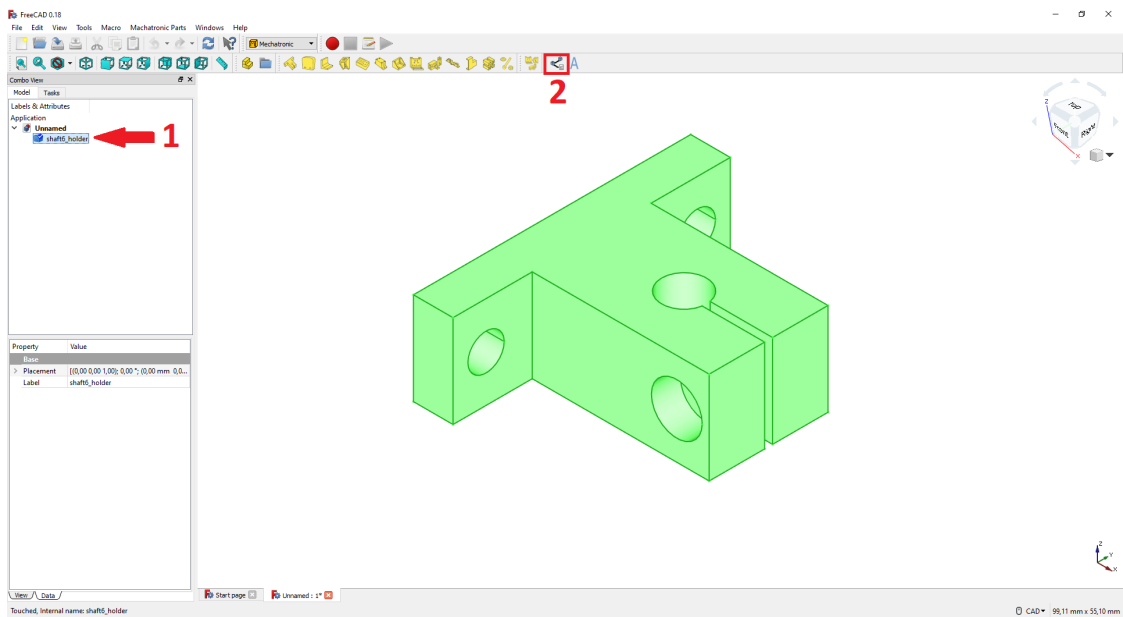
4- When one of the models is selected, the options to modify it will appear in the *Tasks* tab. Enter or select the values you want for the model. When you have finished, select *OK* to create the model.



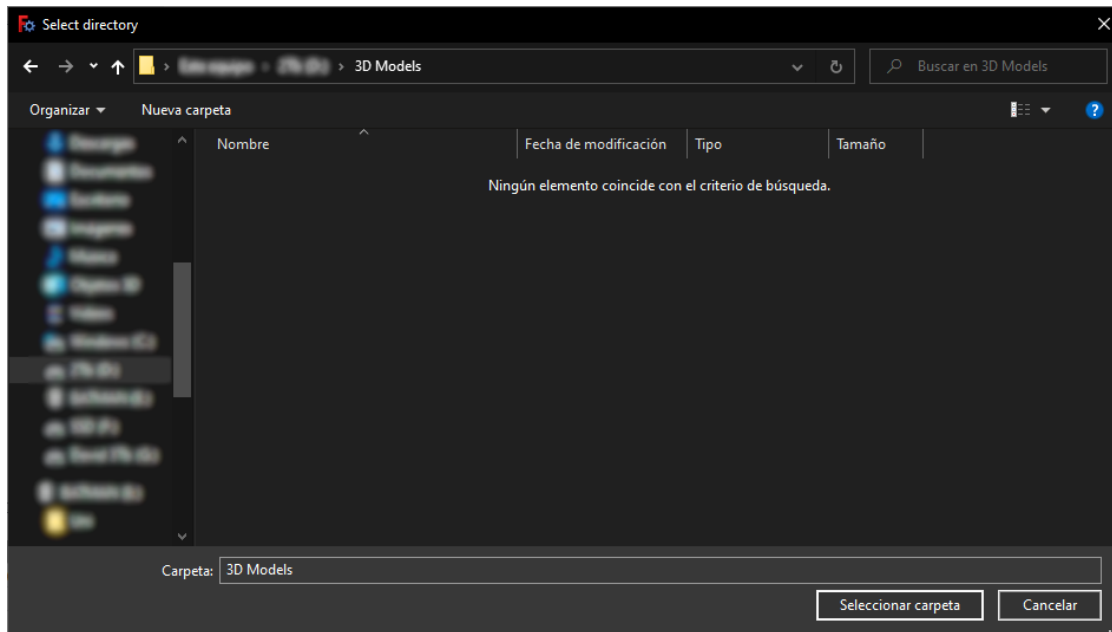
5- The model will be displayed with the options selected.



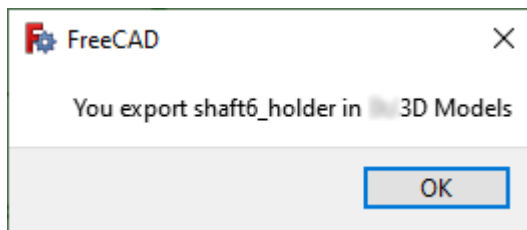
6- If you want to print the model, select it and then select the icon to export in STL format. This function also optimizes the orientation of the model for 3D printing



- A new window will be displayed where you can select the folder to save the model



- The model will be saved with the name and in the folder shown in the



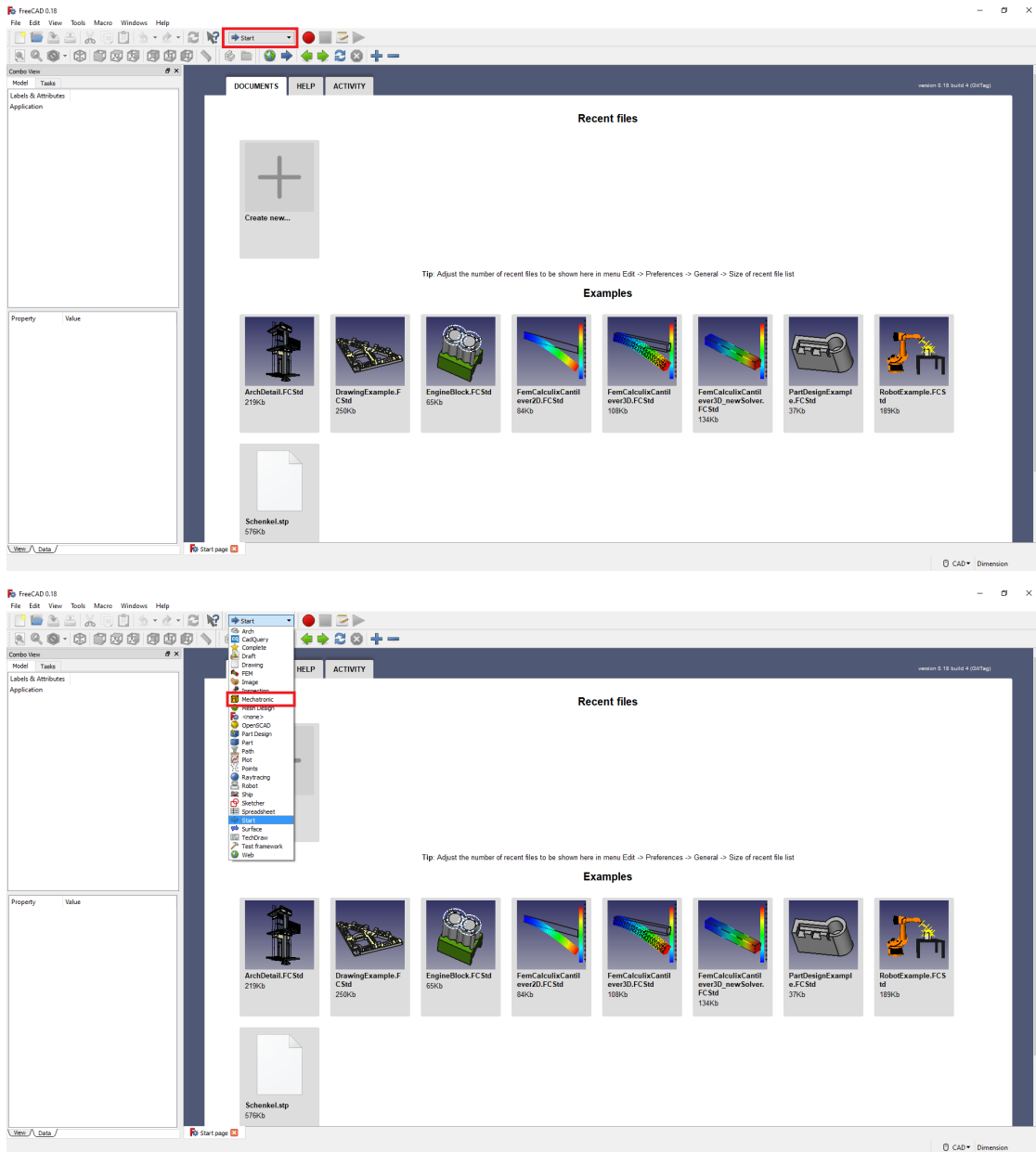
### 1.3.3 Tutorial CAD 2 - Crate a system

**1-** After opening FreeCAD, the first step is to select the Mechatronic workbench. To do this, click from the drop-down menu on the top bar and select Mechatronic.

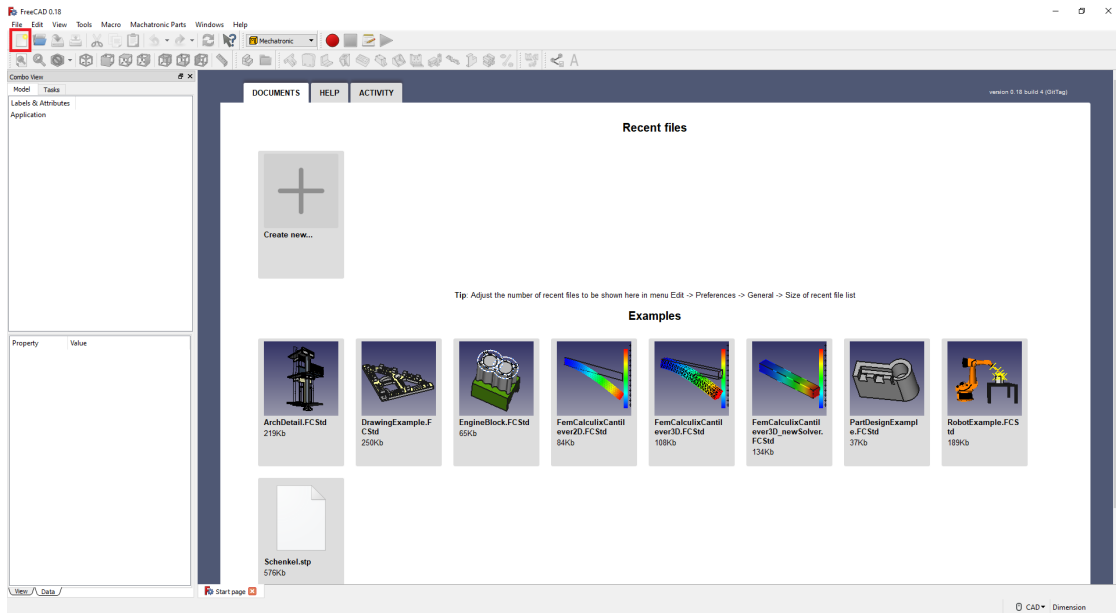
---

**Note:** If Mechatronic does not appear, check that you have followed the installation steps correctly

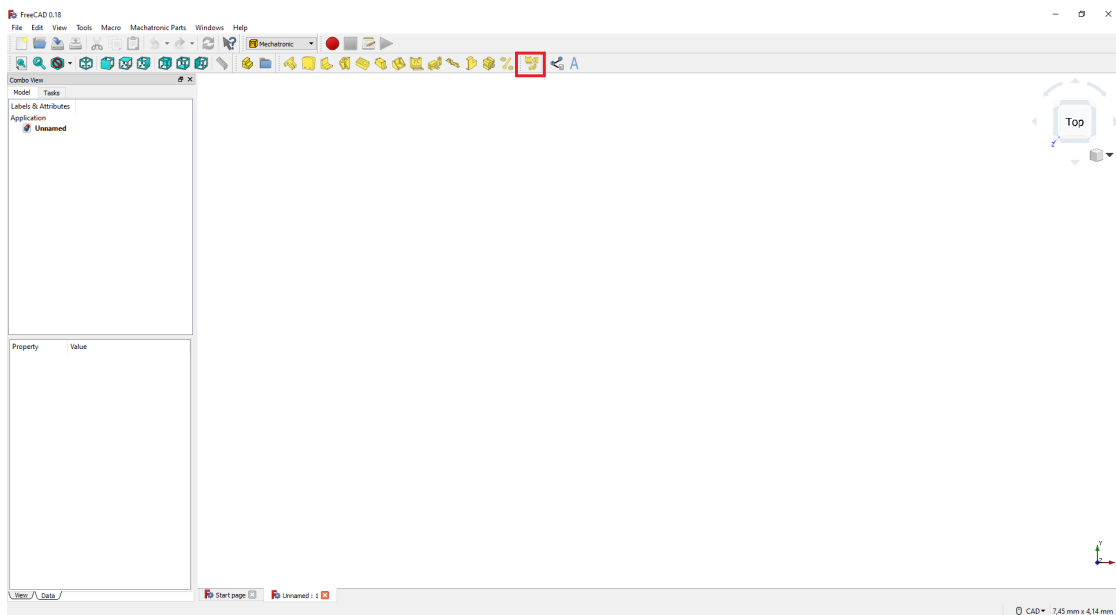
---



2- Once you have selected the Mechatronic workbench, open a new document, if you have not done so before.



3- When you have selected the Mechatronic tool bank, a set of icons should appear at the top. Select one of the systems to make your first part. You can also select the *Mechatronic Systems* menu to view the systems available.

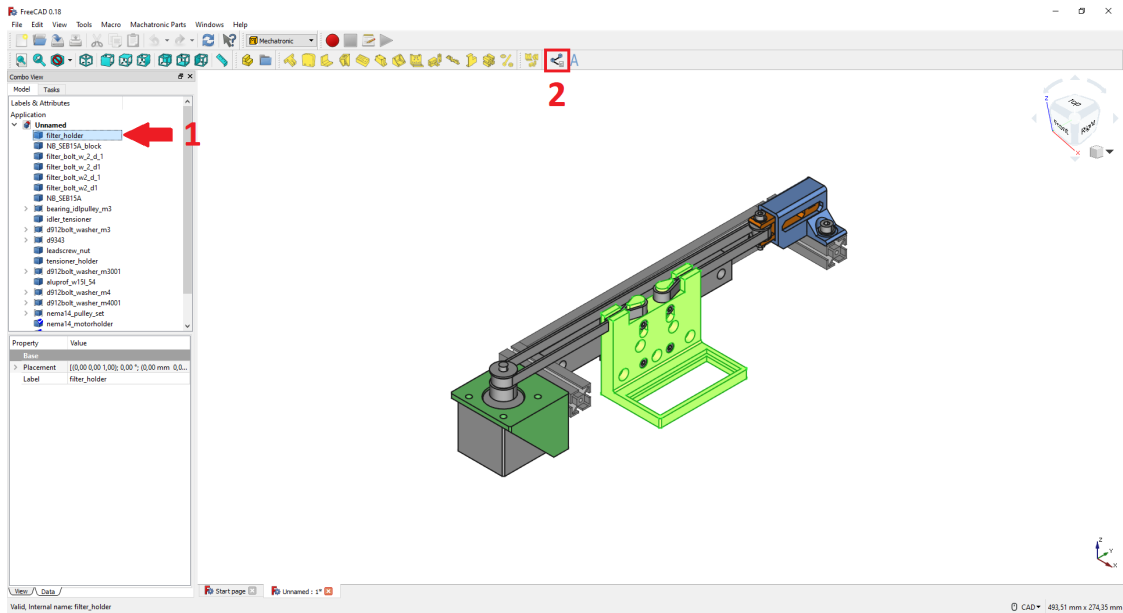


4- When one of the systems is selected, the options to modify it will appear in the *Tasks* tab. Enter or select the values you want for the system. When you have finished, select *OK* to create the system.

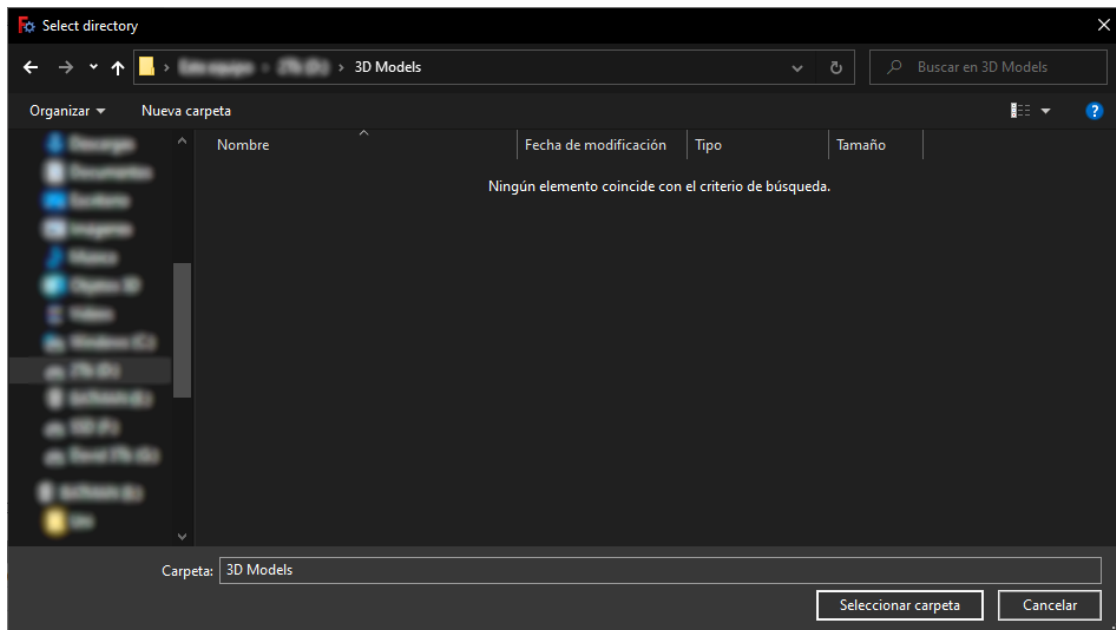


**Note:** This may take some time according to your computer hardware

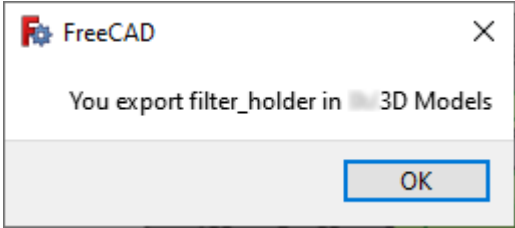




- A new window will be displayed where you can select the folder to save the model



- The model will be saved with the name and in the folder shown in the



1.4 Wiki

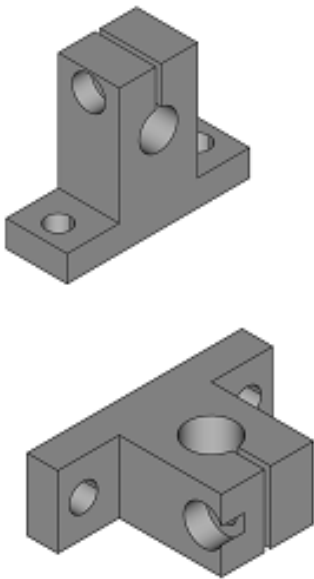
**Note:** This is a basic view of the Wiki

1.4.1 3D Model library

Mechatronic

Shaft Holder

- Size
- Low profile: Only in size 8



Details

<code>Sk_dir(size[, fc_axis_h, fc_axis_d, ...])</code>	SK dimensions: dictionary for the dimensions .
--	--

## Idler Holder

- Size of the profile on which it is mounted
- Bolt metrics
- Height
- Position of the limit switch sensor
- Height of the limit switch sensor

The model will be modified **for** greater efficiency

### Details

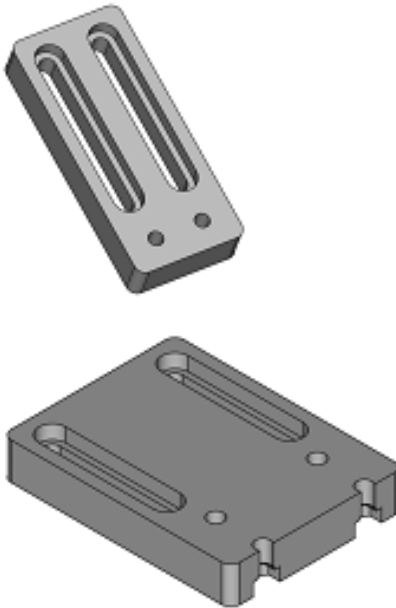
---

*IdlePulleyHolder*(profile\_size, pulleybolt\_d, ...) Creates a holder for a IdlePulley.

---

## Limit Switches Holder

- Type
- Rail distance



### Details

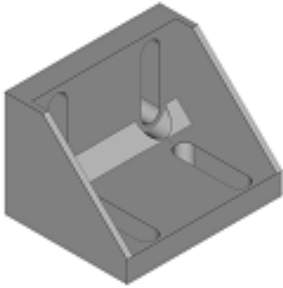
---

*SimpleEndstopHolder*(d\_endstop[, rail\_l, ...]) Very simple endstop holder to be attached to a alu profile and that can be adjusted .

---

## **Hall stop**

- Width
- Thickness
- Metric nut
- Profile size
- Reinforce



### Details

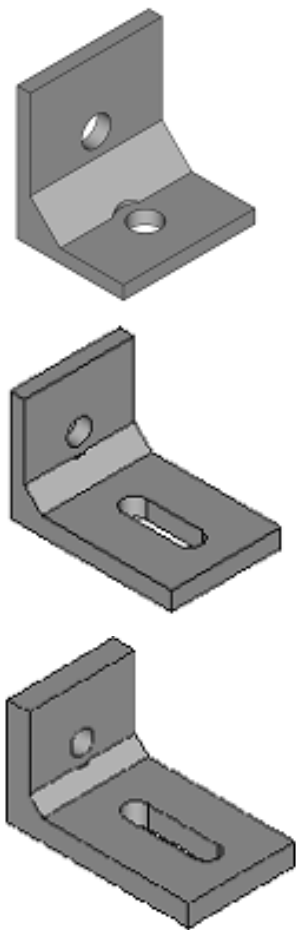
---

```
hallestopholder([stp_w, stp_h, base_thick,  
...])
```

---

## **Bracket**

- Type: 3 options
- Size first profile
- Size second profile
- Thickness
- Metric nut first profile
- Metric nut second profile
- Number of nuts
- Distance between nuts
- Type of hole
- Reinforcement: first type only
- Flap: second type only
- Distance between profiles: third type only

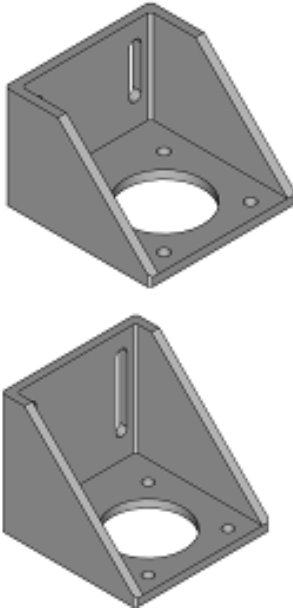


Details

<i>AluProfBracketPerp</i> (alusize_lin, alusize_perp)		Bracket to join 2 aluminum profiles that are perpendicular, that is, they are not on the same plane .
<i>AluProfBracketPerpFlap</i> (alusize_lin, alu-size_perp)	alu-	Bracket to join 2 aluminum profiles that are perpendicular, that is, they are not on the same plane It is wide because it has 2 ears/flaps? on the sides, to attach to the perpendicular profile .
<i>AluProfBracketPerpTwin</i> (alusize_lin, ...[, ...])		Bracket to join 3 aluminum profiles that are perpendicular, that is, they are not on the same plane to the perpendicular profile .

## Motor holder

- Size
- Height
- Thickness



### Details

---

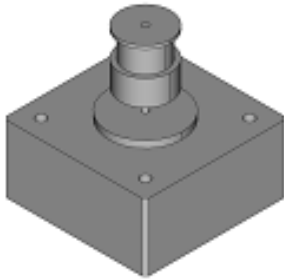
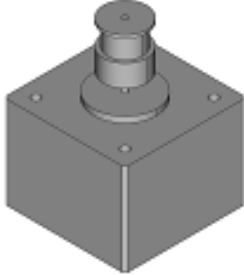
<code>NemaMotorHolder([nema_size, wall_thick, ...])</code>	Creates a holder for a Nema motor
--	-----------------------------------

---

## Motor

- Size
- Height
- Shaft height
- Shaft radius
- Shaft radius base
- Shaft height base
- Chamfer radius
- Bolt deep
- Bolt outside
- Pulley pitch
- Pulley teeth
- Pulley top flange
- Pulley bot flange

- Position in axis d
- Position in axis w
- Position in axis h
- Placement



Details

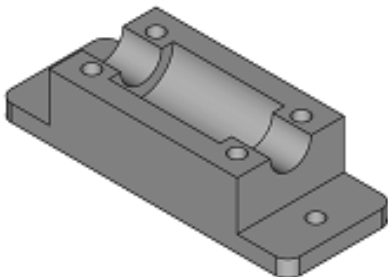
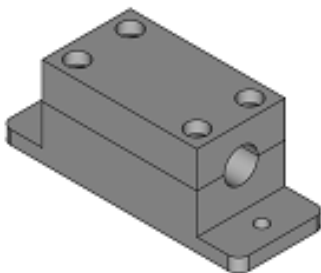
---

<i>NemaMotorPulleySet</i> ([nema_size, base_1, ...])	Set composed of a Nema Motor and a pulley
--	---

---

### Lin bear house

- Type

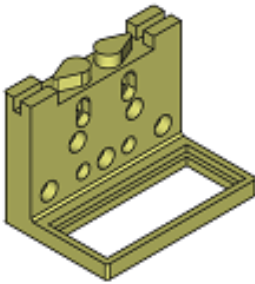


## Details

<code>ThinLinBearHouse1rail(d_lbear[, ...])</code>	Makes a housing for a linear bearing, but it is very thin and intended to be attached to one rail, instead of 2 it has to parts, the lower and the upper part .
<code>ThinLinBearHouse(d_lbear[, fc_slide_axis, ...])</code>	Makes a housing for a linear bearing, but it is very thin and intended to be attached to 2 rail it has to parts, the lower and the upper part .
<code>LinBearHouse(d_lbearhousing[, ...])</code>	Makes a housing for a linear bearing takes the dimensions from a dictionary, like the one defined in kcomp.py it has to parts, the lower and the upper part .
<code>ThinLinBearHouseAsim(d_lbear[, fc_fro_ax, ...])</code>	There are

## Filter holder

- Length
- Width



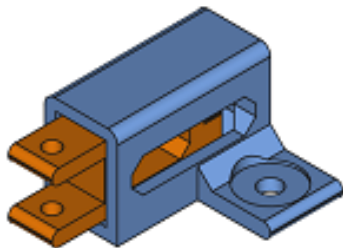
## Details

<code>PartFilterHolder([filter_l, filter_w, ...])</code>	Integration of a ShpFilterHolder object into a PartFilterHolder object, so it is a FreeCAD object that can be visualized in FreeCAD
--	---

## Tensioner

- Belt hight
- Base width
- Thickness
- Metric nut



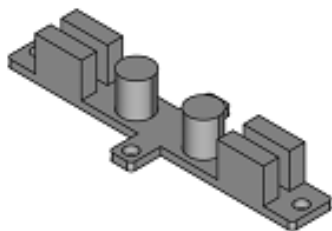
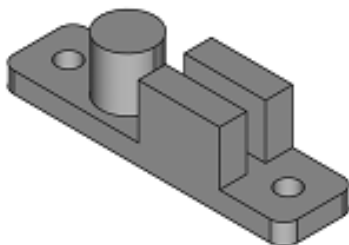


Details

<i>TensionerSet</i> ([aluprof_w, belt_pos_h, ...])	Set composed of the idler pulley and the tensioner
--	--

Belt clamp

- Type
- Length
- Width
- Metric nut

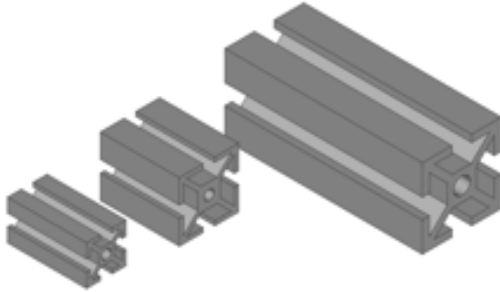


Details

<i>BeltClamp</i> (fc_fro_ax, fc_top_ax[, base_h, ...])	Similar to shp_topbeltclamp, but with any direction, and can have a base Creates a shape of a belt clamp.
<i>DoubleBeltClamp</i> ([axis_h, axis_d, axis_w, ...])	Similar to BeltClamp, but in two ways Creates a shape of a double belt clamp.

## Aluminium profile

- Section
- Length



### Details

---

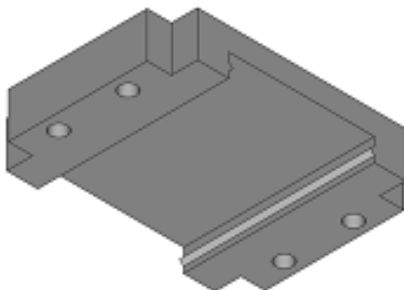
*PartAluProf*(depth, aluprof\_dict[, xtr\_d, ...])

Integration of a ShpAluProf object into a PartAluProf object, so it is a FreeCAD object that can be visualized in FreeCAD. Instead of using all the arguments of ShpAluProf, it will use a dictionary

---

## Linear Guide

- Type:
  - SEBW16
  - SEB15A
  - SEB8
  - SEB10
- Position in axis d
- Position in axis w
- Position in axis h
- Placement



### Details

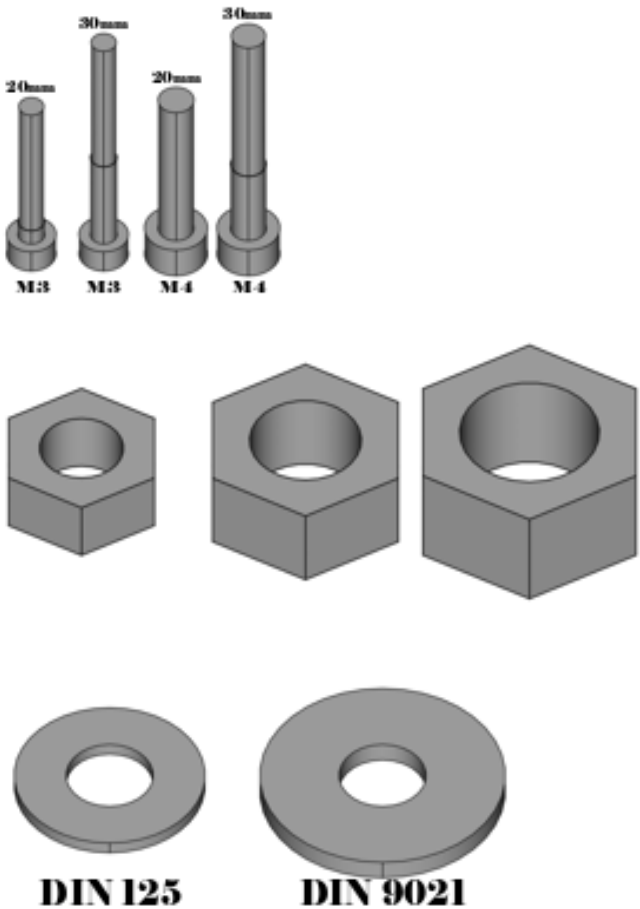
---

<code>PartLinGuideBlock(block_dict, rail_dict[, ...])</code>	Integration of a ShpLinGuideBlock object into a PartLinGuideBlock object, so it is a FreeCAD object that can be visualized in FreeCAD. Instead of using all the arguments of ShpLinGuideBlock, it will use a dictionary
--	---

---

**Bolts, Nuts & Washers**

- Type
- Metric
- Bolt length



Details

---

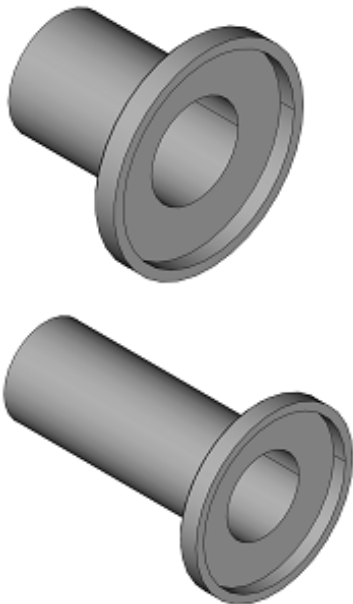
<code>Din934Nut(metric[, axis_d_apo, h_offset, ...])</code>	Din 934 Nut
<code>Din125Washer(metric, axis_h, pos_h[, tol, ...])</code>	Din 125 Washer, this is the regular washer
<code>Din9021Washer(metric, axis_h, pos_h[, tol, ...])</code>	Din 9021 Washer, this is the larger washer
<code>Din912Bolt(metric, shank_l[, ...])</code>	Din 912 bolt.

---

Optical

TubeLens

- Length
- Placement

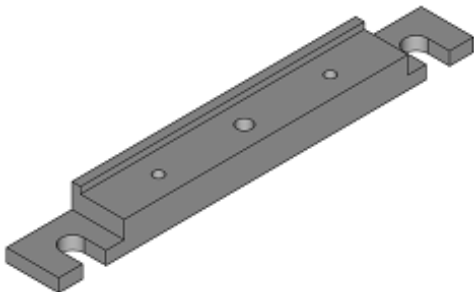


Details

<i>SM1TubeLensSm2</i> (sm1l_size[, fc_axis, ...])	Creates a componente formed by joining: the lens tube SM1LXX + SM1A2 + SM2T2, so we have:
---	---

LCPB1M Base

- Placement

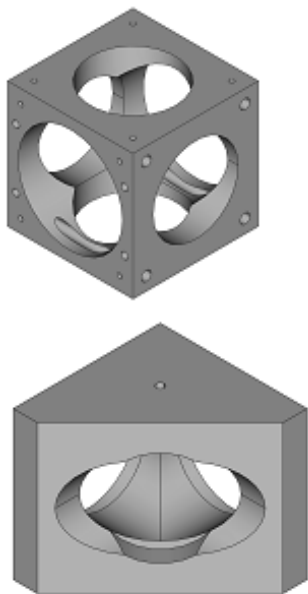


Details

<i>lcpb1m_base</i> ([d_lcpb1m_base, fc_axis_d, ...])	Creates a lcpb1m_base for plates side, it creates from a dictionary
--	---

CageCube

- Type:
  - CageCube
  - CageCubeHalf

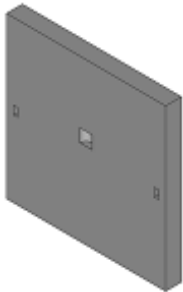


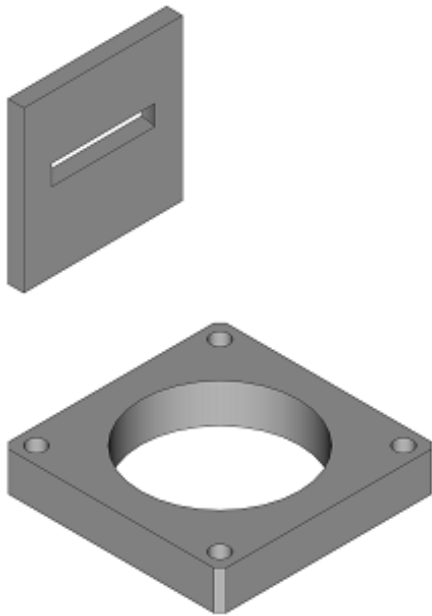
Details

<i>f_cagecube</i> (d_cagecube[, axis_thru_rods, ...])	Creates a cage cube, it creates from a dictionary
<i>f_cagecubehalf</i> (d_cagecubehalf[, axis_1, ...])	Dreates a half cage cube: 2 perpendicular sides, and a 45 degree angle side.

Plate

- Plate dictionary:
  - Lb1cm\_Plate
  - Lb2c\_Plate
  - Lcp01m\_plate
- Placement



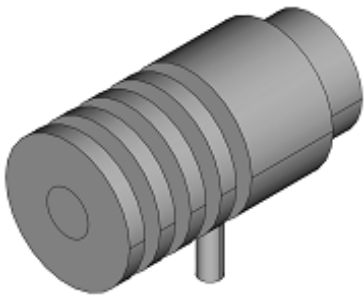


Details

<code>Lb1cPlate(d_plate[, fc_axis_h, fc_axis_l, ...])</code>	Creates a LB1C/M plate from thorlabs. The plate is centered
<code>Lb2cPlate(fc_axis_h, fc_axis_l[, cl, cw, ...])</code>	Same as plate_lb2c, but it creates an object.
<code>lcp01m_plate([d_lcp01m_plate, fc_axis_h, ...])</code>	Creates a lcp01m_plate side.

ThLed30

- Placement

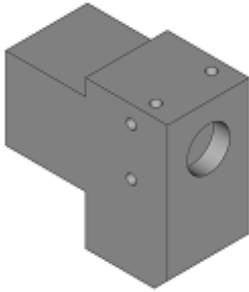


Details

<code>ThLed30([fc_axis, fc_axis_cable, pos, name])</code>	Creates the shape of a Thorlabs Led with 30.5 mm Heat Sink diameter The drawing is very rough .
---	---

## PrizLed

- Placement



### Details

---

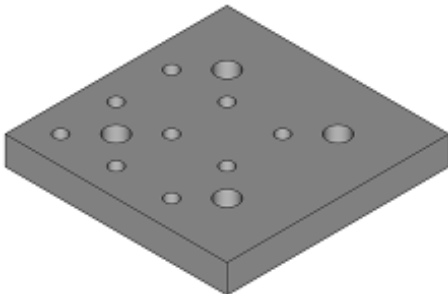
*PrizLed*([fc\_axis\_led, fc\_axis\_clear, pos, name])

Creates the shape of a Prizmatix UHP-T-Led The drawing is very rough, and the original drawing lacks many dimensions .

---

## BreadBoard

- Length
- Width
- Placement



### Details

---

*f\_breadboard*(d\_breadboard, length, width[, ...])

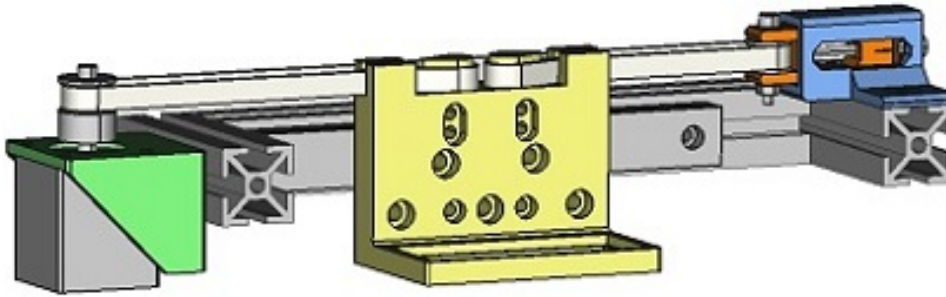
**param d\_breadboard** Dictionary with the values

---

## 1.4.2 Systems library

### Filter Stage

- Move distance
- Filter length
- Filter width
- Base width
- Tensioner stroke
- Tensioner thickness
- Metric nut
- Motor size
- Length rail motor holder
- Motor holder thickness



## 1.4.3 Functions Library

### fcfun

<i>NutHole</i> (nut_r, nut_h, hole_h, name[, extra, ...])	Adding a Nut hole (hexagonal) with a prism attached to introduce the nut.
<i>add2CylsHole</i> (r1, h1, r2, h2, thick[, ...])	Creates a piece formed by 2 hollow cylinders
<i>add3CylsHole</i> (r1, h1, r2, h2, rring, hring, thick)	Creates a piece formed by 2 hollow cylinders, and a ring on the side of the larger cylinder
<i>addBolt</i> (r_shank, l_bolt, r_head, l_head[, ...])	Creates the hole for the bolt shank and the head or the nut Tolerances have to be included
<i>addBoltNut_hole</i> (r_shank, l_bolt, r_head, ...)	Creates the hole for the bolt shank, the head and the nut.
<i>addBox</i> (x, y, z, name[, cx, cy])	Adds a box, centered on the specified axis x and/or y, with its Placement and Rotation at zero.
<i>addBox_cen</i> (x, y, z, name[, cx, cy, cz])	Adds a box, centered on the specified axis, with its Placement and Rotation at zero.
<i>addCyl</i> (r, h, name)	Add cylinder
<i>addCylHole</i> (r_ext, r_int, h, name[, axis, h_disp])	Add cylinder, with inner hole:
<i>addCylHolePos</i> (r_out, r_in, h, name[, ...])	Same as addCylHole, but avoiding the creation of many FreeCAD objects

continues on next page



Table 22 – continued from previous page

<code>addCylPos(r, h, name[, normal, pos])</code>	Same as <code>addCyl_pos</code> , but avoiding the creation of many FreeCAD objects
<code>addCyl_pos(r, h, name[, axis, h_disp])</code>	Add cylinder in a position.
<code>add_fcobj(shp, name[, doc])</code>	Just creates a freeCAD object of the shape, just to save one line
<code>aluprof_vec(width, thick, slot, insquare)</code>	Creates a wire (shape), that is an approximation of a generic alum profile extrusion .
<code>calc_desp_ncen(Length, Width, Height, vec1, vec2)</code>	Similar to <code>calc_rot</code> , but calculates de displacement, when we don't want to have all of the dimensions centered First vector original direction (x,y,z) is (1,0,0) Second vector original direction (x,y,z) is (0,0,-1) The arguments <code>vec1</code> , <code>vec2</code> are tuples (x,y,z) but they may be also FreeCAD.Vectors .
<code>calc_rot(vec1, vec2)</code>	Having an object with an orientation defined by 2 vectors the vectors a tuples, nor FreeCAD.Vectors use the wrapper <code>fc_calc_rot</code> to have FreeCAD.Vector arguments First vector original direction (x,y,z) is (1,0,0) Second vector original direction (x,y,z) is (0,0,-1) we want to rotate the object in an ortogonal direction.
<code>calc_rot_z(v_refz, v_refx)</code>	Calculates de rotation like <code>calc_rot</code> .
<code>edgeonaxis(edge, axis)</code>	It tells if an edge is on an axis
<code>equ(x, y)</code>	Compare numbers that are the same but not exactly the same
<code>fc_calc_desp_ncen(Length, Width, Height, ...)</code>	Same as <code>calc_desp_ncen</code> but using FreeCAD.Vectors arguments
<code>fc_calc_rot(fc_vec1, fc_vec2)</code>	Same as <code>calc_rot</code> but using FreeCAD.Vectors arguments
<code>fc_isonbase(fcv)</code>	Just tells if a vector has 2 of the coordinates zero so it is on just a base vector
<code>fc_isparal(fc1, fc2)</code>	Return 1 if <code>fc1</code> and <code>fc2</code> are paralell (colinear), 0 if they are not
<code>fc_isparal_nrm(fc1, fc2)</code>	Very similar to <code>fc_isparal</code> , but in this case the arguments are normalized so, less operations to do.
<code>fc_isperp(fc1, fc2)</code>	Return 1 if <code>fc1</code> and <code>fc2</code> are perpendicular, 0 if they are not
<code>fillet_len(box, e_len, radius, name)</code>	Make a new object with fillet
<code>filletchamfer(fco, e_len, name[, fillet, ...])</code>	Fillet or chamfer edges of a certain length, on a certain axis and a certain coordinate
<code>fuseshplist(shp_list)</code>	Since multifuse methods needs to be done by a shape and a list, and usually I have a list that I want to fuse, I make this function to save the inconvenience of doing everytime what I will do here Fuse multiFuse
<code>get_bolt_bearing_sep(bolt_d, hasnut, lbearing_r)</code>	same as <code>get_bolt_end_sep</code> , but when there is a bearing.
<code>get_bolt_end_sep(bolt_d, hasnut[, sep])</code>	Calculate Bolt separation
<code>get_fc_perpend1(fcv)</code>	gets a 'random' perpendicular FreeCAD.Vector
<code>get_fc_list_4perp2_fcvec(fcvec)</code>	Gets a list of 4 FreCAD.Vector perpendicular to one base vector <code>fcvec</code> can only be: * (1,0,0) * (0,1,0) * (0,0,1) * (-1,0,0) * (0,-1,0) * (0,0,-1)
<code>get_fc_list_4perp2_vecname(vecname)</code>	Gets a list of 4 FreCAD.Vector perpendicular to one vecname different from <code>get_fc_list_4perp_vecname</code> For example: .

continues on next page

Table 22 – continued from previous page

<code>get_fclist_4perp_fcvec(fcvec)</code>	Gets a list of 4 FreeCAD.Vector perpendicular to one base vector fcvec can only be: * (1,0,0) * (0,1,0) * (0,0,1) * (-1,0,0) * (0,-1,0) * (0,0,-1)
<code>get_fclist_4perp_vecname(vecname)</code>	Gets a list of 4 FreeCAD.Vector perpendicular to one vecname for example: .
<code>get_fcvector(tup)</code>	Gets the FreeCAD.Vector of a tuple
<code>get_nameofbasevec(fcvec)</code>	From a base vector either: (1,0,0), (0,1,0), (0,0,1), (-1,0,0), (0,-1,0), (0,0,-1) Gets its name: 'x', 'y',....
<code>get_positive_vecname(vecname)</code>	It just get 'x' when vecname is 'x' or '-x', and the same for the others, because some functions receive only positive base vector
<code>get_rot(v1, v2)</code>	Calculate the rotation from v1 to v2 the difference with previous versions, such fc_calc_rot, calc_rot, calc_rot is that it is for any vector direction.
<code>get_tangent_2circles(center1_pt, center2_pt, ...)</code>	Returns a list of lists (matrix) with the 2 tangent points for each of the 2 tangent lines .
<code>get_tangent_circle_pt(ext_pt, center_pt, ...)</code>	Get the point of the tangent to the circle
<code>get_vecname_perpend1(vecname)</code>	Gets a perpendicular vecname
<code>get_vecname_perpend2(vecname)</code>	Gets the other perpendicular vecname (see get_vecname_perpend)
<code>get_fcvec_ofname(axis)</code>	Returns the FreeCAD.Vector of the vector name given
<code>get_vec_ofname(axis)</code>	Get axis name returns the vector
<code>regpolygon_dir_vec1(n_sides, radius, ...)</code>	Similar to regpolygon_vec1 but in any place and direction of the space calculates the vertexes of a regular polygon.
<code>regpolygon_vec1(n_sides, radius[, x_angle])</code>	Calculates the vertexes of a regular polygon.
<code>rotateview([axisX, axisY, axisZ, angle])</code>	Rotate the camera
<code>shpRndRectWire([x, y, r, zpos])</code>	Creates a wire (shape), that is a rectangle with rounded edges.
<code>shp_2stadium_dir(length, r_s, r_l, h_tot, h_rl)</code>	Makes to concentric stadiums, useful for making rails for bolts the length is the same for both.
<code>shp_aluwire_dir(width, thick, slot, insquare)</code>	Creates a wire (shape), that is an approximation of a generic alum profile extrusion.
<code>shp_belt_dir(center_sep, rad1, rad2, height)</code>	Makes a shape of 2 tangent circles (like a belt joining 2 circles).
<code>shp_belt_wire_dir(center_sep, rad1, rad2[, ...])</code>	Makes a shape of a wire with 2 circles and exterior tangent lines check <a href="#">here</a> It is not easy to draw it well rad1 and rad2 can be exchanged, rad1 doesn't have to be larger.
<code>shp_bolt(r_shank, l_bolt, r_head, l_head[, ...])</code>	Similar to addBolt, but creates a shape instead of a FreeCAD Object Creates a shape of the bolt shank and head or the nut Tolerances have to be included if you want it for making a hole
<code>shp_bolt_dir(r_shank, l_bolt, r_head, l_head)</code>	Similar to shp_bolt, but it can be done in any direction Creates a shape, not a FreeCAD Object Creates a shape of the bolt shank and head or the nut Tolerances have to be included if you want it for making a hole
<code>shp_boltnut_dir_hole(r_shank, l_bolt, ...[, ...])</code>	Similar to addBoltNut_hole, but in any direction and creates shapes, not FreeCAD Objects Creates the hole for the bolt shank, the head and the nut.

continues on next page

Table 22 – continued from previous page

<i>shp_box_dir</i> (box_w, box_d, box_h[, ...])	Makes a shape of a box given its 3 dimensions: width, depth and height and the direction of the height and depth dimensions.
<i>shp_box_dir_xtr</i> (box_w, box_d, box_h[, ...])	Makes a shape of a box given its 3 dimensions: width, depth and height and the direction of the height and depth dimensions.
<i>shp_box_rot</i> (box_w, box_d, box_h[, axis_w, ...])	Makes a box with width, depth, height and then rotation will be referred to <i>axis_w</i> = (1,0,0) and <i>axis_nh</i> = (0,0,-1).
<i>shp_boxcen</i> (x, y, z[, cx, cy, cz, pos])	Adds a shape of box, referenced on the specified axis, with its Placement and Rotation at zero.
<i>shp_boxcenchmf</i> (x, y, z, chmfrad[, fx, fy, ...])	Same as <i>shp_boxcen</i> but with a chamfered dimension
<i>shp_boxcenfill</i> (x, y, z, fillrad[, fx, fy, ...])	Same as <i>shp_boxcen</i> but with a filleted dimension
<i>shp_boxcenxtr</i> (x, y, z[, cx, cy, cz, xtr_nx, ...])	The same as <i>shp_boxcen</i> , but when it is used to cut.
<i>shp_boxdir_fillchmfplane</i> (box_w, box_d, box_h)	Creates a box shape (cuboid) along 3 axis.
<i>shp_cableturn</i> (d, w, thick_d, corner_r, ...)	Creates a shape of an electrical cable turn, in any direction But it is a shape in FreeCAD See function <i>wire_cableturn</i> .
<i>shp_cir_fillchmf</i> (shp[, circen_pos, fillet, ...])	Fillet or chamfer edges that is a circle, the shape has to be a cylinder
<i>shp_cyl</i> (r, h[, normal, pos])	Same as <i>addCylPos</i> , but just creates the shape
<i>shp_cyl_gen</i> (r, h[, axis_h, axis_ra, ...])	This is a generalization of <i>shp_cylcenxtr</i> .
<i>shp_cylcenxtr</i> (r, h[, normal, ch, xtr_top, ...])	Add cylinder, can be centered on the position, and also can have an extra mm on top and bottom to make cuts
<i>shp_cylfilletchamfer</i> (shp[, fillet, radius])	Fillet or chamfer all edges of a cylinder
<i>shp_cylhole</i> (r_ext, r_int, h[, axis, h_disp])	Same as <i>addCylHole</i> , but just a shape
<i>shp_cylhole_arc</i> (r_out, r_in, h[, axis_h, ...])	This is similar to make <i>shp_cylhole_gen</i> but not for a whole, just an arc.
<i>shp_cylhole_bolthole</i> (r_out, r_in, h[, ...])	This is a generalization of <i>shp_cylholedir</i> and <i>shp_cylhole</i> Makes a hollow cylinder in any position and direction, with optional extra heights, and inner and outer radius, and various locations in the cylinder
<i>shp_cylhole_gen</i> (r_out, r_in, h[, axis_h, ...])	This is a generalization of <i>shp_cylholedir</i> .
<i>shp_cylholedir</i> (r_out, r_in, h[, normal, pos])	Same as <i>addCylHolePos</i> , but just a shape Same as <i>shp_cylhole</i> , but this one accepts any normal
<i>shp_extrud_face</i> (face, length, vec_extr_axis)	Extrudes a face on any plane
<i>shp_extrud_face_rot</i> (face, vec_facenormal, ...)	Extrudes a face that is on plane XY, includes a rotation
<i>shp_face_lgrail</i> (rail_w, rail_h[, axis_l, axis_b])	Adds a shape of the profile (face) of a linear guide rail, the dent is just rough, to be able to see that it is a profile .
<i>shp_face_rail</i> (rail_w, rail_ws, rail_h[, ...])	Adds a shape of the profile (face) of a rail
<i>shp_filletchamfer</i> (shp, e_len[, fillet, ...])	Fillet or chamfer edges of a certain length, on a certain axis and a certain coordinate
<i>shp_filletchamfer_dir</i> (shp[, fc_axis, ...])	Fillet or chamfer edges on a certain axis
<i>shp_filletchamfer_dirpt</i> (shp[, fc_axis, ...])	Fillet or chamfer edges on a certain axis and a point contained in that axis
<i>shp_filletchamfer_dirpts</i> (shp, fc_axis, fc_pts)	Fillet or chamfer edges on a certain axis and a list of point contained in that axis

continues on next page

Table 22 – continued from previous page

<code>shp_filletchamfer_dirs(shp, fc_axis_l[, ...])</code>	Same as <code>shp_filletchamfer_dir</code> , but with a list of directions
<code>shp_hollowbelt_dir(center_sep, rad1, rad2, ...)</code>	Makes a shape of 2 tangent circles (like a belt joining 2 circles).
<code>shp_nuthole(nut_r, nut_h, hole_h[, xtr_nut, ...])</code>	Similar to <code>NutHole</code> , but creates a shape, in any direction.
<code>shp_regpolygon_dir_face(n_sides, radius[, ...])</code>	Similar to <code>shp_regpolygon_face</code> , but in any direction of the space makes the shape of a face of a regular polygon
<code>shp_regpolygon_face(n_sides, radius[, ...])</code>	Makes the shape of a face of a regular polygon
<code>shp_regprism(n_sides, radius, length[, ...])</code>	Makes a shape of a face of a regular polygon
<code>shp_regprism_dirxtr(n_sides, radius, length)</code>	Similar to <code>shp_regprism_xtr</code> , but in any direction makes a shape of a face of a regular polygon.
<code>shp_regprism_xtr(n_sides, radius, length[, ...])</code>	makes a shape of a face of a regular polygon.
<code>shp_rndrect_face(x, y[, r, pos_z])</code>	Same as <code>shpRndRectWire</code>
<code>shp_stadium_dir(length, radius, height[, ...])</code>	Makes a stadium shape in any direction
<code>shp_stadium_face(l, r[, axis_rect, pos_z])</code>	Same as <code>shp_stadium_wire</code> , but returns a face
<code>shp_stadium_wire(l, r[, axis_rect, pos_z])</code>	Creates a wire (shape), that is a rectangle with semicircles at a pair of opposite sides.
<code>shp_stadium_wire_dir(length, radius[, ...])</code>	Same as <code>shp_stadium_wire</code> but in any direction Also called <code>discorectangle</code> .
<code>vecname_paral(vec1, vec2)</code>	Given to vectors by name 'x', '-x', ...
<code>wire_beltclamp(d, w, corner_r, conn_d, conn_sep)</code>	Creates a wire following 2 pulleys and ending in a belt clamp But it is a wire in FreeCAD, has no volumen .
<code>wire_cableturn(d, w, corner_r, conn_d, conn_sep)</code>	Creates a electrical wire turn, in any direction But it is a wire in FreeCAD, has no volumen .
<code>wire_lgrail(rail_w, rail_h[, axis_w, ...])</code>	Creates a wire of a linear guide rail, the dent is just rough, to be able to see that it is a profile
<code>wire_sim_xy(vecList)</code>	Creates a wire (shape), from a list of points on the positive quadrant of XY the wire is simmetrical to both X and Y .

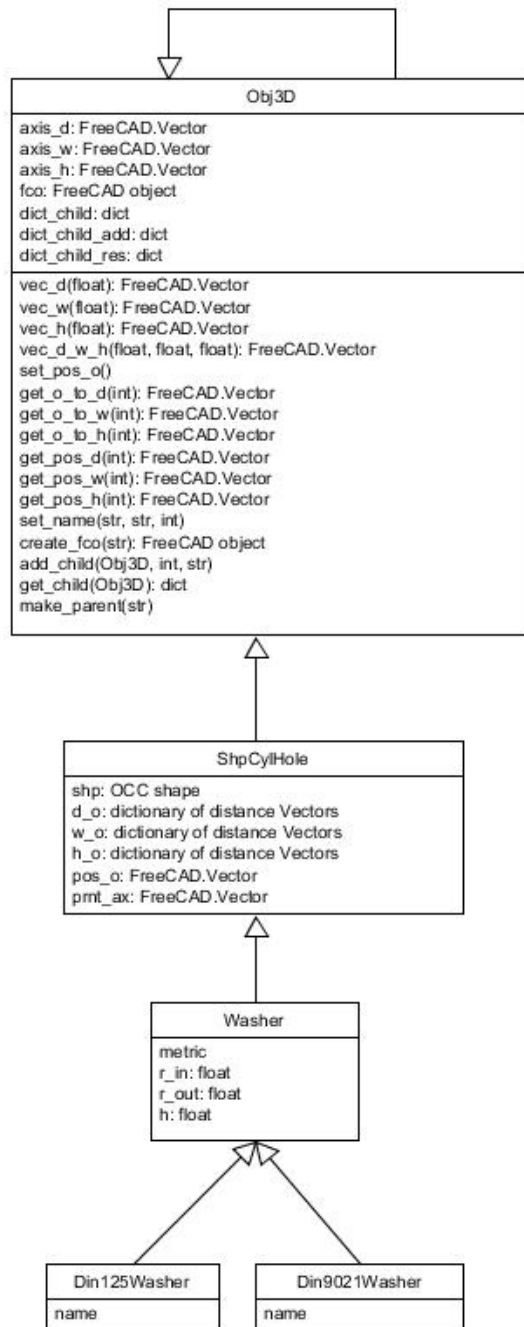
## 1.4.4 UML

The UML (Unified Modeling Language) is the base diagram for software development. It is a visual description of the relationships between class objects.

The main class will be “*Obj3D*” which will contain the basic information of the model:

- Internal axis:
  - `axis_d`
  - `axis_w`
  - `axis_h`
- Children’s dictionary:
  - `dict_child`
  - `dict_child_sum`
  - `dict_child_res`

The rest of the classes that generate the different 3D models will be part of the *Obj3D* class



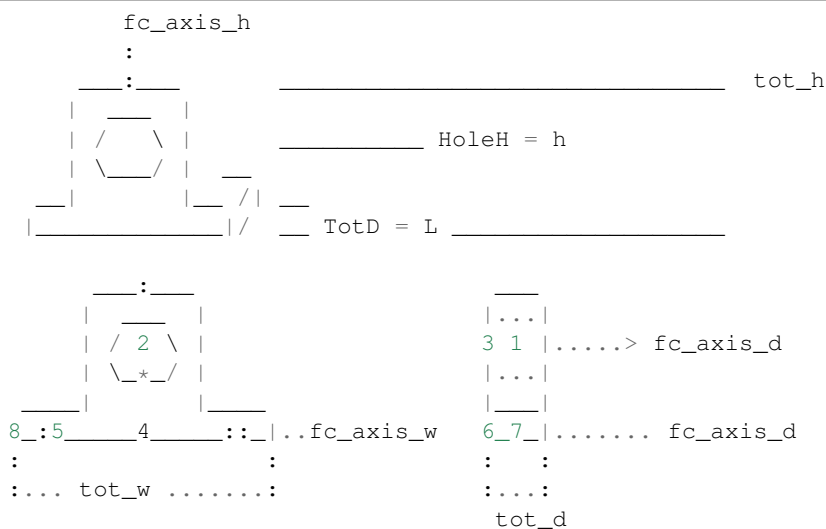
## 1.4.5 3D model details

### Mechanical

```
class comps.Sk_dir(size,          fc_axis_h=FreeCAD.Vector,          fc_axis_d=FreeCAD.Vector,
                  fc_axis_w=FreeCAD.Vector, ref_hr=1, ref_wc=1, ref_dc=1, pillow=0,
                  pos=FreeCAD.Vector, wfco=1, tol=0.3, name='shaft_holder')
```

SK dimensions: dictionary for the dimensions

```
mbolt: is mounting bolt. it corresponds to its metric
tbolt: is the tightening bolt.
SK12 = { 'd':12.0, 'H':37.5, 'W':42.0, 'L':14.0, 'B':32.0, 'S':5.5,
        'h':23.0, 'A':21.0, 'b': 5.0, 'g':6.0, 'I':20.0,
        'mbolt': 5, 'tbolt': 4}
```



#### Parameters

- **fc\_axis\_h** (*FreeCAD.Vector*) – Axis on the height direction
- **fc\_axis\_d** (*FreeCAD.Vector*) – Axis on the depth (rod) direction
- **fc\_axis\_w** (*FreeCAD.Vector*) – Width (perpendicular) dimension, only useful if I finally include the tightening bolt, or if ref\_wc != 1
- **ref\_hr** (*int*) –
  - 1: reference at the Rod Height dimension (rod center): points 1, 2, 3
  - 0: reference at the base: points 4, 5
- **ref\_wc** (*int*) –
  - 1: reference at the center on the width dimension (fc\_axis\_w) points: 2, 4,
  - 0: reference at one of the bolt holes, point 5
  - -1: reference at one end. point 8
- **ref\_dc** (*int*) –

- 1: reference at the center of the depth dimension (`fc_axis_d`) points: 1,7
- 0: reference at one of the ends on the depth dimension points 3, 6
- **pillow** (*int*) –
  - 1 to make it the same height of a pillow block
- **pos** (*FreeCAD.Vector*) – Placement
- **wfco** (*int*) –
  - 1 to create a FreeCAD Object
- **tol** (*float*) – Tolerance of the axis
- **name** (*str*) – FreeCAD Object name

**Returns** FreeCAD Object of a shaft holder

**Return type** FreeCAD Object

```
class comps.PartAluProf(depth, aluprof_dict, xtr_d=0, xtr_nd=0, axis_d=FreeCAD.Vector,
                        axis_w=FreeCAD.Vector, axis_h=FreeCAD.Vector, pos_d=0, pos_w=0,
                        pos_h=0, pos=FreeCAD.Vector, model_type=1, name="")
```

Integration of a ShpAluProf object into a PartAluProf object, so it is a FreeCAD object that can be visualized in FreeCAD Instead of using all the arguments of ShpAluProf, it will use a dictionary

#### Parameters

- **depth** (*float*) – (depth) length of the bar, the extrusion
- **aluprof\_dict** (*dictionary*) – Dictionary with all the information about the profile in kcomp.py there are some dictionaries that can be used, they are not exact – same as ShpAluProf
- **xtr\_d** (*float*) – If >0 it will be that extra depth (length) on the direction of `axis_d`
- **xtr\_nd** (*float*) – If >0 it will be that extra depth (length) on the oposite direction of `axis_d` can be V0 if `pos_h` = 0
- **axis\_d** (*FreeCAD.Vector*) – Axis along the length (depth) direction
- **axis\_w** (*FreeCAD.Vector*) – Axis along the width direction
- **axis\_h** (*FreeCAD.Vector*) – Axis along the width direction
- **pos\_d** (*int*) – Location of pos along `axis_d` (see drawing)
  - 0: start point, counting `xtr_nd`, if `xtr_nd` == 0 -> `pos_d` 0 and 1 will be the same
  - 1: start point, not counting `xtr_nd`
  - 2: middle point not counting `xtr_nd` and `xtr_d`
  - 3: middle point counting `xtr_nd` and `xtr_d`
  - 4: end point, not counting `xtr_d`
  - 5: end point considering `xtr_d`
- **pos\_w** (*int*) – Location of pos along `axis_w` (see drawing). Symmetric, negative indexes means the other side
  - 0: at the center of symmetry
  - 1: at the inner square
  - 2: at the interior side of the outer part of the rail (thickness of the 4 side)

- 3: at the end of the profile along axis\_w
- **pos\_h** (*int*) – Same as pos\_w
- **pos** (*FreeCAD.Vector*) – Position of point defined by pos\_d, pos\_w, pos\_h
- **model\_type** (*int*) – Kind of model
  - 1: dimensional model: it can be printed to assemble a model, but the part will not work as defined. For example, if printed this aluminum profile will not work as defined, and also, it is not exact
- **name** (*str*) – Name of the object

```
class comps.PartLinGuideBlock (block_dict, rail_dict, axis_d=FreeCAD.Vector,
                                axis_w=FreeCAD.Vector, axis_h=FreeCAD.Vector, pos_d=0,
                                pos_w=0, pos_h=0, pos=FreeCAD.Vector, model_type=1,
                                name="")
```

Integration of a ShpLinGuideBlock object into a PartLinGuideBlock object, so it is a FreeCAD object that can be visualized in FreeCAD. Instead of using all the arguments of ShpLinGuideBlock, it will use a dictionary

#### Parameters

- **block\_dict** (*dictionary*) – Dictionary with the information about the block
- **rail\_dict** (*dictionary*) – Dictionary with the information about the rail, it is not necessary, but if not provided, the block will not have the rail hole
- **axis\_d** (*FreeCAD.Vector*) – The axis along the depth (length) of the block (and rail)
- **axis\_w** (*FreeCAD.Vector*) – The axis along the width of the block
- **axis\_h** (*FreeCAD.Vector*) – The axis along the height of the block, pointing up
- **pos\_d** (*int*) – Location of pos along axis\_d (see drawing). Symmetric, negative indexes means the other side
  - 0: at the center (symmetric)
  - 1: at the bolt hole
  - 2: at the end of the smaller part of the block
  - 3: at the end of the end of the block
- **pos\_w** (*int*) – Location of pos along axis\_w (see drawing). Symmetric, negative indexes means the other side
  - 0: at the center of symmetry
  - 1: at the inner hole of the rail
  - 2: at the bolt holes (it can be after the smaller part of the block)
  - 3: at the end of the smaller part of the block
  - 4: at the end of the end of the block
- **pos\_h** (*int*) – Location of pos along axis\_h (see drawing)
  - 0: at the bottom (could make more sense to have 0 at the top instead)
  - 1: at the top of the rail hole
  - 2: at the bottom of the bolt holes, if thruholes, same as 0
  - 3: at the top end
  - 4: at the bottom of the rail (not the block), if the rail has been defined



- ```
class parts.IdlePulleyHolder(profile_size, pulleybolt_d, holdbolt_d, above_h, rail=0, min-
depth=0, attach_dir='-y', endstop_side=0, endstop_posh=0,
pos=FreeCAD.Vector, name='idlepulleyhold')
```

```

hole for endstop
/   []: hole for the nut

|_| + above_h
|_| : Z=0
|_| : aluminum profile
| O O | :
|_| + profile_size
|_| :

O: holes for bolts to attach to the profile

      Z
      :
      :
      :
    /  \
   /____\ /| :
  /_____ /| :
 | |_| | | | + height
 |   [] | | | :
 |   | | | | ..:
 | O  O | / 
 |_____| /.. + depth
 :       :
 :.....:
      + width

attach_dir = '-y' enstop_side= 1 TOP VIEW

      Y
      :
      :
      :
     ____
    | : | |_| |
    | (: ) |
... |_| : _____ X
```

- **profile\_size** (*float*) – Size of the aluminum profile. 20mm, 30mm
- **pulleybolt\_d** (*float*) – Diameter of the bolt used to hold the pulley
- **holdbolt\_d** (*float*) – Diameter of the bolts used to attach this part to the aluminum profile
- **above\_h** (*float*) – Height of this piece above the aluminum profile
- **rail** (*float*) – Possibility of having a rail instead of holes for mounting the holder. It has been made fast, so there may be bugs
- **mindepth** (*float*) – If there is a minimum depth. Sometimes needed for the endstop to

reach its target

- **attach\_dir** (*str*) – Normal vector to where the holder is attached: 'x', '-x', 'y', '-y' NOW ONLY -y IS SUPPORTED. YOU CAN ROTATE IT
- **endstop\_side** (*int*) – -1, 0, 1. Side where the endstop will be if attach\_dir= 'x', this will be referred to the y axis if 0, there will be no endstop
- **endstop\_h** (*float*) – Height of the endstop. If 0 it will be just on top of the profile
- **pos** (*FreeCAD.Vector*) – Object Placement

#### depth

Depth of the holder

**Type** float

#### width

Width of the holder

**Type** float

#### height

Height of the holder

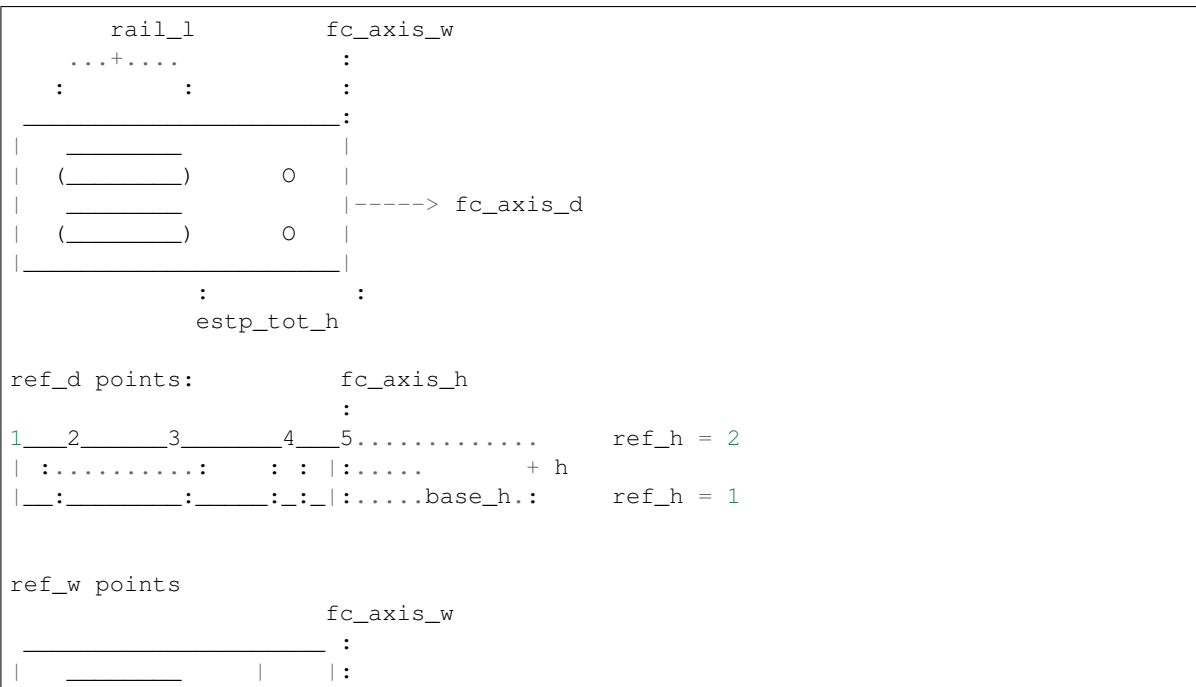
**Type** float

#### fcoFat

Cad object of the compound

```
class parts.SimpleEndstopHolder (d_endstop, rail_l=15, base_h=5.0, h=0, holder_out=2.0,
                                mbolt_d=3.0, endstop_nut_dist=0, min_d=0,
                                fc_axis_d=FreeCAD.Vector, fc_axis_w=FreeCAD.Vector,
                                fc_axis_h=FreeCAD.Vector, ref_d=1, ref_w=1,
                                ref_h=1, pos=FreeCAD.Vector, wfco=1,
                                name='simple_endstop_holder')
```

Very simple endstop holder to be attached to a alu profile and that can be adjusted



(continues on next page)

(continued from previous page)

```

| (_____) ---| 0 | :
1  _____ ---| | :-----> fc_axis_d.
3  (_____) ---| 2 | :
4  _____|_____| :

_____ .....
| :          : : : | :.....: endstop_nut_dist
| :.....: : : : | :
|__ :_____:____:____: | :.....

    if endstop_nut_dist == 0
        just take the length+TOL of the nut

_____
| :          : : : | :
| :.....: : : : | :.....
|__ :_____:____:____: | :.....kcomp.NUT_D934_L[estp_bolt_d]+TOL

```

### Parameters

- **d\_endstop** – Dictionary of the endstop
- **rail\_1** (*float*) – Length of the rail, but only the internal length, not counting the arches to make the semicircles for the bolts just from semicircle center to the other semicircle center
- **h** (*float*) – Total height, if 0 it will be the minimum height
- **base\_h** (*float*) – Height for the base (for the mounting bolts)
- **holder\_out** (*float*) – The endstop holder can end a little bit before to avoid it to be the endstop
- **mbolt\_d** (*float*) – Diameter (metric) of the mounting bolts (for the holder not for the endstop)
- **endstop\_nut\_dist** – Distance from the top to the endstop nut. if zero
- **min\_d** (*int*) – 1: make the endstop axis\_d dimension the minimum
- **fc\_axis\_d** (*FreeCAD Vector*) – Axis along the depth
- **fc\_axis\_w** (*FreeCAD Vector*) – Axis along the width
- **fc\_axis\_h** (*FreeCAD Vector*) – Axis along the height
- **ref\_d** (*int*) – Reference (zero) of fc\_axis\_d
  - 1 = at the end on the side of the rails
  - 2 = at the circle center of one rail (closer to 1)
  - 3 = at the circle center of the other rail, closer to endstop
  - 4 = at the bolt of the endstop
  - 5 = at the end of the endstop (the holder ends before that)
- **ref\_w** (*int*) – Reference on fc\_axis\_w. it is symmetrical, only the negative side
  - 1 = centered
  - 2 = at one endstop bolt the other endstop bolt will be on the direction of fc\_axis\_w
  - 3 = at one rail center the rail center will be on the direction of fc\_axis\_w

- 4 = at the end the end will be on the direction of `fc_axis_w`
- **ref\_h** (*int*) – Reference (zero) of `fc_axis_h`
  - 1: at the bottom
  - 2: on top
- **pos** (*FreeCAD.Vector*) – Object placement
- **wfco** (*int*) – 1 a freecad object will be created
- **name** (*str*) – Name of the freecad object, if created
- **rails can be countersunk to make space for the bolts** (*the*) –

```
class parts.AluProfBracketPerp (alusize_lin,      alusize_perp,      br_perp_thick=3.0,
                                br_lin_thick=3.0, bolt_lin_d=3, bolt_perp_d=0, nbolts_lin=1,
                                bolts_lin_dist=0,   bolts_lin_rail=0,   xtr_bolt_head=3,
                                xtr_bolt_head_d=0, reinforce=1, fc_perp_ax=FreeCAD.Vector,
                                fc_lin_ax=FreeCAD.Vector, pos=FreeCAD.Vector, wfco=1,
                                name='bracket')
```

Bracket to join 2 aluminum profiles that are perpendicular, that is, they are not on the same plane

```

aluprof_perp (perpendicular to the bracket)
  / / / bracket (not drawn)
  / / / _____
  / / / _____|
/_/ / _____|/ aluprof_lin (it is in line with the bracket)
|_|/

          fc_perp_ax (is not the axis of the perpendicular
          :           profile, but the axis of the bracket
aluprof_perp :         attached to the perpendicular profile
          _____:
          | | | \ bracket
          | | | _____ \ .....> fc_line_ax
alusize_lin +          aluprof_lin
          : _____

          fc_perp_ax
          :
          :br_perp_thick
          .+.
          ....:
          : | | \
alusize_perp + | | \
          : | | _____ \..
          :...| _____|...: br_lin_thick .....> fc_lin_ax
          :.....:

```

## Parameters

- **alusize\_lin** (*float*) – Width of the aluminum profile on the line
- **alusize\_perp** (*float*) – Width of the perpendicular aluminum profile
- **br\_lin\_thick** (*float*) – Thickness of the line bracket
- **br\_perp\_thick** (*float*) – Thickness of the perpendicular bracket
- **bolt\_lin\_d** (*int*) – Metric of the bolt 3, 4, ... (integer)

- **bolt\_perp\_d** (*int*) – Metric of the bolt 3, 4, ... (integer) on the profile line if 0, the same as bolt\_lin\_d
- **nbolts\_lin** (*int*) – Number of bolts one bolt on the fc\_lin\_ax, number of bolts: two bolts on the fc\_lin\_ax
- **bolts\_lin\_dist** (*float*) – If more than one bolt on fc\_lin\_ax, defines the distance between them. if zero, takes min distance
- **bolts\_lin\_rail** (*int*) – Instead of bolt holes, it will be a rail it doesnt make sense to have number of bolts with this option it will work on 2 bolts or more. If nbolts\_lin == 3, it will make a rail between them. so it will be the same to have nbolts\_lin = 2 and bolts\_lin\_dist = 20 nbolts\_lin = 3 and bolts\_lin\_dist = 10 The rail will be 20, and it will look the same, it will be more clear to have the first option: 2 bolts
- **xtr\_bolt\_head** (*float*) – Extra space for the bolt head length, and making a space for it
- **xtr\_bolt\_head\_d** (*float*) – Extra space for the bolt head diameter, and making a space for it. For the wall bolt
- **reinforce** (*int*) – 1, if it is reinforced on the sides of lin profile
- **fc\_perp\_ax** (*FreeCAD.Vector*) – Axis of the bracket on the perpendicular prof, see picture
- **fc\_lin\_ax** (*FreeCAD.Vector*) – Axis of the bracket on the aligned profile, see picture
- **pos** (*FreeCAD.Vector*) – Position of the center of the bracket on the intersection
- **wfco** (*int*) –
  - if 1: With FreeCad Object: a freecad object is created
  - if 0: only the shape
- **name** (*str*) – Name of the freecad object, if created

```
class parts.AluProfBracketPerpFlap (alusize_lin,      alusize_perp,      br_perp_thick=3.0,
                                   br_lin_thick=3.0,      bolt_lin_d=3,      bolt_perp_d=0,
                                   nbolts_lin=1,      bolts_lin_dist=0,      bolts_lin_rail=0,
                                   xtr_bolt_head=1,      sunk=1,      flap=1,
                                   fc_perp_ax=FreeCAD.Vector, fc_lin_ax=FreeCAD.Vector,
                                   pos=FreeCAD.Vector, wfco=1, name='bracket_flap')
```

Bracket to join 2 aluminum profiles that are perpendicular, that is, they are not on the same plane It is wide because it has 2 ears/flaps? on the sides, to attach to the perpendicular profile

```
aluprof_perp (perpendicular to the bracket)
  /  /  /  bracket (not drawn)
  /  /  /  _____
  /  /  /  _____|
/___/  /_____|| aluprof_lin (it is in line with the bracket)
|___||

          fc_perp_ax (is not the axis of the perpendicular
          :          profile, but the axis of the bracket
aluprof_perp :          attached to the perpendicular profile
          :
          :___:
          |___| \ bracket
          |___|_____ \_____ .....> fc_line_ax
```

(continues on next page)

(continued from previous page)

```

alusize_lin +          aluprof_lin
      :_____

      fc_perp_ax
      :
      :br_perp_thick
      .+.
      ....:___:
      : | | \
alusize_perp + | | \
      : | | _____\..
      :...|_|_____|...: br_lin_thick .....> fc_lin_ax
      :.....:

```

### Parameters

- **alusize\_lin** (*float*) – Width of the aluminum profile on the line
- **alusize\_perp** (*float*) – Width of the perpendicular aluminum profile
- **br\_lin\_thick** (*float*) – Thickness of the line bracket
- **br\_perp\_thick** (*float*) – Thickness of the perpendicular bracket
- **bolt\_lin\_d** (*int*) – Metric of the bolt 3, 4, ... (integer)
- **bolt\_perp\_d** (*int*) – Metric of the bolt 3, 4, ... (integer) on the profile line if 0, the same as bolt\_lin\_d
- **nbolts\_lin** (*int*) –
  - 1: just one bolt on the fc\_lin\_ax, or two bolts
  - 2: two bolts on the fc\_lin\_ax, or two bolts
- **bolts\_lin\_dist** (*float*) – If more than one bolt on fc\_lin\_ax, defines the distance between them. if zero, takes min distance
- **bolts\_lin\_rail** (*int*) – Instead of bolt holes, it will be a rail it doesnt make sense to have number of bolts with this option it will work on 2 bolts or more. If nbolts\_lin == 3, it will make a rail between them. so it will be the same to have nbolts\_lin = 2 and bolts\_lin\_dist = 20 nbolts\_lin = 3 and bolts\_lin\_dist = 10 The rail will be 20, and it will look the same, it will be more clear to have the first option: 2 bolts
- **xtr\_bolt\_head** (*float*) – Extra space for the bolt head on the line to the wall (perpendicular)
- **sunk** (*int*) –
  - 0: just drilled
  - 1: if the top part is removed,
  - 2: No reinforcement at all
- **flap** (*int*) – If it has flaps, if it hasnt flaps, it is kind of useless because it is just the middle part without bolts on the wall, but it can be used to make an union with other parts
- **fc\_perp\_ax** (*FreeCAD.Vector*) – Axis of the bracket on the perpendicular prof, see picture
- **fc\_lin\_ax** (*FreeCAD.Vector*) – Axis of the bracket on the aligned profile, see picture

- ```
class parts.AluProfBracketPerpTwin (alusize_lin, alusize_perp, alu_sep, br_perp_thick=3.0,
br_lin_thick=3.0, bolt_lin_d=3, bolt_perp_d=0,
nbolts_lin=1, bolts_lin_dist=1, bolts_lin_rail=0,
bolt_perp_line=0, xtr_bolt_head=3, sunk=0,
fc_perp_ax=FreeCAD.Vector,fc_lin_ax=FreeCAD.Vector,
fc_wide_ax=FreeCAD.Vector, pos=FreeCAD.Vector,
wfco=1, name='bracket_twin')
```

```

aluprof_perp (perpendicular to the bracket)
      . fc_wide_ax

      .
      /  /  /|.
      /  /  /_____
      /  /  /_____ /| aluprof_lin (it is in line with the bracket)
      /  /  /_____ | /-----
      /  /  /_____ . alu_sep
      /  /  *_____ /|
      /___ /_____ | /-----
      |___ /|
      aluprof_lin (it is in line with the bracket)
      * shows the reference for the position (argument pos)
      the direction of fc_wide_ax indicates where the other
      line of the bracket will be

      fc_perp_ax (is not the axis of the perpendicular
      :          profile, but the axis of the bracket
aluprof_perp  :          attached to the perpendicular profile
      :
      :_____
      | | | \ bracket
      | | | \_____ .....> fc_line_ax
alusize_lin + aluprof_lin
      :_____

      fc_perp_ax
      :
      :br_perp_thick
      .+.
      ....:___:
      : | | \
alusize_perp + | | \
      : | | \_____ \..
      :...|_____|...: br_lin_thick .....> fc_lin_ax
      :.....:

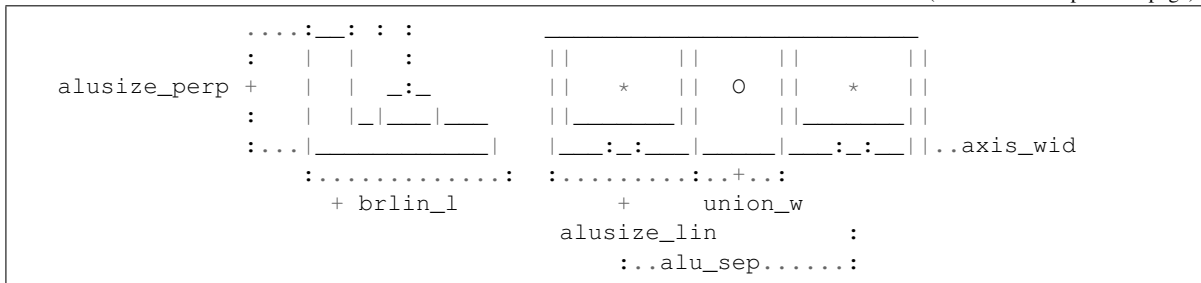
      * bolt_perp_line
      1: * there is a bolt hole
      0: * no bolt hole there

```

(continues on next page)

43

(continued from previous page)



## Parameters

- **alusize\_lin** (*float*) – Width of the aluminum profile on the line
- **alusize\_perp** (*float*) – Width of the perpendicular aluminum profile
- **alu\_sep** (*float*) – Separation of the 2 paralell profiles, from their centers
- **br\_lin\_thick** (*float*) – Thickness of the line bracket
- **br\_perp\_thick** (*float*) – Thickness of the perpendicular bracket
- **bolt\_lin\_d** (*int*) – Metric of the bolt 3, 4, ... (integer)
- **bolt\_perp\_d** (*int*) – Metric of the bolt 3, 4, ... (integer) on the profile line if 0, the same as bolt\_lin\_d
- **nbolts\_lin** (*int*) –
  - 1: just one bolt on the fc\_lin\_ax, or two bolts
  - 2: two bolts on the fc\_lin\_ax, or two bolts
- **bolts\_lin\_dist** (*float*) – If more than one bolt on fc\_lin\_ax, defines the distance between them. if zero, takes min distance
- **bolts\_lin\_rail** (*int*) – Instead of bolt holes, it will be a rail it doesnt make sense to have number of bolts with this option it will work on 2 bolts or more. If nbolts\_lin == 3, it will make a rail between them. so it will be the same to have nbolts\_lin = 2 and bolts\_lin\_dist = 20 nbolts\_lin = 3 and bolts\_lin\_dist = 10 The rail will be 20, and it will look the same, it will be more clear to have the first option: 2 bolts
- **bolt\_perp\_line** (*int*) –
  - 1: if it has a bolt on the wall (perp) but in line with the line aluminum profiles
  - 0: no bolt
- **xtr\_bolt\_head** (*float*) – Extra space for the bolt head, and making a space for it only makes sense if bolt\_perp\_line == 1
- **sunk** (*int*) –
  - 0: No sunk, just drill holes: bolt\_perp\_line should be 0
  - 1: sunk, but with reinforcement if possible
  - 2: no reinforcement
- **fc\_perp\_ax** (*FreeCAD.Vector*) – Axis of the bracket on the perpendicular prof, see picture
- **fc\_lin\_ax** (*FreeCAD.Vector*) – Axis of the bracket on the aligned profile, see picture



- **fc\_wide\_ax** (*FreeCAD.Vector*) – Axis of the bracket on wide direction, see picture its direction shows where the other aligned profile is
- **pos** (*FreeCAD.Vector*) – Position of the center of the bracket on the intersection
- **wfco** (*int*) –
  - 1: With FreeCad Object: a freecad object is created
  - 0: only the shape
- **name** (*str*) – Name of the freecad object, if created

```
class parts.NemaMotorHolder (nema_size=17, wall_thick=4.0, motor_thick=4.0, reinf_thick=4.0,
                             motor_min_h=10.0, motor_max_h=20.0, rail=1, mo-
                             tor_xtr_space=2.0, motor_xtr_space_d=- 1, bolt_wall_d=4.0,
                             bolt_wall_sep=30.0, chmf_r=1.0, fc_axis_h=FreeCAD.Vector,
                             fc_axis_n=FreeCAD.Vector, ref_axis=1, pos=FreeCAD.Vector,
                             wfco=1, name='nema_holder')
```

Creates a holder for a Nema motor

```

|| _____ || | |
||  O      _  O  ||
||   /      \   ||
||  |  1  |   ||
||   \      /   ||
||  O      _  O  ||
|| _____ || .....
|| _____ || ..... wall_thick

                                     motor_xtr_space_d
                                     :  :
                                     :  :
_____3_____ 3_:::_____ .....
|  ::  :  :  :  : | + motor_thick
|_:::_:1_:::_:| 2.....1.....|.....> fc_axis_n
||           || /
||           || /
||           || /
||           || /
|| _____ || |_/
::           :           :
+ reinf_thick :.....tot_d.....:

      fc_axis_h
      :
      :_____ .....
|  ::  :  :  :  : | :
|_:::_:1_:::_:| .....:
||           || .....+ motor_min_h :
||           || .....: +tot_h
||           || .....+ motor_max_h :
||           || .....:
||           || .....:
|| _____ || .....:
:  :           :  :
:  :           :  :
:  :.....:  :
```

(continues on next page)

(continued from previous page)

```

:   bolt_wall_sep   :
:                   :
:                   :
:.....tot_w.....:
:                   ::
:                   motor_xtr_space

1: ref_axis = 1 & ref_bolt = 0
2: ref_axis = 0 & ref_bolt = 0
--3: ref_axis = 0 & ref_bolt = 0

```

### Parameters

- **nema\_size** (*int*) – Size of the motor (NEMA)
- **wall\_thick** (*float*) – Thickness of the side where the holder will be screwed to
- **motor\_thick** (*float*) – Thickness of the top side where the motor will be screwed to
- **reinf\_thick** (*float*) – Thickness of the reinforcement walls
- **motor\_min\_h** (*float*) – Distance of from the inner top side to the top hole of the bolts to attach the holder (see drawing)
- **motor\_max\_h** (*float*) – Distance of from the inner top side to the bottom hole of the bolts to attach the holder
- **rail** (*int*) –
  - 2: the rail goes all the way to the end, not closed
  - 1: the holes for the bolts are not holes, there are 2 rails, from motor\_min\_h to motor\_max\_h
  - 0: just 2 pairs of holes. One pair at defined by motor\_min\_h and the other defined by motor\_max\_h
- **motor\_xtr\_space** (*float*) – Extra separation between the motor and the sides
- **motor\_xtr\_space\_d** (*float*) – Extra separation between the motor and the wall side (where the bolts) it didn't exist before, so for compatibility
  - -1: same has motor\_xtr\_space (compatibility), considering bolt head length
  - 0: no separation
  - >0: exact separation
- **bolt\_wall\_d** (*int/float*) – Metric of the bolts to attach the holder
- **bolt\_wall\_sep** (*float*) – Separation between the 2 bolt holes (or rails). Optional.
- **chmf\_r** (*float*) – Radius of the chamfer, whenever chamfer is done
- **fc\_axis\_h** (*FreeCAD Vector*) – Axis along the axis of the motor
- **fc\_axis\_n** (*FreeCAD Vector*) – Axis normal to surface where the holder will be attached to
- **ref\_axis** (*int*) –
  - 1: the zero of the vertical axis (axis\_h) is on the motor axis
  - 0: the zero of the vertical axis (axis\_h) is at the wall

- **pos** (*FreeCAD.Vector*) – Position of the holder (considering *ref\_axis*)
- **wfco** (*int*) –
  - 1: creates a FreeCAD object
  - 0: only creates a shape
- **name** (*string*) – Name of the FreeCAD object

[illegible]

Makes a housing for a linear bearing, but it is very thin and intended to be attached to one rail, instead of 2 it has to parts, the lower and the upper part

:::::		:::::::::::	
::	Upper part	::	::
-:: ( ) ::	-----	::	::
:::::	Lower part	:::::::::::	
_::__::		::	::
_:: _::		_::: _::: _::: _:::	

--> fc\_slide\_axis

A 6x6 grid with a top and bottom row of zeros and vertical columns of dots.

0	:	:	:	:	0
:	:	:	:	:	:
:	:	:	:	:	:
:	:	:	:	:	:
:	:	:	:	:	:
0	:	:	:	:	0

```

      :
      :
      v
fc_bot_axis

```

```

|  ::...:: |
|  ::      |
| -:: ( ) :: ----- Upper part
|  ::...:: |
| _::_____ Lower part
|  ::      |
|  ::      |

```

```

      | ::::::::::: |
      | ::          :: |
      | ::      *axis_center = 1
      | ::::::::::: |
_____ | ::          :: | _____
| _:::_:::___*_____:::_:::___|
      axis center = 0

```

|  
 |  
 v  
 always centered **in** this axis

```
1: axis_center=1
   mid_center =1
2: axis_center=0
   mid_center =1
3: axis_center=0
   mid_center =0
4: axis_center=1
   mid_center =0
```

```

      | : : : : : : : : |
      | : : : : : : : : |
4      | : :      1 --> fc_slide_axis
      | : : : : : : : : |
      | : : : : : : : : |
      | : : : : : : : : |
      | : 3 : : : : 2 : : : |

```

## Parameters

- **d\_lbear**(*dictionary*) – Dictionary with the dimensions of the linear bearing

- **fc\_slide\_axis** (*FreeCAD.Vector*) – Direction of the slide
- **fc\_bot\_axis** (*FreeCAD.Vector*) – Direction of the bottom
- **axis\_center** (*int*) – See picture, indicates the reference point
- **mid\_center** (*int*) – See picture, indicates the reference point
- **pos** (*FreeCAD.Vector*) – Position of the reference point,

**n1\_slide\_axis**

Type *FreeCAD.Vector*

**n1\_bot\_axis**

Type *FreeCAD.Vector*

**n1\_perp**

Type *FreeCAD.Vector*

**axis\_h**

Type *float*

**boltcen\_axis\_dist**

Type *float*

**boltcen\_perp\_dist**

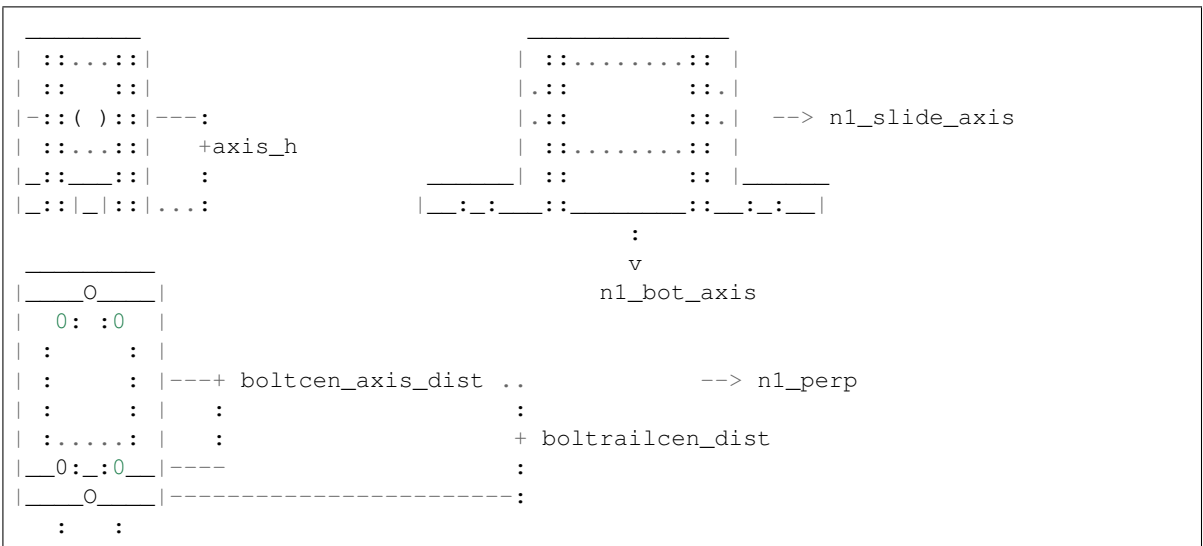
Type *float*

#### Dimensions:

- tot\_h, tot\_w, tot\_l
- housing\_l, base\_h

#### FreeCAD objects:

- fco\_top : Top part of the linear bearing housing
- fco\_bot : Bottom part of the linear bearing housing



(continues on next page)

(continues on next page)

```

| :...: |
| ::  : |      Upper part
|--:( )--:-----
| :...: |      Lower part
|_::__::_|

| :.....: |
| ::      : |
| :...:   : |    --> fc_slide_axis
| :.....: |
|_::_____:|_

          :
          :
          v
        fc_bot_axis

| 0: :0 |
| :      : |
| :      : |
| :      : |-----> fc_perp_axis
| :      : |
| :.....: |
|_0:_:0_|

| :...: |
| ::  : |      Upper part
|--:( )--:-----> fc_perp_axis
| :...: |      Lower part
|_::__::_|

| |
| |
v  v
centered in any of these axes

l: axis_center=1

| :.....: |

```

(continues on next page)

(continued from previous page)

```

mid_center =1          |.:|      :.|
2: axis_center=0        |.:4  1 --> fc_slide_axis
mid_center =1          | :.....: |
3: axis_center=0        | :|      :|
mid_center =0          |_:3__2____:|_
4: axis_center=1

```

And 8 more possibilities:

```

5: bolt_center = 1
6: bolt_center = 0

| 5:6: |
| :   : |
| :   : |
| :   : |-----> fc_perp_axis
| :   : |
| :.....: |
|_0:_:0_|
mid_center =0

```

### Parameters

- **d\_lbear** (*dictionary*) – Dictionary with the dimensions of the linear bearing
- **fc\_slide\_axis** (*FreeCAD.Vector*) – Direction of the slide
- **fc\_bot\_axis** (*FreeCAD.Vector*) – Direction of the bottom
- **fc\_perp\_axis** (*FreeCAD.Vector*) – Direction of the other perpendicular direction.  
Not useful unless bolt\_center == 1 if = V0 it doesn't matter
- **axis\_h** (*int*) – Distance from the bottom to the rod axis
  - 0: take the minimum distance
  - X: (any value) take that value, if it is smaller than the minimum it will raise an error and would not take that value
- **bolts\_side** (*int*) – See picture, indicates the side where is bolt
- **axis\_center** (*int*) – See picture, indicates the reference point
- **mid\_center** (*int*) – See picture, indicates the reference point
- **bolt\_center** (*int*) – See picture, indicates the reference point, if it is on the bolt or on the axis
- **pos** (*FreeCAD.Vector*) – Position of the reference point,

**n1\_slide\_axis**

Type FreeCAD.Vector

**n1\_bot\_axis**

Type FreeCAD.Vector

**n1\_perp**

Type FreeCAD.Vector

**axis\_h**



(continued from previous page)

```
| :.....: |          |_0:_____:0_|
|__0:__:0__|
```

```
class parts.LinBearHouse (d_lbearhousing,          fc_slide_axis=FreeCAD.Vector,
                          fc_bot_axis=FreeCAD.Vector,  axis_center=1,    mid_center=1,
                          pos=FreeCAD.Vector, name='linbearhouse')
```

Makes a housing for a linear bearing takes the dimensions from a dictionary, like the one defined in kcomp.py it has to parts, the lower and the upper part

```
| ::      ____      :: |          | ::.....: :: | | | |
| :: /      \      :: |  Upper part | ::      :: |
| ---|-----|---|-----|         | ::      :: |
| :: \____/      :: |  Lower part  | ::.....: :: |
| ::_____: :: |          |_::_____::_|
```

```
| 0 :      : 0 |
| :      : |
| :      : |
| :      : |
| :      : |
|_0:_____:0_|
```

```
1: axis_center=1
   mid_center =1
2: axis_center=0
   mid_center =1
3: axis_center=0
   mid_center =0
4: axis_center=1
   mid_center =0

| : :.....: : |
|.: :      : :|
|.:4:  1 ----->: fc_slide_axis
| : :.....: : |
| : :      : : |
|_:3:___2_____::_|
```

```
class parts.ThinLinBearHouseAsim (d_lbear,          fc_fro_ax=FreeCAD.Vector,
                                   fc_bot_ax=FreeCAD.Vector,  fc_sid_ax=FreeCAD.Vector,
                                   axis_h=0,  bolts_side=1,  refcen_hei=1,  refcen_dep=1,
                                   refcen_wid=1,  bolt2cen_wid_n=0,  bolt2cen_wid_p=0,
                                   pos=FreeCAD.Vector, name='thinlinbearhouse_asim')
```

There are

3 axis:	3 planes (normal to axis)	3 distances to plane
fc_fro_ax	fro: front	D: dep: depth
fc_bot_ax	hor: horizontal)	H: hei: height
fc_sid_ax	lat: lateral (medial)	W: wid: width

The planes are on the center of the sliding rod (height and width), and on the middle of the piece (width)

The 3 axis are perpendicular, but the cross product of 2 vectors may result on the other vector or its negative.

fc\_fro\_ax points to the front of the figure, but it is symmetrical so it can point to the back fc\_bot\_ax points to the bottom of the figure (not symmetrical) fc\_sid\_ax points to the side of the figure. Not symmetrical if bolt2cen\_wid\_n or bolt2cen\_wid\_p are not zero



Makes a housing for a linear bearing, but it is very thin and intended to be attached to 2 rail it has to parts, the lower and the upper part

```

| : : : : : |
| : : : : | Upper part
| - : ( ) : : | ----- Horizontal plane
| : : : : : | Lower part
| _ : : : : : |

| : : : : : |
| : : : : |
| : : : : |
| : : : : | -----> fc_sid_ax
| : : : : |
| : : : : |
| _ 0 : : 0 _ |

| : : : : : |
| : : : : | Upper part
| - : ( ) : : | -----> fc_sid_ax
| : : : : : | Lower part
| _ : : : : : |
| |
| |
V V
centered in any of these axes
refcen_hei: reference centered on the height
    =1: the horizontal plane (height) is on the axis of the rod
    =0: the horizontal plane is at the bottom
refcen_dep: reference centered on the depth
    =1: the frontal plane (depth) is on the middle of the piece
    =0: the frontal plane is at the bolts
refcen_wid=1: reference centered on the width
    the lateral plane (width) is on the medial axis, dividing
    the piece on the right and left
    =0: the lateral plane is at the bolts

1: refcen_hei=1
   fro_center =1
2: refcen_hei=0
   fro_center =1
3: refcen_hei=0
   fro_center =0
4: refcen_hei=1

| : : : : : |
| : : : : |
| : : : : |
| : : : : | -----> fc_fro_ax
| : : : : : |
| : : : : |
| _ : 3 _ 2 _ : : _ |

And 8 more possibilities:
5: refcen_wid = 0
6: refcen_wid = 1

| : : : : : |
| : : : : |
| : : : : |
| : : : : | -----> fc_sid_ax

```

(continues on next page)

(continued from previous page)

```
| :      : |
| :.....: |
|__0:__:0__|
```

**Parameters**

- **d\_lbear** (*dictionary*) – Dictionary with the dimensions of the linear bearing
- **fc\_fro\_ax** (*FreeCAD.Vector*) – Direction of the slide
- **fc\_bot\_ax** (*FreeCAD.Vector*) – Direction of the bottom
- **fc\_sid\_ax** (*FreeCAD.Vector*) – Direction of the other perpendicular direction. Not useful unless refcen\_wid == 0 if = V0 it doesn't matter
- **axis\_h** (*float*) – Distance from the bottom to the rod axis
  - 0: take the minimum distance
  - X: (any value) take that value, if it is smaller than the minimum it will raise an error and would not take that value
- **refcen\_hei** (*int*) – See picture, indicates the reference point
- **refcen\_dep** (*int*) – See picture, indicates the reference point
- **refcen\_wid** (*int*) – See picture, indicates the reference point, if it is on the bolt or on the axis
- **pos** (*FreeCAD.Vector*) – Position of the reference point,

**nfro\_ax****Type** FreeCAD.Vector normalized fc\_fro\_ax**nbot\_ax****Type** FreeCAD.Vector normalized fc\_bot\_ax**nsid\_ax****Type** FreeCAD.Vector**axis\_h****Type** float**bolt2cen\_dep****Type** float**bolt2cen\_wid\_n****Type** float**bolt2cen\_wid\_p****Type** float**bolt2bolt\_wid****Type** bolt2cen\_wid\_n + bolt2cen\_wid\_p**Dimensions:**

- D : float

- housing\_d
- W : float
- housing\_w
- H : float
- housing\_h

**FreeCAD objects:**

- fco\_top : Top part of the linear bearing housing
- fco\_bot : Bottom part of the linear bearing housing

```

| .....|
| : : : |
| : : : |
| -:: ( ) : : | ---:
| : : : : | +axis_h
| _:: : : : |
| _:: : : : | ...:

| .....|
| : : : : : |
| : : : : : |
| : : : : : | --> nfro_ax
| : : : : : |
| : : : : : |
| : : : : : |
| _:: : : : : |
|
| :
| v
| nbot_ax

| 0: :0 |
| : : : |
| : : : | ..... --> nsid_ax
| : : : | :
| : : : | + boltcen_dep
| _0: :0_ | ---:
| : : :
| : : :
| : : :
| : + bolt2cen_wid_p: distance form the bolt to the center
| : on the width dimension. The bolt on the positive side
| + bolt2cen_wid_n: distance form the bolt to the center
| : on the width dimension. The bolt on the negative side
| :
| + if refcen_wid=0 the reference will be on the bolt2cen_wid_n

| ..... D .....
| : : :
| : : :
| .....|
| : : : : : |
| : : : : : |
| : : : : : | --> nfro_ax
| : : : : : |
| : : : : : |
| _:: : : : : |
|
| :
| :
| : .....:
| +
| W

bolts_side = 0 bolts_side = 1

| 0: :0 |
| : : : |
| : : : |
| 0: :0 |

```

(continues on next page)

(continued from previous page)

$\begin{array}{ccc ccc} & : & & & : & \\ & : & & & : & \\ & : & & & : & \\ & : & & & : & \\ & : & & & : & \\ \hline & 0 & : & 0 & & \end{array}$	$\begin{array}{ccc ccc} & : & & & : & \\ & : & & & : & \\ & : & & & : & \\ & : & & & : & \\ & : & & & : & \\ \hline & 0 & : & & : & 0 \end{array}$
--	--

```
class filter_holder_cls.PartFilterHolder (filter_l=60.0,    filter_w=25.0,    filter_t=2.5,
base_h=6.0,    hold_d=12.0,    filt_supp_in=2.0,
filt_rim=3.0,    filt_cen_d=0,    fillet_r=1.0,
boltcol1_dist=10.0,    boltcol2_dist=12.5,
boltcol3_dist=25,    boltrow1_h=0,
boltrow1_2_dist=12.5,    boltrow1_3_dist=20.0,
boltrow1_4_dist=25.0,    bolt_cen_mtr=4,
bolt_linguide_mtr=3,    beltclamp_t=3.0,
beltclamp_l=12.0,    beltclamp_h=8.0,
clamp_post_dist=4.0,    sm_beltpost_r=1.0,
tol=0.4,    axis_d=FreeCAD.Vector,
axis_w=FreeCAD.Vector,
axis_h=FreeCAD.Vector,    pos_d=0,    pos_w=0,
pos_h=0, pos=FreeCAD.Vector, model_type=0,
name="")
```

Integration of a ShpFilterHolder object into a PartFilterHolder object, so it is a FreeCAD object that can be visualized in FreeCAD

```
class filter_holder_cls.ShpFilterHolder (filter_l=60.0,    filter_w=25.0,    filter_t=2.5,  
base_h=6.0,    hold_d=12.0,    filt_supp_in=2.0,  
filt_rim=3.0,    filt_cen_d=0,    fillet_r=1.0,  
boltcol1_dist=10.0,    boltcol2_dist=12.5,  
boltcol3_dist=25,    boltrow1_h=0,  
boltrow1_2_dist=12.5,    boltrow1_3_dist=20.0,  
boltrow1_4_dist=25.0,    bolt_cen_mtr=4,  
bolt_linguide_mtr=3,    beltclamp_t=3.0,  
beltclamp_l=12.0,    beltclamp_h=8.0,  
clamp_post_dist=4.0,    sm_beltpost_r=1.0,  
tol=0.4,    axis_d=FreeCAD.Vector,  
axis_w=FreeCAD.Vector,  
axis_h=FreeCAD.Vector, pos_d=0, pos_w=0,  
pos_h=0, pos=FreeCAD.Vector)
```

Creates the filter holder shape

```

                                beltpost_l = 3*lr_beltpost_r + sm_beltpost_r
pos_h                axis_h      :      :
|                    :      :   clamp_post_dist
v pos_w              :      :   ....
9 7___6   5___4      :      :___:   ___
8 |   |   |   |   :   |   |   |   |
7 |...|___|___|___:___|___|___|...|...
|               _               _               |   2 * bolt_linguide_head_r_tol
6 |               |o|               |o|               |-----|
5 |               |o|               |o|               |-----| +boltrow1_4_dist
|               |               |               |               :   :
|               |               |               |               +boltrow1_3_dist
4 |               (O)               (O)               |--:   :   :
|               |               |               |               +boltrow1_2_dist   :   :
|               |               |               |               :   :
3 | (O)   (o)   (O)   (o)   (O)   |---:-----:---:

```

(continues on next page)

(continued from previous page)

```

|_____| + boltrow1_h
2 |_____| ..:.....
1 | :.....: |..: filt_hole_h :
| : : | + base_h
0 |___:_____x_____:_|.....axis_w
    : : : :
    :.....: : :
    : + boltcol1_dist
    : : :
    :.....: :
    : + boltcol2_dist
    : :
    :.....:
    boltcol3_dist

    3      21      0      pos_w (position of the columns)
7 6 5 4      pos_w (position of the belt clamps)

    beltclamp_l
    clamp_post ..+...
    V : :
    _____x_____:_|.....> axis_w
|_____| |_____|.. beltclamp_blk_t :
|_____| < ) ( > _____|..: beltclamp_t :+ hold_d
|_____|_____|_____|.....:
|_____|
|_____|.....
| | :.....: | | : :
| | : : | | : :+filt_hole_d
| | : : | | + filt_supp_d :
| | :.....: | | ..: :
| |_____| | |.....:
\_____/_|.....filt_rim
: : : : :
: : : : :
: : : :+: :
: : : filt_supp_in : :
: : : : :
: : :.... filt_supp_w .....: :
: : : : :
: : : : :
: : :.... filt_hole_w .....: :
: : :+: :
: : : filt_rim :
: : :
: : : tot_w .....:
: : :

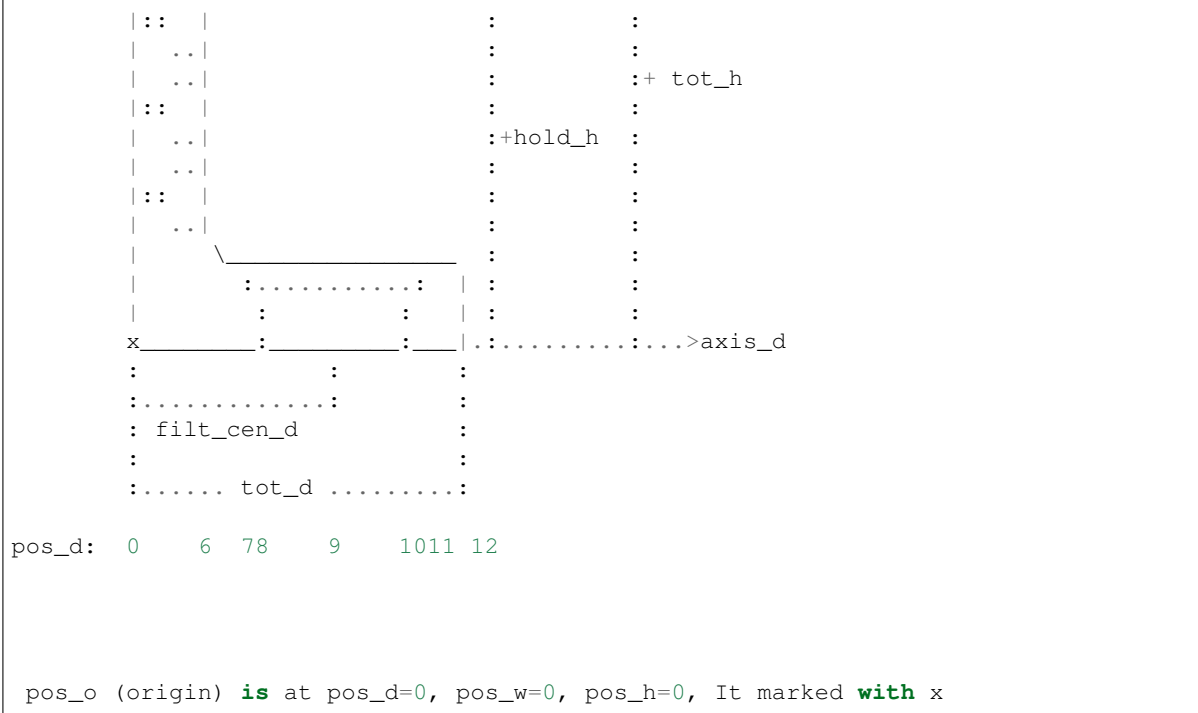
0123 pos_d
0 45 pos_d

|_|_|_| + beltclamp_h :
|_|_|_|.....:
|_|_|_| :
|:|_| :

```

(continues on next page)

(continued from previous page)



### Parameters

- **filter\_l** (*float*) – Length of the filter (it will be along axis\_w). Larger dimension
- **filter\_w** (*float*) – Width of the filter (it will be along axis\_d). Shorter dimension
- **filter\_t** (*float*) – Thickness/height of the filter (it will be along axis\_h). Very short
- **base\_h** (*float*) – Height of the base
- **hold\_d** (*float*) – Depth of the holder (just the part that holds)
- **filt\_supp\_in** (*float*) – How much the filter support goes inside from the filter hole
- **filt\_cen\_d** (*float*) – Distance from the filter center to the beginning of the filter holder along axis\_d
  - 0: it will take the minimum distance or if it is smaller than the minimum distance
- **filt\_rim** (*float*) – Distance from the filter to the edge of the base
- **fillet\_r** (*float*) – Radius of the fillets
- **boltcol1\_dist** (*float*) – Distance to the center along axis\_w of the first column of bolts
- **boltcol2\_dist** (*float*) – Distance to the center along axis\_w of the 2nd column of bolts
- **boltcol3\_dist** (*float*) – Distance to the center along axis\_w of the 3rd column of bolts This column could be closer to the center than the 2nd, if distance is smaller
- **boltrow1\_h** (*float*) – Distance from the top of the filter base to the first row of bolts
  - 0: the distance will be the largest head diameter in the first row in any case, it has to be larger than this

- **boltrow1\_2\_dist** (*float*) – Distance from the first row of bolts to the second
- **boltrow1\_3\_dist** (*float*) – Distance from the first row of bolts to the third
- **boltrow1\_4\_dist** (*float*) – Distance from the first row of bolts to the 4th
- **bolt\_cen\_mtr** (*integer (could be float: 2.5)*) – Diameter (metric) of the bolts at the center or at columns other than 2nd column
- **bolt\_linguide\_mtr** (*integer (could be float: 2.5)*) – Diameter (metric) of the bolts at the 2nd column, to attach to a linear guide
- **beltclamp\_t** (*float*) – Thickness of the hole for the belt. Inside de belt clamp blocks (along axis\_d)
- **beltclamp\_l** (*float*) – Length of the belt clamp (along axis\_w)
- **beltclamp\_h** (*float*) – Height of the belt clamp: belt width + 2 (along axis\_h)
- **clamp\_post\_dist** (*float*) – Distance from the belt clamp to the belt clamp post
- **sm\_beltpost\_r** (*float*) – Small radius of the belt post
- **tol** (*float*) – Tolerances to print
- **axis\_d** (*FreeCAD.Vector*) – Length/depth vector of coordinate system
- **axis\_w** (*FreeCAD.Vector*) – Width vector of coordinate system if V0: it will be calculated using the cross product: axis\_d x axis\_h
- **axis\_h** (*FreeCAD.Vector*) – Height vector of coordinate system
- **pos\_d** (*int*) – Location of pos along the axis\_d (0,1,2,3,4,5), see drawing
  - 0: at the back of the holder
  - 1: at the end of the first clamp block
  - 2: at the center of the holder
  - 3: at the beginning of the second clamp block
  - 4: at the beginning of the bolt head hole for the central bolt
  - 5: at the beginning of the bolt head hole for the linguide bolts
  - 6: at the front side of the holder
  - 7: at the beginning of the hole for the porta
  - 8: at the inner side of the porta thruhole
  - 9: at the center of the porta
  - 10: at the outer side of the porta thruhole
  - 11: at the end of the porta
  - 12: at the end of the piece
- **pos\_w** (*int*) – Location of pos along the axis\_w (0-7) symmetrical
  - 0: at the center of symmetry
  - 1: at the first bolt column
  - 2: at the second bolt column
  - 3: at the third bolt column

- 4: at the inner side of the clamp post (larger circle)
- 5: at the outer side of the clamp post (smaller circle)
- 6: at the inner side of the clamp rails
- 7: at the end of the piece
- **pos\_h** (*int*) – Location of pos along the axis\_h (0-8)
  - 0: at the bottom (base)
  - 1: at the base for the porta
  - 2: at the top of the base
  - 3: first row of bolts
  - 4: second row of bolts
  - 5: third row of bolts
  - 6: 4th row of bolts
  - 7: at the base of the belt clamp
  - 8: at the middle of the belt clamp
  - 9: at the top of the piece
- **pos** (*FreeCAD.Vector*) – Position of the cylinder, taking into account where the center is

---

**Note:** All the parameters and attributes of parent class SinglePart

---

#### **Dimensional attributes**

##### **filt\_hole\_d**

depth of the hole for the filter (for filter\_w)

**Type** float

##### **filt\_hole\_w**

width of the hole for the filter (for filter\_l)

**Type** float

##### **filt\_hole\_h**

height of the hole for the filter (for filter\_t)

**Type** float

##### **beltclamp\_blk\_t**

thickness (along axis\_d) of each of the belt clamp blocks

**Type** float

##### **beltpost\_l**

length of the belt post (that has a shap of 2 circles and the tangent

**Type** float

##### **lr\_beltpost\_r**

radius of the larger belt post (it has a belt shape)

**Type** float



# **clamp\_lrbeltpostcen\_dist**

distance from the center of the larger belt post cylinder to the clamp post

**Type** float

# **prnt\_ax**

Best axis to print (normal direction, pointing upwards)

**Type** FreeCAD.Vector

# **d0\_cen**

**Type** int

# **w0\_cen**

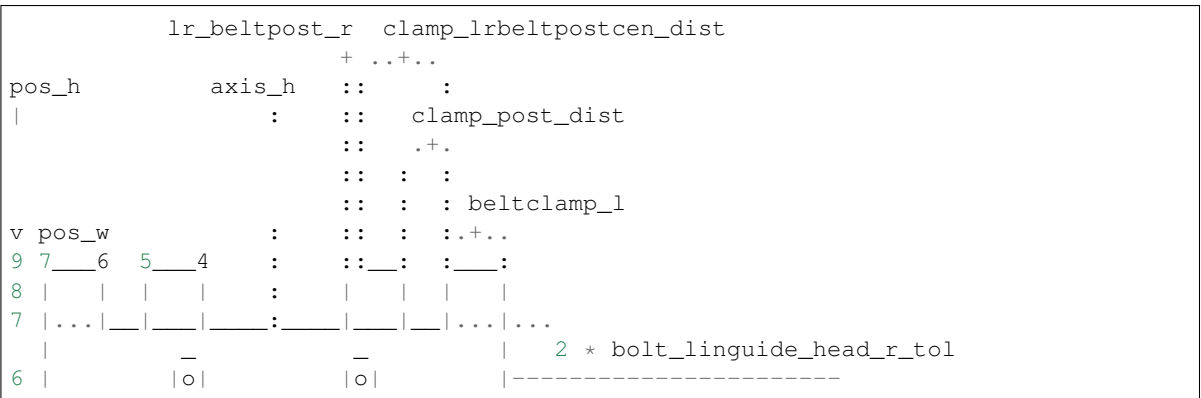
**Type** int

# **h0\_cen**

indicates if pos\_h = 0 (pos\_d, pos\_w) is at the center along axis\_h, axis\_d, axis\_w, or if it is at the end.

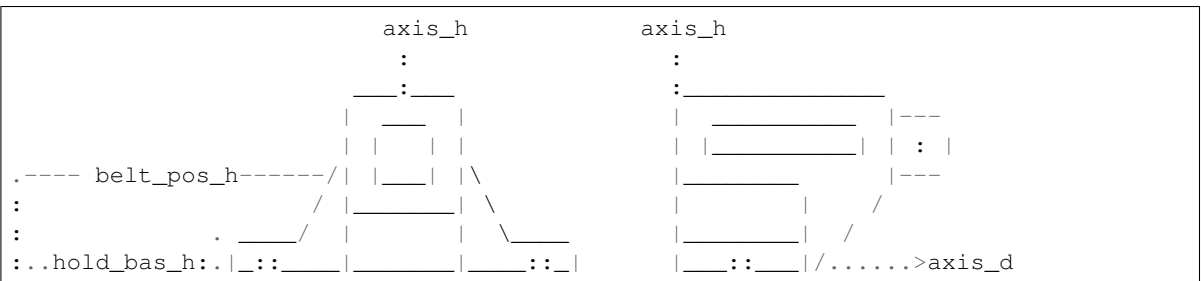
- 1 : at the center (symmetrical, or almost symmetrical)
- 0 : at the end

**Type** int



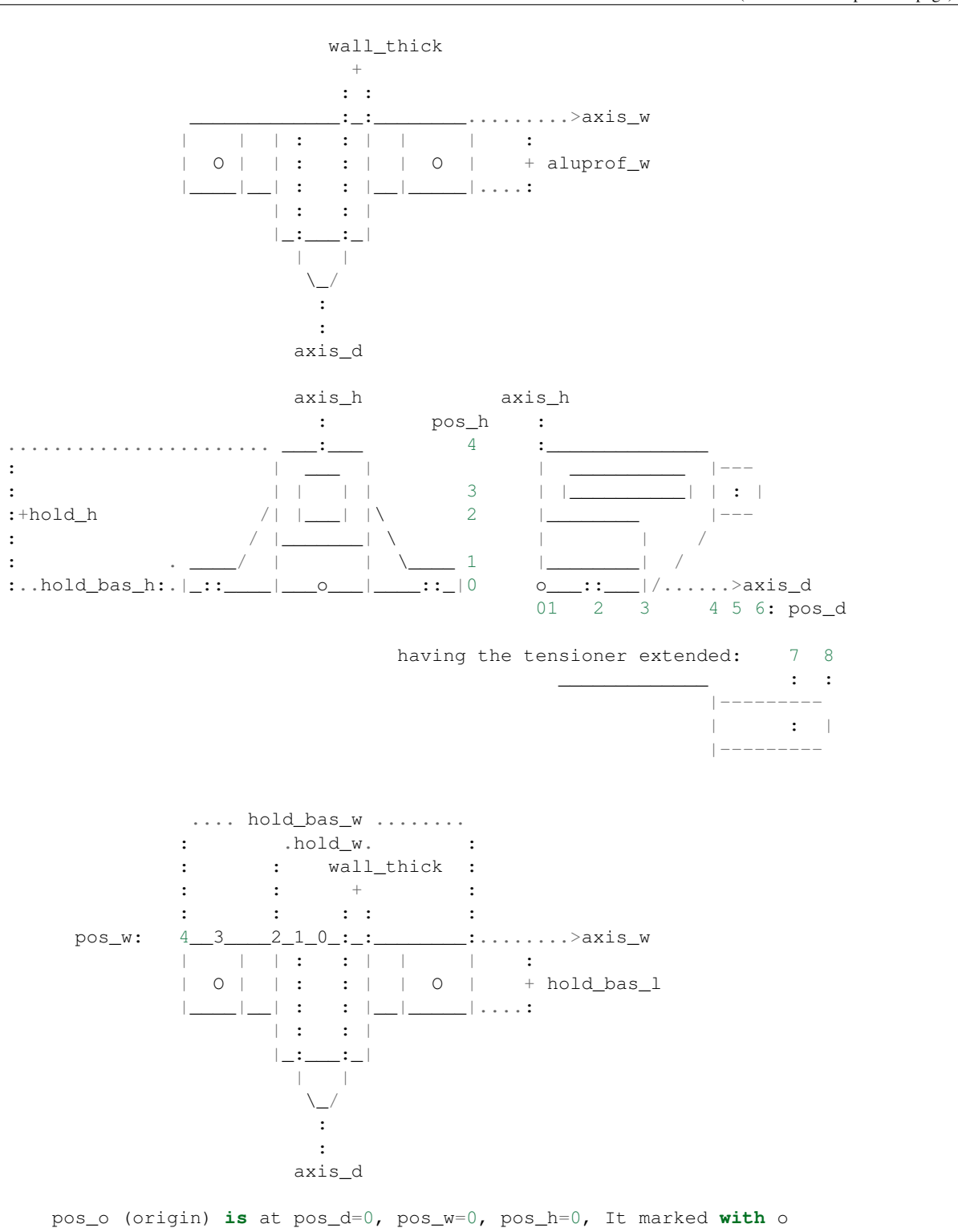
```
class tensioner_cls.TensionerSet (aluprof_w=20.0,    belt_pos_h=20.0,    hold_bas_h=0,
                                hold_hole_2sides=0,    boltidler_mtr=3,    bolttens_mtr=3,
                                boltaluprof_mtr=3,    tens_stroke=20.0,    wall_thick=3.0,
                                in_fillet=2.0,    pulley_stroke_dist=0,    nut_holder_thick=4.0,
                                opt_tens_chmf=1,    min_width=0,    tol=0.4,
                                axis_d=FreeCAD.Vector,    axis_w=FreeCAD.Vector,
                                axis_h=FreeCAD.Vector,    pos_d=0,    pos_w=0,    pos_h=0,
                                pos=FreeCAD.Vector,    group=0,    name="")
```

Set composed of the idler pulley and the tensioner



(continues on next page)

(continued from previous page)



## Parameters

- **aluprof\_w** (*float*) – Width of the aluminum profile

- **belt\_pos\_h** (*float*) – The position along axis\_h where the idler pulley that conveys the belt starts. THIS POSITION IS CENTERED at the idler pulley
- **tens\_h** (*float*) – Height of the idler tensioner
- **tens\_w** (*float*) – Width of the idler tensioner
- **tens\_d\_inside** (*float*) – Max length (depth) of the idler tensioner that is inside the holder
- **wall\_thick** (*float*) – Thickness of the walls
- **in\_fillet** (*float*) – Radius of the inner fillets
- **boltaluprof\_mtr** (*float*) – Diameter (metric) of the bolt that attaches the tensioner holder to the aluminum profile (or whatever is attached to)
- **bolt\_tens\_mtr** (*float*) – Diameter (metric) of the bolt for the tensioner
- **hold\_base\_h** (*float*) – Height of the base of the tensioner holder if 0, it will take wall\_thick
- **opt\_tens\_chmf** (*int*) –
  - 1: there is a chamfer at every edge of tensioner, inside the holder
  - 0: there is a chamfer only at the edges along axis\_w, not along axis\_h
- **hold\_hole\_2sides** (*int*) – In the tensioner holder there is a hole to see inside, it can be at each side of the holder or just on one side
  - 0: only at one side
  - 1: at both sides
- **min\_width** (*int*) – Make the rim the minimum: the diameter of the washer
  - 0: normal width: the width of the aluminum profile
  - 1: minimum width: diameter of the washer
- **tol** (*float*) – Tolerances to print
- **axis\_d** (*FreeCAD.Vector*) – Depth vector of coordinate system
- **axis\_w** (*FreeCAD.Vector*) – Width vector of coordinate system if V0: it will be calculated using the cross product: axis\_l x axis\_h
- **axis\_h** (*FreeCAD.Vector*) – Height vector of coordinate system
- **pos\_d** (*int*) – Location of pos along the axis\_d
  - 0: at the back of the holder
  - 1: at the place where the tensioner can reach all the way inside
  - 2: at the center of the base along axis\_d, where the bolts to attach
  - the holder base to the aluminum profile
  - 3: at the end of the base
  - 4: at the end of the holder
  - 5: at the center of the pulley
  - 6: at the end of the idler tensioner
  - 7: at the center of the pulley, when idler is all the way out

- 8: at the end of the idler tensioner, when it is all the way out
- **pos\_w** (*int*) – Location of pos along the axis\_w
  - 0: at the center of symmetry
  - 1: at the inner walls of the holder, which is the pulley radius
  - 2: at the end of the holder (the top part, where the base starts)
  - 3: at the center of the bolt holes to attach the holder base to the aluminum profile
  - 4: at the end of the piece along axis\_w axes have direction. So if pos\_w == 3, the piece will be drawn along the positive side of axis\_w
- **pos\_h** (*int*) – Location of pos along the axis\_h (0,1,2,3,4)
  - 0: at the bottom of the holder
  - 1: at the top of the base of the holder (for the bolts)
  - 2: at the bottom of the hole where the idler tensioner goes
  - 3: at the middle point of the hole where the idler tensioner goes
  - 4: at the top of the holder
- **pos** (*FreeCAD.Vector*) – position of the piece
- **for the set** (*Parameters*) –
- **tens\_in\_ratio** (*float*) – from 0 to 1, the ratio of the stroke that the tensioner is inside.
  - if 1: it is all the way inside
  - if 0: it is all the way outside (all the tens\_stroke)

---

**Note:** All the parameters and attributes of father class SinglePart

---

**prnt\_ax**

Best axis to print (normal direction, pointing upwards)

**Type** FreeCAD.Vector

**d0\_cen**

**Type** int

**w0\_cen**

**Type** int

**h0\_cen**

indicates if pos\_h = 0 (pos\_d, pos\_w) is at the center along axis\_h, axis\_d, axis\_w, or if it is at the end.

- 1 : at the center (symmetrical, or almost symmetrical)
- 0 : at the end

**Type** int

**tot\_d**

total depth, including the idler tensioner

**Type** float

**tot\_d\_extend**

total depth including the idler tensioner, having it extended

Type float

```
class partset.NemaMotorPulleySet (nema_size=17, base_l=32.0, shaft_l=24.0, shaft_r=0,
    circle_r=11.0, circle_h=2.0, chmf_r=1, rear_shaft_l=0,
    bolt_depth=3.0, pulley_pitch=2.0, pulley_n_teeth=20,
    pulley_toothed_h=7.5, pulley_top_flange_h=1.0,
    pulley_bot_flange_h=0, pulley_tot_h=16.0, pul-
    ley_flange_d=15.0, pulley_base_d=15.0, pulley_tol=0,
    pulley_pos_h=- 1, axis_d=FreeCAD.Vector, axis_w=None,
    axis_h=FreeCAD.Vector, pos_d=0, pos_w=0, pos_h=1,
    pos=FreeCAD.Vector, group=1, name="")
```

Set composed of a Nema Motor and a pulley

Number positions of the pulley will be after the positions of the motor

```
axis_h
:
:
|_____|_____|.....11 <-> 5
|_____|_____|.....10 <-> 4
|   |   |   |   |
|   |   |   |   |.....9 <-> 3
|   |   |   |   |
|_____|_____|.....8 <-> 2
|_____|_____|.....7 <-> 1
|   |   |   |   |
|   |   |   |   |
|_____|_____|.....6 <-> 0 (for the pulley)
:   :   :
:   :   :
0...56789.....axis_d, axis_w
|
01 23456 (for the pulley)

axis_h
:
:
2 .....
| |
| | ..... + shaft_l
| | .....
|_____|_____|.....circle_h.:
| :: 3 :: | :
| :
| :
| : + base_l
| :
| :
|_____|_____|.....4
: :
: :
: : :+ rear_shaft_l (optional)
:5: :
```

(continues on next page)

(continued from previous page)

```

01...2...3...4.....:.....axis_d (same as axis_w)
|   |   |   |
|   |   |   v
|   |   | end of the motor
|   |   | v
|   |   | bolt holes
|   |   | v
|   |   | radius of the circle (cylinder)
|   |   | v
|   |   | radius of the shaft

axis_w
:
:
:-----:.....
/         \.....: chmf_r
|  O             O  |
|         -         |
|         .   .     |
|      (  ( )  )     |.....axis_d
|         .   .     |
|         -         |
|  O             O  |
\         /
:             :
:.....:
+
motor_w (same as d): Nema size in inches /10
pos_o (origin) is at pos_d=0, pos_w=0, pos_h=1

```

## Parameters

- **nema\_size** (*dict*) – List of sizes defines in kcomps NEMA motor dimensions.
- **base\_l** (*float*,) – Length (height) of the base
- **shaft\_l** (*float*,) – Length (height) of the shaft, including the small cylinder (circle) at the base
- **shaft\_r** (*float*,) – Radius of the shaft, if not defined, it will take the dimension defined in kcomp
- **circle\_r** (*float*,) – Radius of the cylinder (circle) at the base of the shaft if 0 or circle\_h = 0 -> no cylinder
- **circle\_h** (*float*,) – Height of the cylinder at the base of the shaft if 0 or circle\_r = 0 -> no cylinder
- **chmf\_r** (*float*,) – Chamfer radius of the chamfer along the base length (height)
- **rear\_shaft\_l** (*float*) – Length of the rear shaft, 0 : no rear shaft
- **bolt\_depth** (*float*) – Depth of the bolt holes of the motor
- **pulley\_pitch** (*float/int*) – Distance between teeth: Typically 2mm, or 3mm
- **pulley\_n\_teeth** (*int*) – Number of teeth of the pulley

- **pulley\_toothed\_h** (*float*) – Height of the toothed part of the pulley
- **pulley\_top\_flange\_h** (*float*) – Height (thickness) of the top flange, if 0, no top flange
- **pulley\_bot\_flange\_h** (*float*) – Height (thickness) of the bot flange, if 0, no bottom flange
- **pulley\_tot\_h** (*float*) – Total height of the pulley
- **pulley\_flange\_d** (*float*) – Flange diameter, if 0, it will be the same as the base\_d
- **pulley\_base\_d** (*float*) – Base diameter
- **pulley\_tol** (*float*) – Tolerance for radius (it will subtracted to the radius) twice for the diameter. Or added if a shape to subtract
- **pulley\_pos\_h** (*float*) – position in mm of the pulley along the shaft
  - 0: it is at the base of the shaft
  - -1: the top of the pulley will be aligned with the end of the shaft
- **axis\_d** (*FreeCAD.Vector*) – Depth vector of coordinate system (perpendicular to the height)
- **axis\_w** (*FreeCAD.Vector*) – Width vector of coordinate system if V0: it will be calculated using the cross product: axis\_h x axis\_d
- **axis\_h** (*FreeCAD.Vector*) – Height vector of coordinate system
- **pos\_d** (*int*) – location of pos along the axis\_d see drawing
  - Locations coinciding with the motor
    - \* 0: at the axis of the shaft
    - \* 1: at the radius of the shaft
    - \* 2: at the end of the circle(cylinder) at the base of the shaft
    - \* 3: at the bolts
    - \* 4: at the end of the piece
  - Locations of the pulley
    - \* 5: at the inner radius
    - \* 7: at the external radius
    - \* 7: at the pitch radius (outside the toothed part)
    - \* 8: at the end of the base (not the toothed part)
    - \* 9: at the end of the flange (V0 is no flange)
- **pos\_w** (*int*) – location of pos along the axis\_w see drawing
  - Same locations of pos\_d
- **pos\_h** (*int*) – location of pos along the axis\_h, see drawing
  - 0: at the base of the shaft (not including the circle at the base of the shaft)
  - 1: at the end of the circle at the base of the shaft
  - 2: at the end of the shaft
  - 3: at the end of the bolt holes

- 4: at the bottom base
- 5: at the end of the rear shaft, if no rear shaft, it will be the same as pos\_h = 4
- 6: at the base of the pulley
- 7: at the base of the bottom flange of the pulley
- 8: at the base of the toothed part of the pulley
- 9: at the center of the toothed part of the pulley
- 10: at the end (top) of the toothed part of the pulley
- 11: at the end (top) of the pulley of the pulley
- **pos** (*FreeCAD.Vector*) – Position of the model
- **name** (*str*) – Object name

```
class beltcl.BeltClamp(fc_fro_ax, fc_top_ax, base_h=2, base_l=0, base_w=0, bolt_d=3,
                        bolt_csunk=0, ref=1, pos=FreeCAD.Vector, extra=1, wfco=1, intol=0,
                        name='belt_clamp')
```

Similar to shp\_topbeltclamp, but with any direction, and can have a base Creates a shape of a belt clamp. Just the rail and the cylinder and may have a rectangular base just one way: 2 clamp blocks It is referenced on the base of the clamp, but it may have 5 different positions

#### Parameters

- **fc\_fro\_ax** (*FreeCAD.Vector*) – FreeCAD.Vector pointing to the front, see picture
- **fc\_top\_ax** (*FreeCAD.Vector*) – FreeCAD.Vector pointing to the top, see picture
- **base\_h** (*float*) – Height of the base,
  - if 0 and bolt\_d=0: no base
  - if 0 and bolt\_d!= 0: minimum base to have the bolt head and not touching with the belt (countersunk) if bolt\_csunk > 0
- **base\_l** (*float*) – Length of the base, if base\_h not 0.
  - if 0 and bolt\_d=0: will have the minimum length, defined by the clamp
  - if 0 and bolt\_d!=0: will have the minimum length, defined by the clamp plus the minimum separation due to the bolt holes
- **base\_w** (*float*) – Width of the base, if base\_h not 0.
- **bolt\_d** (*float*) – Diameter of the bolts, if zero, no bolts
- **bolt\_csunk** (*float*) – If the bolt is countersunk
  - if >0: there is a whole to countersink the head of the bolt there will be an extra height if not enough bolt\_d has to be > 0
  - if 0: no whole for the height, and no extra height
  - if >0, the size will determine the minimum height of the base, below the countersink hole
- **ref** (*int*) – Reference of the position (see picture below)
- **extra** (*float*) – If extra, it will have an extra height below the zero height, this is to be joined to some other piece
- **wfco** (*int*) –
  - if 1: With FreeCad Object: a freecad object is created



(continues on next page)

(continued from previous page)

```

fc_fro_ax...|:::_____*:__::_|....+ bolt_csunk (if not 0)

```

```
class beltcl.DoubleBeltClamp(axis_h=FreeCAD.Vector, axis_d=FreeCAD.Vector,  
                             axis_w=FreeCAD.Vector, base_h=2, base_l=0, base_w=0,  
                             bolt_d=3, bolt_csunk=0, ref=1, pos=FreeCAD.Vector, extra=1,  
                             wfco=1, intol=0, name='double_belt_clamp')
```

Similar to BeltClamp, but in two ways Creates a shape of a double belt clamp. positions

## Parameters

- **fc\_fro\_ax** (*FreeCAD.Vector*) – FreeCAD.Vector pointing to the front, see picture
- **fc\_top\_ax** (*FreeCAD.Vector*) – FreeCAD.Vector pointing to the top, see picture
- **base\_h** (*float*) – Height of the base,
  - if 0 and bolt\_d=0: no base
  - if 0 and bolt\_d!= 0: minimum base to have the bolt head and not touching with the belt (countersunk) if bolt\_csunk > 0
- **base\_l** (*float*) – Length of the base, if base\_h not 0.
  - if 0 and bolt\_d=0: will have the minimum length, defined by the clamp
  - if 0 and bolt\_d!=0: will have the minimum length, defined by the clamp plus the minimum separation due to the bolt holes
- **base\_w** (*float*) – Width of the base, if base\_h not 0.
- **bolt\_d** (*float*) – Diameter of the bolts, if zero, no bolts
- **bolt\_csunk** (*float*) – If the bolt is countersunk
  - if >0: there is a whole to countersink the head of the bolt there will be an extra height if not enough bolt\_d has to be > 0
  - if 0: no whole for the height, and no extra height
  - if >0, the size will determine the minimum height of the base, below the countersink hole
- **ref** (*int*) – Reference of the position (see picture below)
- **extra** (*float*) – If extra, it will have an extra height below the zero height, this is to be joined to some other piece
- **wfco** (*int*) –
  - if 1: With FreeCad Object: a freecad object is created
  - if 0: only the shape
- **intol** (*float*) – Internal extra tolerance to the dimension CB\_IW, subtracting to CB\_W. If negative, makes CB\_IW smaller.
- **name** (*str*) – Name of the freecad object, if created

[illegible]

(continues on next page)

```
axis_d...|_::_____*_|_____|_*_____::|.:
```

```
clamp2end clamp2end
...+... ...+...
: : : :
: bolt2end : bolt2end
:+...+: : +...+:
: : : :
: : : :
: : : :
CB_W { : | | / \ | | :
CB_IW { | 0 | | * | | * | | 0 | CCYL: CLAMPCYL +
base_w
CB_W { | | | \ / | \ / | | :
: : : : : :
: : : : : O :
: :CB_L:.CS...: | | :
: + :
: CCYL_R :
: ..... base_l ..... :
```

```
      | |
      | O |
      | |
CB_W { | | | / \ | / \ | | |
CB_IW { -4 -3 -2 | -1 | 0 | 1 | | 2 3 4
CB_W { | | | \ / | \ / | | |
      | |
      | O |
      | |
```

```
1: center
2: center of left cylinder
3: center of right cylinder
4: front of the left clamps
5: front of the right clamps
6: left bolt
7: right bolt
8: left base
9: right base
```

```
      3
      | |
      | 2 |
      | |
CB_W { | | . . / \ . 1 . / \ . . | |
CB_IW { | | | 0 | | |
CB_W { | | . . \ / . -1 . \ / . . | |
      | |
      | -2 |
      | -3 |
```

---

(continues on next page)

(continued from previous page)

```

1: center
2: bolt up
3: bolt down
4: up base
5: down base

```

```

class fc_cls.Din934Nut (metric,axis_d_apo=0,h_offset=0,axis_h=FreeCAD.Vector,axis_d=None,
                        axis_w=None, pos_h=0, pos_d=0, pos_w=0, pos=FreeCAD.Vector,
                        model_type=0, name="")

```

Din 934 Nut

**Parameters**

- **metric** (*int* (maybe *float*: 2.5)) –
- **axis\_h** (*FreeCAD.Vector*) –
- **axis\_d\_apo** (*int*) –
  - 0: default: axis\_d points to the vertex
  - 1: axis\_d points to the center of a side
- **h\_offset** (*float*) – Distance from the top, just to place the Nut, see pos\_h if negative, from the bottom
  - 0: default
- **axis\_h** – Vector along the axis, height
- **axis\_d** (*FreeCAD.Vector*) – Vector along the first vertex, a direction perpendicular to axis\_h. It is not necessary if pos\_d == 0. It can be None, but if None, axis\_w has to be None
- **axis\_w** (*FreeCAD.Vector*) – Vector along the cylinder radius, a direction perpendicular to axis\_h and axis\_d. It is not necessary if pos\_w == 0. It can be None
- **pos\_h** (*int*) – Location of pos along axis\_h
  - 0: at the center
  - -1: at the base
  - 1: at the top
  - -2: at the base + h\_offset
  - 2: at the top + h\_offset
- **pos\_d** (*int*) – Location of pos along axis\_d (-2, -1, 0, 1, 2)
  - 0: pos is at the circumference center (axis)
  - 1: pos is at the inner circumsference, on axis\_d, at r\_in from the circle center
  - 2: pos is at the apothem, on axis\_d
  - 3: pos is at the outer circumsference, on axis\_d, at r\_out from the circle center
- **pos\_w** (*int*) – Location of pos along axis\_w (-2, -1, 0, 1, 2)
  - 0: pos is at the circumference center
  - 1: pos is at the inner circumsference, on axis\_w, at r\_in from the circle center
  - 2: pos is at the apothem, on axis\_w
  - 3: pos is at the outer circumsference, on axis\_w, at r\_out from the center

- **pos** (*FreeCAD.Vector*) – Position of the prism, taking into account where the center is
- **model\_type** (*0*) – Not to print, just an outline
- **name** (*str*) – Name of the bolt

```
class fc_clss.Din125Washer (metric, axis_h, pos_h, tol=0, pos=FreeCAD.Vector, model_type=0,  
                             name="")
```

Din 125 Washer, this is the regular washer

#### Parameters

- **metric** (*int (maybe float: 2.5)*) –
- **axis\_h** (*FreeCAD.Vector*) – Vector along the cylinder height
- **pos\_h** (*int*) – Location of pos along axis\_h (0,1)
  - 0: the cylinder pos is at its base
  - 1: the cylinder pos is centered along its height
- **tol** (*float*) – Tolerance for the inner and outer radius. It is the tolerance for the diameter, so the radius will be added/subs have of this tolerance.
  - tol will be added to the inner radius (so it will be larger).
  - tol will be subtracted to the outer radius (so it will be smaller).
- **model\_type** (*int*) – Type of model:
  - 0: exact
  - 1: outline
- **pos** (*FreeCAD.Vector*) – Position of the cylinder, taking into account where the center is

---

**Note:** All the parameters and attributes of father class CylHole

---

#### **metric**

Metric of the washer

**Type** int or float (in case of M2.5) or even str for inches ?

#### **model\_type**

**Type** int

```
class fc_clss.Din9021Washer (metric, axis_h, pos_h, tol=0, pos=FreeCAD.Vector, model_type=0,  
                             name="")
```

Din 9021 Washer, this is the larger washer

#### Parameters

- **metric** (*int (maybe float: 2.5)*) –
- **axis\_h** (*FreeCAD.Vector*) – Vector along the cylinder height
- **pos\_h** (*int*) – Location of pos along axis\_h (0,1)
  - 0: the cylinder pos is at its base
  - 1: the cylinder pos is centered along its height
- **tol** (*float*) – Tolerance for the inner and outer radius. It is the tolerance for the diameter, so the radius will be added/subs have of this tolerance

- tol will be added to the inner radius (so it will be larger)
- tol will be subtracted to the outer radius (so it will be smaller)
- **model\_type** (*int*) – Type of model:
  - 0: exact
  - 1: outline
- **pos** (*FreeCAD.Vector*) – Position of the cylinder, taking into account where the center is

---

**Note:** All the parameters and attributes of father class CylHole

---

**metric**

Metric of the washer

**Type** int or float (in case of M2.5) or even str for inches ?

**model\_type**

**Type** int

```
class fc_cls.Din912Bolt (metric, shank_l, shank_l_adjust=0, shank_out=0, head_out=0,
                        axis_h=FreeCAD.Vector, axis_d=None, axis_w=None, pos_h=0,
                        pos_d=0, pos_w=0, pos=FreeCAD.Vector, model_type=0, name="")
```

Din 912 bolt. hex socket bolt

#### Parameters

- **metric** (*int* (may be float: 2.5) –
- **shank\_l** (*float*) – length of the bolt, not including the head
- **shank\_l\_adjust** (*int*) –
  - 0: shank length will be the size of the parameter shank\_l
  - -1: shank length will be the size of the closest shorter or equal to shank\_l available lengths for this type of bolts
  - 1: shank length will be the size of the closest larger or equal to shank\_l available lengths for this type of bolts
- **shank\_out** (*float*) – Distance to the end of the shank, just for positioning, it doesnt change shank\_l
  - 0: default

---

**Note:** I dont think it is necessary, but just in case

---

- **head\_out** (*float*) – Distance to the end of the head, just for positioning, it doesnt change head\_l
  - 0: default

---

**Note:** I dont think it is necessary, but just in case

---

- **axis\_h** (*FreeCAD.Vector*) – Vector along the axis of the bolt, pointing from the head to the shank

- **axis\_d** (*FreeCAD.Vector*) – Vector along the radius, a direction perpendicular to axis\_h If the head is hexagonal, the direction of one vertex
- **axis\_w** (*FreeCAD.Vector*) – Vector along the cylinder radius, a direction perpendicular to axis\_h and axis\_d. It is not necessary if pos\_w == 0. It can be None
- **pos\_h** (*int*) – Location of pos along axis\_h
  - 0: top of the head, considering head\_out,
  - 1: position of the head not considering head\_out if head\_out = 0, it will be the same as pos\_h = 0
  - 2: end of the socket, if no socket, will be the same as pos\_h = 0
  - 3: union of the head and the shank
  - 4: where the screw starts, if all the shank is screwed, it will be the same as pos\_h = 2
  - 5: end of the shank, not considering shank\_out
  - 6: end of the shank, if shank\_out = 0, will be the same as pos\_h = 5
  - 7: top of the head, considering xtr\_head\_l, if xtr\_head\_l = 0 will be the same as pos\_h = 0
- **pos\_d** (*int*) – Location of pos along axis\_d (symmetric)
  - 0: pos is at the central axis
  - 1: radius of the shank
  - 2: radius of the head
- **pos\_w** (*int*) – Location of pos along axis\_d (symmetric)
  - 0: pos is at the central axis
  - 1: radius of the shank
  - 2: radius of the head
- **pos** (*FreeCAD.Vector*) – Position of the bolt, taking into account where the pos\_h, pos\_d, pos\_w are
- **model\_type** (*0*) – Not to print, just an outline
- **name** (*str*) – Name of the bolt

## Optical

```
comp_optic.f_breadboard(d_breadboard, length, width, cl=1, cw=1, ch=1,
                        fc_dir_h=FreeCAD.Vector, fc_dir_w=FreeCAD.Vector,
                        pos=FreeCAD.Vector, name='breadboard')
```

### Parameters

- **d\_breadboard** (*dict*) – Dictionary with the values
- **length** (*float*) –
- **width** (*float*) –
- **cl** (*int*) –
  - 1: the length dimension is centered

- 0: it is not centered
- **cw**(*int*) –
  - 1: the width dimension is centered
  - 0: it is not centered
- **ch**(*int*) –
  - 1: the height dimension is centered
  - 0: it is not centered
- **fc\_dir\_h**(*FreeCAD.Vector*) – Vector with the direction of the height
- **fc\_dir\_w**(*FreeCAD.Vector*) – Vector with the direction of the width
- **pos**(*FreeCAD.Vector*) – Placement of the model
- **name**(*str*) – object name

**Returns** Object with the shape of a BreadBoard

**Return type** FreeCAD Object

`comp_optic.f_cagecube(d_cagecube, axis_thru_rods='x', axis_thru_hole='y', name='cagecube',  
toprint_tol=0)`

Creates a cage cube, it creates from a dictionary

#### Parameters

- **d\_cagecube** – Dictionary with the dimensions of the cage cube, defined in `kcomp_optic.py`
- **axis\_thru\_rods**(*str*) – Direction of rods: 'x', 'y', 'z'
- **axis\_thru\_hole**(*str*) – Direction big thru\_hole: 'x', 'y', 'z'.

---

**Note:** Cannot be the same as `axis_thru_rods` There are 6 possible orientations: Thru-rods can be on X, Y or Z axis thru-hole can be on X, Y, or Z axis, but not in the same as thru-rods

---

- **toprint\_tol**(*float*) –
  - 0, dimensions as they are.
  - >0 value of tolerances of the holes. multiplies the normal tolerance in `kcomp.TOL`

#### Returns

- *CageCube*. The freeCAD object can be accessed by the
- attribute *fco*

`comp_optic.f_cagecubehalf(d_cagecubehalf, axis_1='x', axis_2='y', name='cagecubehalf')`

Creates a half cage cube: 2 perpendicular sides, and a 45 degree angle side. It creates from a dictionary

#### Parameters

- **d\_cagecubehalf**(*dict*) – Dictionary with the dimensions of the cage cube, defined in `kcomp_optic.py`
- **axis\_1**(*str*) – Direction of the first right side: 'x', 'y', 'z', '-x', '-y', '-z'



- Note:** Cannot be the same as axis\_1, or its negated. Has to be perpendicular There are 24 possible orientations: 6 possible axis\_1 and 4 axis\_2 for each axis\_1

- ```
class comp_optic.Lb1cPlate(d_plate, fc_axis_h=FreeCAD.Vector, fc_axis_l=FreeCAD.Vector,  
                           ref_in=1, pos=FreeCAD.Vector, name='lb1c_plate')
```
- Creates a LB1C/M plate from thorlabs. The plate is centered

```

fc_axis_1: axis on the large separation
|
|
|-- sy_hole_sep -:
|      :
|cbore_hole_sep_s:
|      :      :      :
|
+-----+
|   O       O   | -----
| 0         0    | ....          :
|               |           :
|               |           :
|      ( )      | +sym_hole_sep + cbore_hole_sep_1
|               |           :
| 0         0    | ....          :
|   O       O   | -----
|_____+_____|
|
| :: : ::::: | fc_axis_h
|__::__H_____| if ref_in=1 | -> h=0
|
|
if ref_in=0 -> h=0
| :: : ::::: |
|__::__H_____| V fc_axis_h

```

## Parameters

- **d\_plate** (*dict*) – Dictionary with the dimensions
- **fc\_axis\_h** (*FreeCAD.Vector*) – Direction of the vertical (thickness)
- **fc\_axis\_l** (*FreeCAD.Vector*) – Direction of the large distance of the counterbored asymmetrical holes
- **ref\_in** (*int*) –
  - 1: fc\_axis\_h starts on the inside to outside of the plate
  - 0: fc\_axis\_h starts on the outside to inside of the plate
- **pos** (*FreeCAD.Vector*) – Position of the center. The center is on the center of the plate, but on the axis\_h can be in either side depending on ref\_in
- **name** (*str*) – Name

```
class comp_optic.Lb2cPlate (fc_axis_h, fc_axis_l, cl=1, cw=1, ch=0, pos=FreeCAD.Vector,
                             name='lb2c_plate')
```

Same as plate\_lb2c, but it creates an object.

**fc\_axis\_h: FreeCAD.Vector** Direction of the vertical (thickness)

**fc\_axis\_l: FreeCAD.Vector** Direction of the large distance of the counterbored asymmetrical holes

**cl: int**

- 1: centered on the fc\_axis\_l direction

**cw: int**

- 1: centered on the axis\_small direction (perpendicular to fc\_axis\_l and fc\_axis\_h)

**ch: int**

- 1: centered on the vertical direction (thickness)

**pos: FreeCAD.Vector**

Position of the center. The center is on the center of the plate, but on the axis\_h can be in either side depending on ref\_in

**name: str** Name

```
comp_optic.lcp01m_plate(d_lcp01m_plate={'L': 71.11999999999999, 'chamfer_r': 2.0, 'mhole_d': 4.0, 'mhole_depth': 6.5, 'sym_hole_d': 6.0, 'sym_hole_sep': 60.0, 'thick': 12.7, 'thruhole_d': 51.689}, fc_axis_h=FreeCAD.Vector, fc_axis_m=FreeCAD.Vector, fc_axis_p=FreeCAD.Vector, cm=1, cp=1, ch=1, pos=FreeCAD.Vector, wfco=1, name='LCP01M_PLATE')
```

Creates a lcp01m\_plate side.

It creates from a dictionary

#### Parameters

- **d\_lcp01m\_plate** (*dict*) – Dictionary with the dimensions of the plate defined in kcomp\_optic.py
- **fc\_axis\_h** (*FreeCAD.Vector*) – Direction of the vertical (thickness) from the inside of the plate
- **fc\_axis\_m** (*FreeCAD.Vector*) – Direction of the mounting hole goes in the mounting hole
- **fc\_axis\_p** (*FreeCAD.Vector*) – Perpendicular direction of axis\_h and axis\_m, only used if not centered on this axis
- **cm** (*int*) –
  - 1: centered on the fc\_axis\_m direction (point 2)
  - 0: it will be on the mounting hole (point 1)
- **cp** (*int*) –
  - 1: centered on the perpendicular direction of fc\_axis\_m and fc\_axis\_h, if 0, fc\_axis\_p needs to be defined
  - 0: points 5 (cm==0) or 6 (cm==1)
- **ch** (*int*) –
  - 1: centered on the vertical direction (thickness)
- **pos** (*FreeCAD.Vector*) – Position of the center.

- **wfco** (*int*) –
  - 1: a FreeCAD object is created
  - 0: only de shape is created
- **name** (*str*) – name of the freecad object (if created)

```
comp_optic.lcpb1m_base (d_lcpb1m_base={'d_lip': 2.5, 'd_mount': 8.9, 'd_tot': 15.2,
   'h_slot': 3.8, 'h_sup': 8.9, 'h_tot': 10.8, 'l_mbolt_d': 4.0,
   's_mholes_d': 3.0, 's_mholes_dist': 50.0, 'slot_d': 6.0, 'slot_dist':
   100.0, 'w_sup': 82.6, 'w_tot': 120.7}, fc_axis_d=FreeCAD.Vector,
   fc_axis_w=FreeCAD.Vector, fc_axis_h=FreeCAD.Vector, ref_d=1, ref_w=1,
   ref_h=1, pos=FreeCAD.Vector, wfco=1, toprint=0, name='Lcpb1mBase')
```

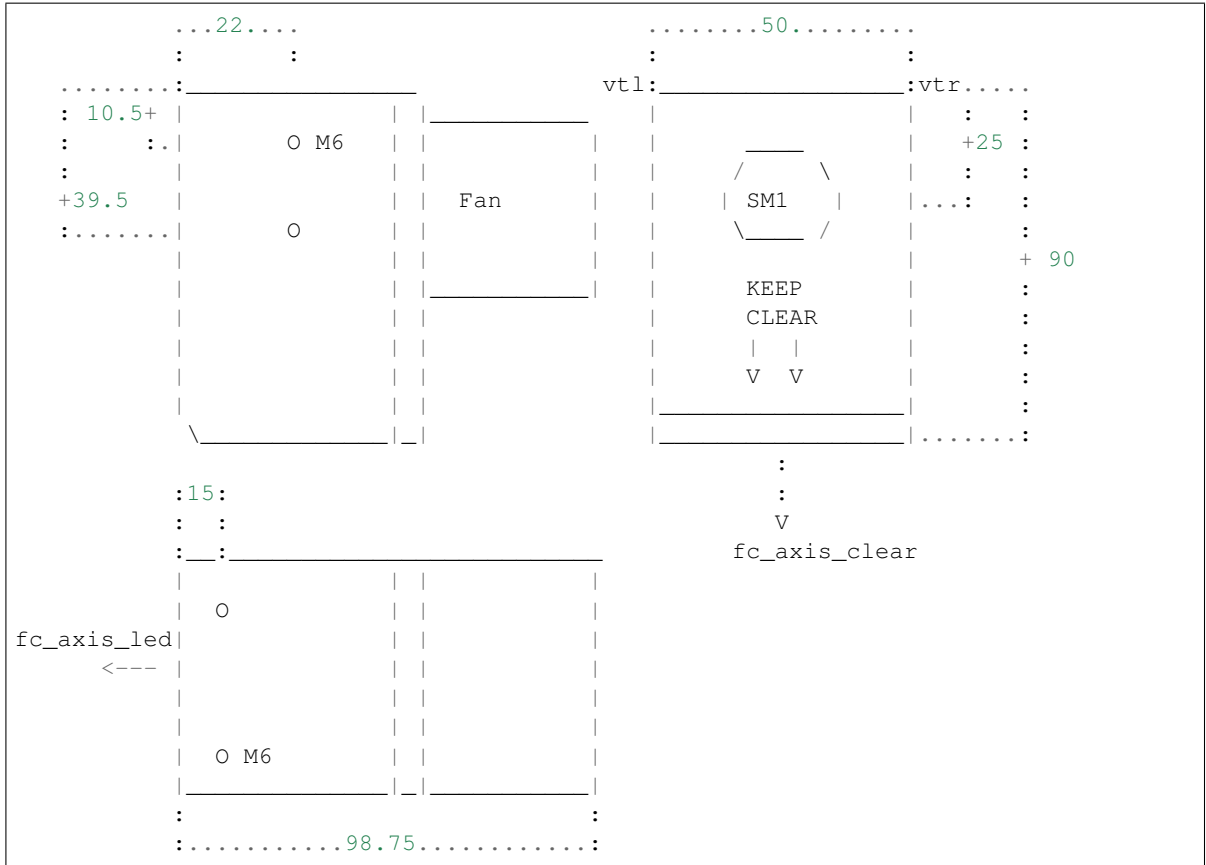
Creates a lcpb1m\_base for plates side, it creates from a dictionary

#### Parameters

- **d\_lcpb1m\_base** (*dict*) – Dictionary with the dimensions
- **fc\_axis\_d** (*FreeCAD.Vector*) – Direction of the deep
- **fc\_axis\_w** (*FreeCAD.Vector*) – Direction of the width
- **fc\_axis\_h** (*FreeCAD.Vector*) – Direction of the height
- **ref\_d** (*int*) – Position in the fc\_axis\_d:
  - 1: top side
  - 2: center
  - 3: lower side
- **ref\_w** (*int*) – Postion in the fc\_axis\_w:
  - 1: center
  - 2: center in left slot
  - 3: left side
- **ref\_h** (*int*) – Positionin the fc\_axis\_h:
  - 1: base
  - 2: top
- **pos** (*FreeCAD.Vector*) – Position of the center.
- **wfco** (*int*) –
  - 1: a FreeCAD object is created
  - 0: only de shape is created
- **toprint** (*int*) – 1 if you want to include a triangle between the shank and the head to support the shank and not building the head on the air using kcomp.LAYER3D\_H
- **name** (*str*) – Name of the freecad object (if created)

```
comp_optic.PrizLed (fc_axis_led=FreeCAD.Vector, fc_axis_clear=FreeCAD.Vector,
                    pos=FreeCAD.Vector, name='prizmatix_led')
```

Creates the shape of a Prizmatix UHP-T-Led The drawing is very rough, and the original drawing lacks many dimensions



### Parameters

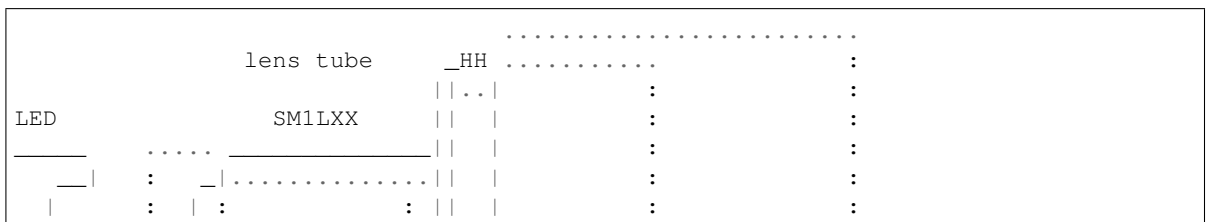
- **fc\_axis\_led** (*FreeCAD.Vector*) – Direction of the led
- **fc\_axis\_clear** (*FreeCAD.Vector*) – Direction of the arrows indicating to keep clear
- **pos** (*FreeCAD.Vector*) – Position of the LED, on the center of the SM1 thread
- **name** (*str*) – Object name

`comp_optic.SM1TubelensSm2 (sm1l_size, fc_axis=FreeCAD.Vector, ref_sm1=1, pos=FreeCAD.Vector, ring=1, name='tubelens_sm1_sm2')`

Creates a componente formed by joining: the lens tube SM1LXX + SM1A2 + SM2T2, so we have:

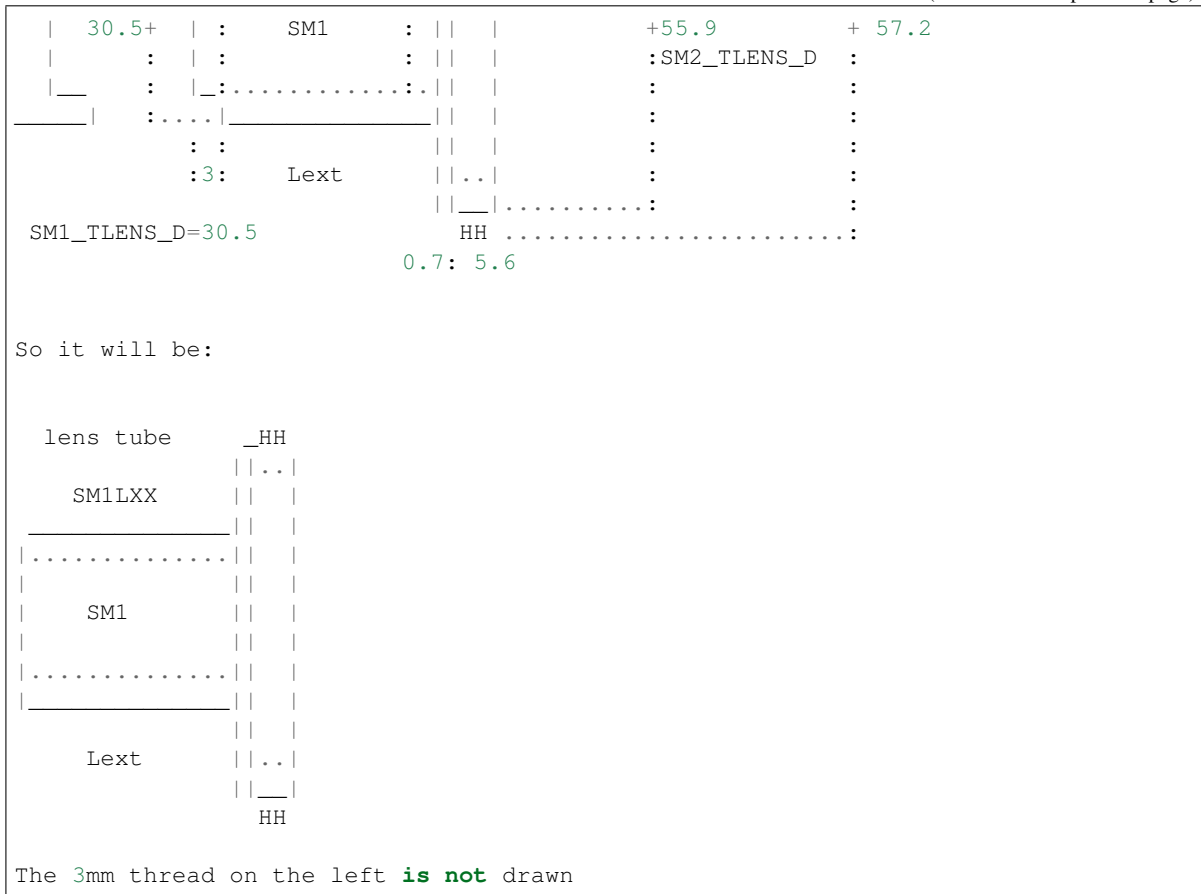
- on one side a SM1 external thread
- on the other side a SM2 external thread

And inside we have a SM1 tube lens Since there are threads, they may be inserted differently, so size may vary. Therefore, size are approximate, and also, details are not drawn, such as threads, or even the part that contains the thread is not drawn:



(continues on next page)

(continued from previous page)

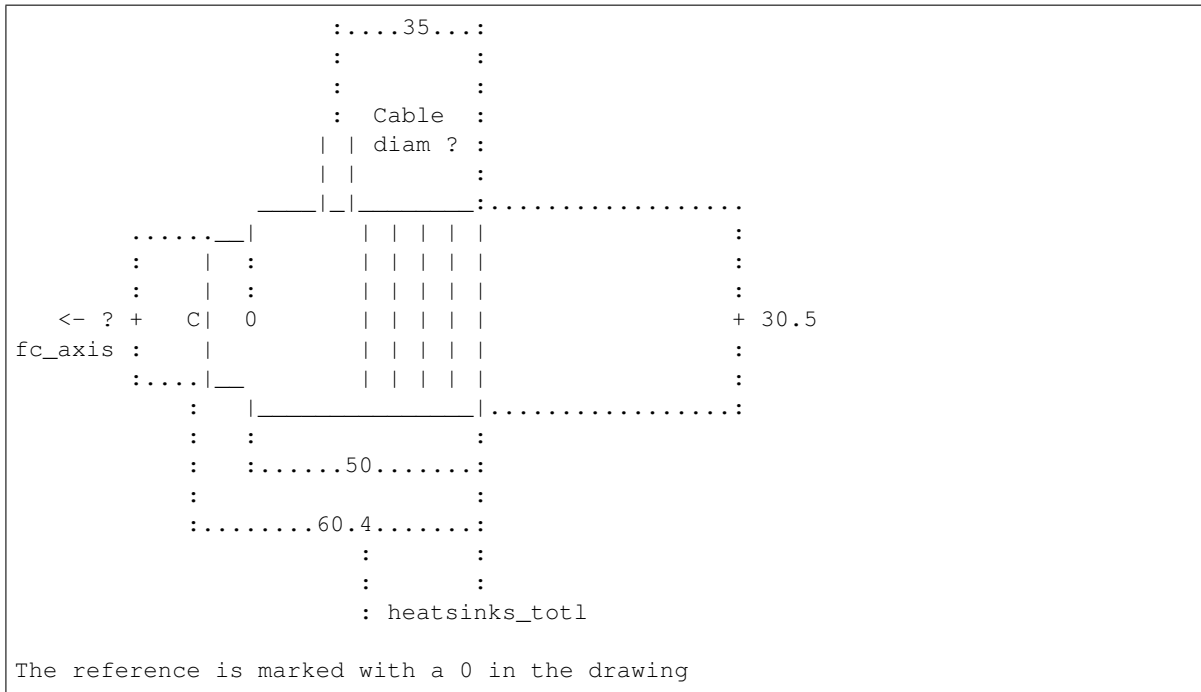


### Parameters

- **sm1l\_size** (*float*) – Length of the side of the cube (then it will be halved)
- **fc\_axis** (*FreeCAD.Vector*) – Direction of the tube lens: *FreeCAD.Vector*
- **axis\_2** (*FreeCAD.Vector*) – Direction big the other right side:
- **ref\_sm1** (*int*) –
  - 1: if the position is referred to the sm1 end
  - 0: if the position is referred to the ring end
- **pos** (*FreeCAD.Vector*) – Position of the object
- **ring** (*int*) –
  - 1: if there is ring
  - 0: there is no ring, so just the thread, at it is not drawn
- **name** (*str*) – Name of the freecad object

`comp_optic.ThLed30` (*fc\_axis=FreeCAD.Vector, fc\_axis\_cable=FreeCAD.Vector, pos=FreeCAD.Vector, name='thled30'*)

Creates the shape of a Thorlabs Led with 30.5 mm Heat Sink diameter The drawing is very rough



#### Parameters

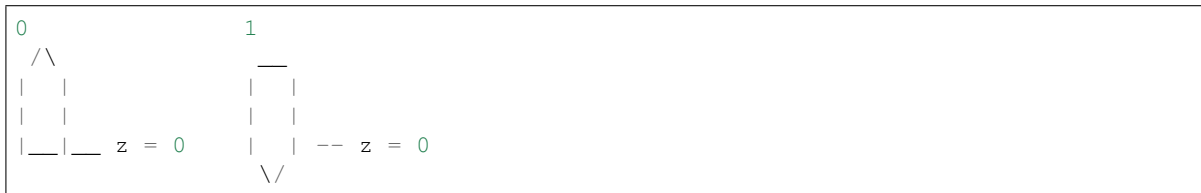
- **fc\_axis** (*FreeCAD.Vector*) – axis on the direction of the led
- **fc\_axis\_cable** (*FreeCAD.Vector*) – axis on the direction of the cable
- **pos** (*FreeCAD.Vector*) – Placement of the object
- **name** (*str*) – Object name

### 1.4.6 Functions details

#### fcfun

**class** fcfun.**NutHole** (*nut\_r, nut\_h, hole\_h, name, extra=1, nuthole\_x=1, cx=0, cy=0, holedown=0*)

Adding a Nut hole (hexagonal) with a prism attached to introduce the nut. Tolerances are included



#### Parameters

- **nut\_r** (*float*) – Circumradius of the hexagon
- **nut\_h** (*float*) – Height of the nut, usually larger than the actual nut height, to be able to introduce it
- **hole\_h** (*float*) – The hole height, from the center of the hexagon to the side it will see light

- **name** (*str*) – Name of the object (string)
- **extra** (*int*) –
  - 1 if you want 1 mm out of the hole, to cut
- **nuthole\_x** (*int*) –
  - 1 : if you want that the nut height to be along the X axis and the 2\*apotheme on the Y axis ie. Nut hole facing X
  - 0 : if you want that the nut height to be along the Y axis ie. Nut hole facing Y
- **cx** (*int*) –
  - 1 : if you want the coordinates referenced to the x center of the piece it can be done because it is a new shape formed from the union
- **cy** (*int*) –
  - 1 : if you want the coordinates referenced to the y center of the piece
- **holedown** (*int*) –

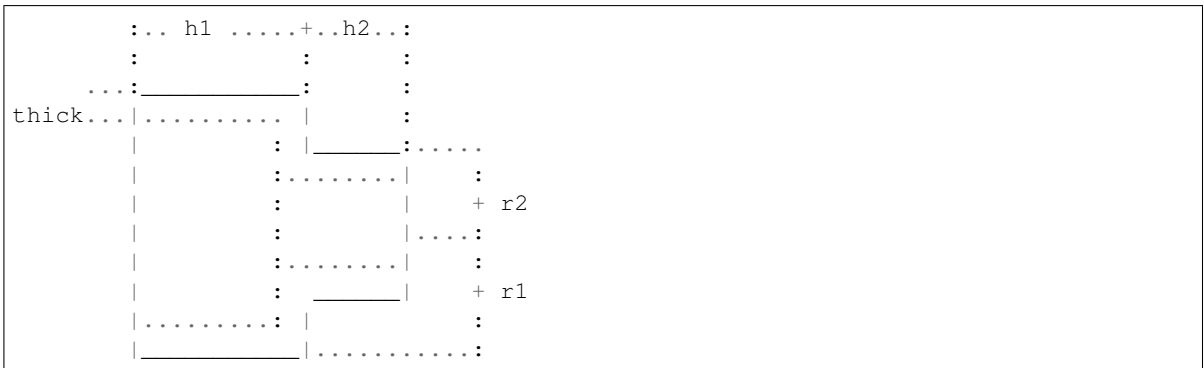
**I THINK IS THE OTHER WAY; CHECK**

- 0: the z0 is the bottom of the square (hole)
- 1: the z0 is the center of the hexagon (nut) it can be done because it is a new shape formed from the union

**Returns** FreeCAD object of a nut hole

**Return type** FreeCAD Object

`fcfun.add2CylsHole (r1, h1, r2, h2, thick, normal=FreeCAD.Vector, pos=FreeCAD.Vector)`  
 Creates a piece formed by 2 hollow cylinders



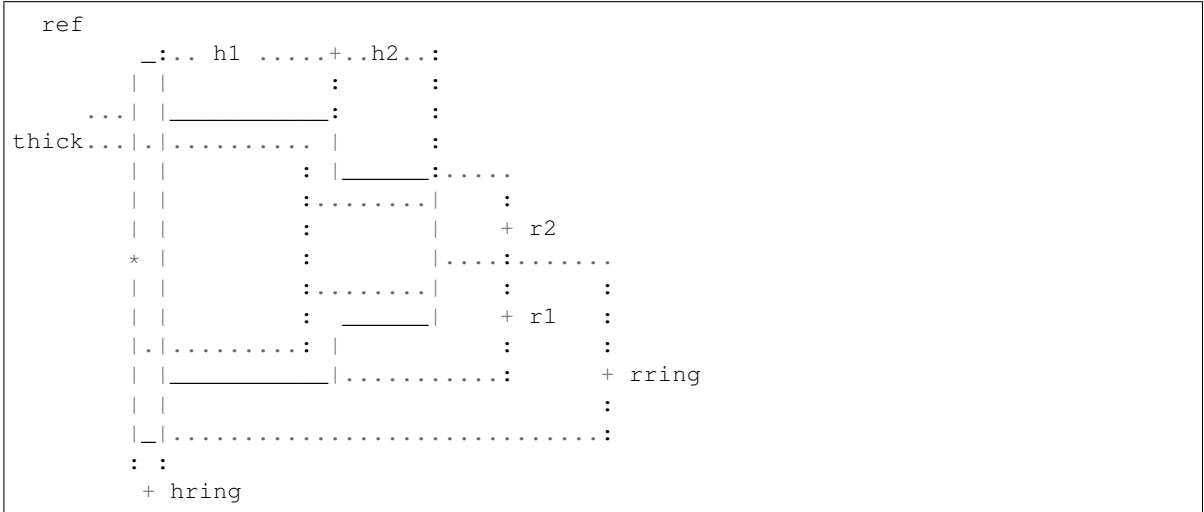
**Parameters**

- **r1** (*float*) – Radius of the 1st cylinder. The first cylinder relative to the position pos
- **h1** (*float*) – Height of the 1st cylinder (seen from outside)
- **r2** (*float*) – Radius of the 2nd cylinder
- **h2** (*float*) – Height of the 2nd cylinder (seen from outside)
- **normal** (*FreeCAD.Vector*) – Direction of the height
- **pos** (*FreeCAD.Vector*) – Position of the center

**Returns** FreeCAD Shape of a two cylinders

### Return type Shape

`fcfun.add3CylsHole(r1, h1, r2, h2, rring, hring, thick, normal=FreeCAD.Vector, pos=FreeCAD.Vector)`  
Creates a piece formed by 2 hollow cylinders, and a ring on the side of the larger cylinder



### Parameters

- **r1** (*float*) – Radius of the 1st cylinder. The first cylinder relative to the position pos (if this is larger than r2, the ring will go first)
- **h1** (*float*) – Height of the 1st cylinder (seen from outside)
- **r2** (*float*) – Radius of the 2nd cylinder
- **h2** (*float*) – Height of the 2nd cylinder (seen from outside)
- **rring** (*float*) – Radius of the ring, it has to be larger than r1, and r2
- **hring** (*float*) – Height of the ring, it has to be larger than r1, and r2
- **thick** (*float*) – Thickness of the walls, excluding the ring
- **normal** (*FreeCAD.Vector*) – Direction of the height
- **pos** (*FreeCAD.Vector*) – Position of the center

**Returns** FreeCAD Shape of a three cylinders

### Return type Shape

`fcfun.addBolt(r_shank, l_bolt, r_head, l_head, hex_head=0, extra=1, support=1, headdown=1, name='bolt')`  
Creates the hole for the bolt shank and the head or the nut Tolerances have to be included

### Parameters

- **r\_shank** (*float*) – Radius of the shank (tolerance included)
- **l\_bolt** (*float*) – Total length of the bolt: head & shank
- **r\_head** (*float*) – Radius of the head (tolerance included)
- **l\_head** (*float*) – Length of the head
- **hex\_head** (*int*) –



**Indicates if the head is hexagonal or rounded**

- 1: hexagonal
- 0: rounded
- **h\_layer3d** (*float*) – Height of the layer for printing, if 0, means that the support is not needed
- **extra** (*int*) – 1 if you want 1 mm on top and bottom to avoid cutting on the same plane pieces after making cuts (boolean difference)
- **support** (*int*) – 1 if you want to include a triangle between the shank and the head to support the shank and not building the head on the air using kcomp.LAYER3D\_H
- **headdown** (*int*) –
  - 1 if the head is down.
  - 0 if it is up

**Returns** FreeCAD Object of a bolt

**Return type** FreeCAD Object

`fcfun.addBoltNut_hole(r_shank, l_bolt, r_head, l_head, r_nut, l_nut, hex_head=0, extra=1, supp_head=1, supp_nut=1, headdown=1, name='bolt')`

Creates the hole for the bolt shank, the head and the nut. The bolt head will be at the bottom, and the nut will be on top. Tolerances have to be already included in the arguments values

**Parameters**

- **r\_shank** (*float*) – Radius of the shank (tolerance included)
- **l\_bolt** (*float*) – Total length of the bolt: head & shank
- **r\_head** (*float*) – Radius of the head (tolerance included)
- **l\_head** (*float*) – Length of the head
- **r\_nut** (*float*) – Radius of the nut (tolerance included)
- **l\_nut** (*float*) – Length of the nut. It doesn't have to be the length of the nut but how long you want the nut to be inserted
- **hex\_head** (*int*) –

**Indicates if the head is hexagonal or rounded**

- 1: hexagonal
- 0: rounded
- **zpos\_nut** (*float*) – Indicates the height position of the nut, the lower part
- **h\_layer3d** (*float*) – Height of the layer for printing, if 0, means that the support is not needed
- **extra** (*int*) – 1 if you want 1 mm on top and bottom to avoid cutting on the same plane pieces after making differences
- **support** (*int*) – 1 if you want to include a triangle between the shank and the head to support the shank and not building the head on the air using kcomp.LAYER3D\_H

**Returns** FreeCAD Object of a Nut Hole

**Return type** FreeCAD Object

`fcfun.addBox(x, y, z, name, cx=False, cy=False)`

Adds a box, centered on the specified axis x and/or y, with its Placement and Rotation at zero. So it can be referenced absolutely from its given position

**Parameters**

- **x** (*float*) – Length
- **y** (*float*) – Width
- **z** (*float*) – Height
- **name** (*str*) – Object Name
- **cx** (*Boolean*) – Centered in axis x
- **cy** (*Boolean*) – Centered in axis y

**Returns** FreeCAD.Object with the shape of a box

**Return type** FreeCAD.Object

`fcfun.addBox_cen(x, y, z, name, cx=False, cy=False, cz=False)`

Adds a box, centered on the specified axis, with its Placement and Rotation at zero. So it can be referenced absolutely from its given position

**Parameters**

- **x** (*float*) – Length
- **y** (*float*) – Width
- **z** (*float*) – Height
- **name** (*str*) – Object Name
- **cx** (*Boolean*) – Centered in the X axis
- **cy** (*Boolean*) – Centered in the Y axis
- **cz** (*Boolean*) – Centered in the Z axis

**Returns** FreeCAD.Object with the shape of a box

**Return type** FreeCAD.Object

`fcfun.addCyl(r, h, name)`

Add cylinder

**Parameters**

- **r** (*float*) – Radius
- **h** (*float*) – Height

**Returns** Cylinder

**Return type** FreeCAD Object

`fcfun.addCylHole(r_ext, r_int, h, name, axis='z', h_disp=0)`

Add cylinder, with inner hole:

**Parameters**

- **r\_ext** (*float*) – External radius,
- **r\_int** (*float*) – Internal radius,
- **h** (*float*) – Height

- **name** (*str*) – Object name
- **axis** (*str*) –  
   ‘x’, ‘y’ or ‘z’
  - ‘x’ will along the x axis
  - ‘y’ will along the y axis
  - ‘z’ will be vertical
- **h\_disp** (*int*) –  
   **Displacement on the height.**
  - if 0, the base of the cylinder will be on the plane
  - if -h/2: the plane will be cutting h/2

**Returns** Cylinder with hole

**Return type** FreeCAD.Object

`fcfun.addCylHolePos (r_out, r_in, h, name, normal=FreeCAD.Vector, pos=FreeCAD.Vector)`

Same as addCylHole, but avoiding the creation of many FreeCAD objects

Add cylinder, with inner hole

#### Parameters

- **r\_out** (*float*) – Outside radius
- **r\_in** (*float*) – Inside radius
- **h** (*float*) – Height
- **name** (*str*) – Object name
- **normal** (*FreeCAD.Vector*) – FreeCAD.Vector pointing to the normal (if its module is not one, the height will be larger than h)
- **pos** (*FreeCAD.Vector*) – Position of the cylinder

**Returns** FreeCAD Shape of a cylinder with hole

**Return type** Shape

`fcfun.addCylPos (r, h, name, normal=FreeCAD.Vector, pos=FreeCAD.Vector)`

Same as addCyl\_pos, but avoiding the creation of many FreeCAD objects

#### Parameters

- **r** (*float*) – Radius,
- **h** (*float*) – Height
- **name** (*str*) – Objet name
- **normal** (*FreeCAD.Vector*) – FreeCAD.Vector pointing to the normal (if its module is not one, the height will be larger than h)
- **pos** (*FreeCAD.Vector*) – Position of the cylinder

**Returns** Cylinder

**Return type** FreeCAD Object

`fcfun.addCyl_pos (r, h, name, axis='z', h_disp=0)`

Add cylinder in a position. So it is in a certain position, with its Placement and Rotation at zero. So it can be referenced absolutely from its given position

#### Parameters

- **r** (*float*) – Radius
- **h** (*float*) – Height
- **name** (*str*) – Name
- **axis** (*str*) –  
  - 'x', 'y' or 'z'
  - 'x' will along the x axis
  - 'y' will along the y axis
  - 'z' will be vertical
- **h\_disp** (*int*) –

#### Displacement on the height.

- if 0, the base of the cylinder will be on the plane
- if -h/2: the plane will be cutting h/2

**Returns** Cylinder

**Return type** FreeCAD Object

`fcfun.add_fcobj (shp, name, doc=None)`

Just creates a freeCAD object of the shape, just to save one line

`fcfun.aluprof_vec (width, thick, slot, insquare)`

Creates a wire (shape), that is an approximation of a generic alum profile extrusion



### Parameters

- **width** (*float*) – The total width of the profile, it is a square
- **thick** (*float*) – The thickness of the side
- **slot** (*float*) – The width of the rail
- **insquare** (*float*) – The width of the inner square
- **indiam** (*float*) – The diameter of the inner hole

**Returns** The points of the aluminum profile positive quadrant

**Return type** Vector

`fcfun.calc_desp_ncen`(*Length, Width, Height, vec1, vec2, cx=False, cy=False, cz=False, H\_extr=False*)

Similar to `calc_rot`, but calculates the displacement, when we don't want to have all of the dimensions centered. First vector original direction (x,y,z) is (1,0,0) Second vector original direction (x,y,z) is (0,0,-1). The arguments `vec1`, `vec2` are tuples (x,y,z) but they may be also FreeCAD.Vectors

```

      Z          . Y          length (x) = 1
:  _          .          width  (y) = 2
: /  /|          heigh  (z) = 3
:/_ /|
| | |          vec1 original (before rotation) = VX
| | /          vec2 original (before rotation) = -VZ
|_|/.....X
    
```

Example after rotation **and** change position

```

Z          . Y          length (x) = 3
:  _          .          width  (y) = 2
: /  /|          heigh  (z) = 1
:/_ //          vec1 = VZ
|_|/.....X          vec2 = VX
    
```

So we have to move X its original height (3), otherwise it would be on the negative side, like this

```

      Z
      :          . Y          length (x) = 3
      :  _          .          width  (y) = 2
      /  : /|          heigh  (z) = 1
      /_ : //          vec1 = VZ
      |_|/.....X          vec2 = VX
    
```

the picture **is** wrong, because originally it **is** centered, that's why the position **is** moved only half of the dimension. But the concept **is** valid

### Parameters

- **vec1** (*tuples*) – Have to be on the axis: x, -x, y, -y, z, -z

`vec1` can be (0,0,0): it means that it doesn't matter how it **is** rotated

- **vec2** (*tuples*) – Have to be on the axis: x, -x, y, -y, z, -z

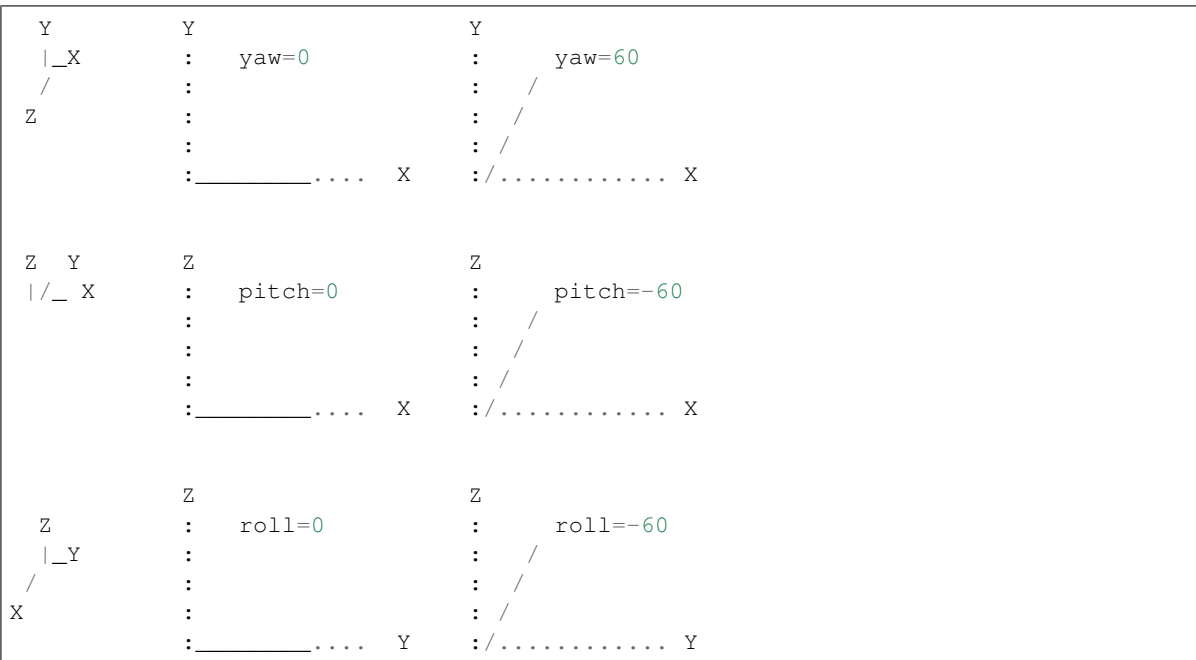
- **Length** (*float*) – Original dimension on X
- **Width** (*float*) – Original dimension on Y
- **Height** (*float*) – Original dimension on Z
- **cx** (*boolean*) – Position centered or not
- **cy** (*boolean*) – Position centered or not
- **cz** (*boolean*) – Position centered or not

**Returns** Vector of the displacement

**Return type** FreeCAD.Vector

`fcfun.calc_rot (vec1, vec2)`

Having an object with an orientation defined by 2 vectors the vectors a tuples, nor FreeCAD.Vectors use the wrapper `fc_calc_rot` to have FreeCAD.Vector arguments First vector original direction (x,y,z) is (1,0,0) Second vector original direction (x,y,z) is (0,0,-1) we want to rotate the object in an ortoghonal direction. The vectors will be in -90, 180, or 90 degrees. this function returns the Rotation given by yaw, pitch and roll In case `vec1` is (0,0,0), means that it doesn't matter that vector. Yaw is the rotation of Z axis. Positive Yaw is like screwing up



### Parameters

- **vec1** (*tuples*) – Direction
- **vec2** (*tuples*) – Direction

### Returns

**Return type** FreeCAD.Rotation

`fcfun.calc_rot_z (v_refz, v_refx)`

Calculates de rotation like `calc_rot`. However uses a different origin axis. `calc_rot` uses: `vec1` original direction (x,y,z) is (0,0,1) `vec2` original direction (x,y,z) is (1,0,0) So it makes a change of axis before calling `calc_rot`

### Parameters

- **v\_refz** (*tuple or FreeCAD.Vector*) – Vector indicating the rotation from (0,0,1) to v\_refz
- **v\_refx** (*tuple or FreeCAD.Vector*) – Vector indicating the rotation from (1,0,0) to v\_refx

#### Returns

**Return type** FreeCAD.Rotation

**fcfun.edgeonaxis** (*edge, axis*)  
It tells if an edge is on an axis

#### Parameters

- **edge** (*Edge*) – A FreeCAD edge, with its vertexes
- **axis** (*str*) – 'x', '-x', 'y', '-y', 'z', '-z'

**Returns** True: edge on an axis False: edge not on an axis

**Return type** boolean

**fcfun.equ** (*x, y*)  
Compare numbers that are the same but not exactly the same

**fcfun.fc\_calc\_desp\_ncen** (*Length, Width, Height, fc\_vec1, fc\_vec2, cx=False, cy=False, cz=False, H\_extr=False*)  
Same as calc\_desp\_ncen but using FreeCAD.Vectors arguments

**fcfun.fc\_calc\_rot** (*fc\_vec1, fc\_vec2*)  
Same as calc\_rot but using FreeCAD.Vectors arguments

**fcfun.fc\_isonbase** (*fcv*)  
Just tells if a vector has 2 of the coordinates zero so it is on just a base vector

**fcfun.fc\_isparal** (*fc1, fc2*)  
Return 1 if fc1 and fc2 are paralell (colinear), 0 if they are not

#### Parameters

- **fc1** (*FreeCAD.Vector*) – Firs vector
- **fc2** (*FreeCAD.Vector*) – Second vector

#### Returns

- \* 1 if fc1 and fc2 are parallel
- \* 0 if they are not

**fcfun.fc\_isparal\_nrm** (*fc1, fc2*)  
Very similar to fc\_isparal, but in this case the arguments are normalized so, less operations to do. return 1 if fc1 and fc2 are paralell (colinear), 0 if they are not

#### Parameters

- **fc1** (*FreeCAD.Vector*) – Firs vector
- **fc2** (*FreeCAD.Vector*) – Second vector

#### Returns

- \* 1 if fc1 and fc2 are parallel
- \* 0 if they are not

`fcfun.fc_isperp (fc1,fc2)`

Return 1 if fc1 and fc2 are perpendicular, 0 if they are not

**Parameters**

- **fc1** (*FreeCAD.Vector*) – First vector
- **fc2** (*FreeCAD.Vector*) – Second vector

**Returns**

- \* 1 if fc1 and fc2 are perpendicular
- \* 0 if they are not

`fcfun.fillet_len (box, e_len, radius, name)`

Make a new object with fillet

**Parameters**

- **box** (*TopoShape*) – Original shape we want to fillet
- **e\_len** (*float*) – Length of the edges that we want to fillet
- **radius** (*float*) – Radius of the fillet
- **name** (*str*) – Name of the shape we want to create

**Returns** FreeCAD Object with fillet made

**Return type** FreeCAD Object

`fcfun.filletchamfer (fco, e_len, name, fillet=1, radius=1, axis='x', xpos_chk=0, ypos_chk=0, zpos_chk=0, xpos=0, ypos=0, zpos=0)`

Fillet or chamfer edges of a certain length, on a certain axis and a certain coordinate

**Parameters**

- **fco** (*FreeCAD Object*) – Original FreeCAD object we want to fillet or chamfer
- **fillet** (*int*) –
  - 1 if we are doing a fillet
  - 0 if it is a chamfer
- **e\_len** (*float*) – Length of the edges that we want to fillet or chamfer if e\_len == 0, chamfer/fillet any length
- **radius** (*float*) – Radius of the fillet or chamfer
- **axis** (*str*) – Axis where the fillet will be
- **xpos\_chk** (*int*) – If the X position will be checked
- **ypos\_chk** (*int*) – If the Y position will be checked
- **zpos\_chk** (*int*) – If the Z position will be checked
- **xpos** (*float*) – The X position
- **ypos** (*float*) – The Y position
- **zpos** (*float*) – The Z position
- **name** (*str*) – Name of the fco we want to create



## Notes

If axis = 'x', x\_pos\_check will not make sense

**Returns** FreeCAD Object with fillet/chamfer made

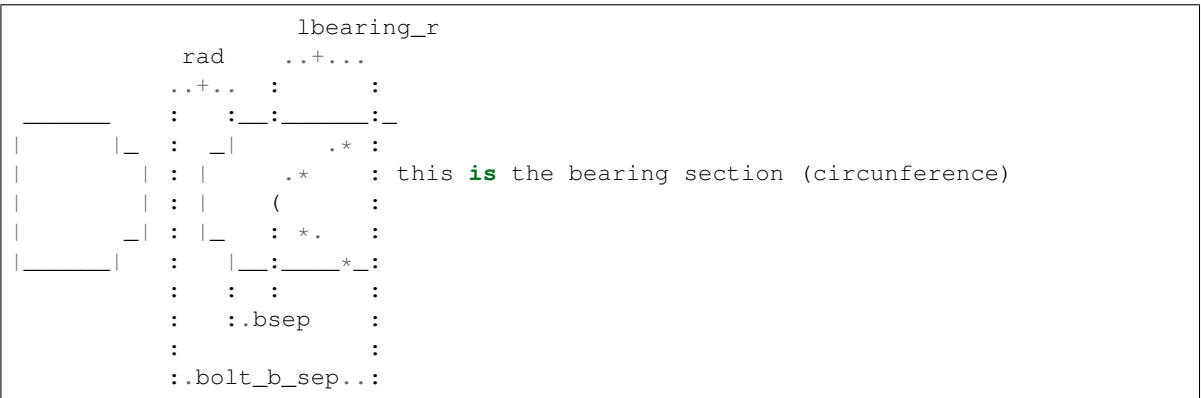
**Return type** FreeCAD Object

`fcfun.fuseshplist (shp_list)`

Since multifuse methods needs to be done by a shape and a list, and usually I have a list that I want to fuse, I make this function to save the inconvenience of doing everytime what I will do here Fuse multiFuse

`fcfun.get_bolt_bearing_sep (bolt_d, hasnut, lbearing_r, bsep=0)`

same as get\_bolt\_end\_sep, but when there is a bearing. If there is a bearing, there will be more space because the nut is at the bottom or top, and the widest side is on the middle



## Parameters

- **bolt\_d** (*int*) – Diameter of the bolt: 3, 4, ... for M3, M4,...
- **hasnut** (*int*) –
  - 1: if there is a nut
  - 0: if there is not a nut, so just the bolt head (smaller)
- **lbearing\_r** (*float*) – Radius of the linear bearing
- **bsep** (*float*) – Separation from the outside of the nut to the end of bearing default value 0mm

**Returns** Minimum separation between the center of the bolt and the bearing

**Return type** float

`fcfun.get_bolt_end_sep (bolt_d, hasnut, sep=2.0)`

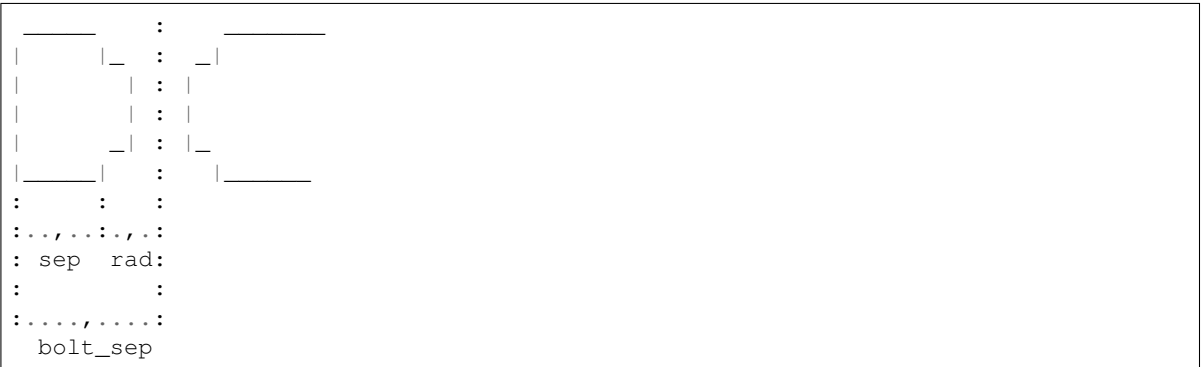
Calculate Bolt separation

Calculates know how much separation is needed for a bolt The bolt (din912) head diameter is usually smaller than the nut (din934) The nut max value is given by its 2\*apotheme (S) (wrench size) so its max diameter is 2A x cos(30)

Example of nut and bolt head sizes:

|           | din912 | din938 |        |
|-----------|--------|--------|--------|
|           | D      | S(max) | D(max) |
| <b>M3</b> | 5.5    | 5.5    | 6,35   |
| <b>M4</b> | 7.0    | 7.0    | 8,08   |
| <b>M5</b> | 8.5    | 8.0    | 9,24   |
| <b>M6</b> | 10.0   | 10.0   | 11,55  |

Therefore, if there is a nut, the nut will be used to calculate the separation



#### Parameters

- **bolt\_d**(*int*) – Diameter of the bolt: 3, 4, ... for M3, M4,...
- **hasnut**(*int*) –
  - 1: if there is a nut
  - 0: if there is not a nut, so just the bolt head (smaller)
- **sep**(*float*) –
  - Separation from the outside of the nut to the end, if empty, default value 2mm

**Returns** Minimum separation between the center of the bolt and the end

**Return type** float

`fcfun.get_fc_perpend1(fcv)`

gets a 'random' perpendicular FreeCAD.Vector

**Parameters** **fcv** (*FreeCAD.Vector*) – Vector from which to get perpendicular vector

**Returns** Random perpendicular vector

**Return type** FreeCAD.Vector

`fcfun.get_fclist_4perp2_fcvec(fcvec)`

Gets a list of 4 FreeCAD.Vector perpendicular to one base vector fcvec can only be: \* (1,0,0) \* (0,1,0) \* (0,0,1) \* (-1,0,0) \* (0,-1,0) \* (0,0,-1)

**For example:**

```
from (1, 0, 0) -> (0, 1, 0), (0, 0, 1), (0, -1, 0), (0, 0, -1)
```

**Parameters** **fcvec** (*vector*) – (1,0,0), (0,1,0), (0,0,1), (-1,0,0), (0,-1,0), (0,0,-1)

**Returns** List of FreeCAD.Vector

**Return type** list

`fcfun.get_fclist_4perp2_vecname (vecname)`

Gets a list of 4 FreeCAD.Vector perpendicular to one vecname different from get\_fclist\_4perp\_vecname For example:

```
from 'x' -> (0,1,1), (0,-1,1), (0,-1,-1), (0,1,-1)
```

**Parameters** `vecname` (*str*) – 'x', '-x', 'y', '-y', 'z', '-z'

**Returns** List of FreeCAD.Vector

**Return type** list

`fcfun.get_fclist_4perp_fcvec (fcvec)`

Gets a list of 4 FreeCAD.Vector perpendicular to one base vector fcvec can only be: \* (1,0,0) \* (0,1,0) \* (0,0,1) \* (-1,0,0) \* (0,-1,0) \* (0,0,-1)

**For example:**

```
from (1,0,0) -> (0,1,0), (0,0,1), (0,-1,0), (0,0,-1)
```

**Parameters** `fcvec` (*vector*) – (1,0,0), (0,1,0), (0,0,1), (-1,0,0), (0,-1,0), (0,0,-1)

**Returns** List of FreeCAD.Vector

**Return type** list

`fcfun.get_fclist_4perp_vecname (vecname)`

Gets a list of 4 FreeCAD.Vector perpendicular to one vecname for example:

```
from 'x' -> (0,1,0), (0,0,1), (0,-1,0), (0,0,-1)
```

**Parameters** `vecname` (*str*) – 'x', '-x', 'y', '-y', 'z', '-z'

**Returns** List of FreeCAD.Vector

**Return type** list

`fcfun.get_fcvector (tup)`

Gets the FreeCAD.Vector of a tuple

**Parameters** `tup` (*tuple*) – Tuple of 3 elements

**Returns** FreeCAD.Vector of a tuple

**Return type** FreeCAD.Vector

`fcfun.get_nameofbasevec (fcvec)`

From a base vector either: (1,0,0), (0,1,0), (0,0,1), (-1,0,0), (0,-1,0), (0,0,-1) Gets its name: 'x', 'y',....

**Returns** Vector name

**Return type** str

`fcfun.get_positive_vecname (vecname)`

It just get 'x' when vecname is 'x' or '-x', and the same for the others, because some functions receive only positive base vector

**Parameters** `vecname` (*str*) – 'x', '-x', 'y', '-y', 'z', '-z'

**Returns** Vector name

**Return type** str

`fcfun.get_rot(v1, v2)`

Calculate the rotation from v1 to v2 the difference with previous versions, such `fc_calc_rot`, `calc_rot`, `calc_rot` is that it is for any vector direction. The difference with `DraftVecUtils.getRotation` is that `getRotation` doesn't work for vectors with 180 degrees.

## Notes

MAYBE IT IS NOT NECESSARY, just use `FreeCAD.Rotation rotation.Axis, math.degrees(rotation.Angle)`

**Parameters**

- **v1** (`FreeCAD.Vector`) – Vector to calculate the rotation
- **v2** (`FreeCAD.Vector`) – Vector to calculate the rotation

**Returns** Tuple representing a quaternion rotation between v2 and v1

**Return type** `FreeCAD.Rotation`

`fcfun.get_tangent_2circles(center1_pt, center2_pt, rad1, rad2, axis_n, axis_side=None)`

Returns a list of lists (matrix) with the 2 tangent points for each of the 2 tangent lines

```
(difficult to draw in using ASCII text)

          axis_p
          :
      T2 :          axis_side
          . * r1      / -----
          .           /
          .           /
T1 .           . r2-r1 (r_diff) + r2*sin(beta)
    *           .           :
r1 . alpha      beta       :
    *-----* ----- axis_c (axis going thru centers)
C1           : C2
::           :
::           :
::           :
::           +       :
::           r2*cos(beta):
::           :
::           :
::           :
::           +
::           C1_C2_d (hypotenuse)
::           :
::           :
r1*cos(beta)

alpha = atan(r_diff/C1_C2_d)
beta = 90 - alpha

tangent points along axis_c and axis_p

T2_c = C2_c - r2 * cos(beta)
T2_p = C2_p - r2 * sin(beta)
```

(continues on next page)

(continued from previous page)

```
Tl_c = Cl_c - r1 * cos(beta)
Tl_p = Cl_p - r1 * sin(beta)
```

### Parameters

- **center1\_pt** (*FreeCAD.Vector*) – Center of the circle 1
- **center2\_pt** (*FreeCAD.Vector*) – Center of the circle 2
- **rad1** (*float*) – Radius of the circle 1
- **rad2** (*float*) – Radius of the circle 2
- **axis\_n** (*FreeCAD.Vector*) – Direction of the normal of the circle
- **axis\_side** (*FreeCAD.Vector*) – Direction to the side of the tangent line, if not given, it will return the 2 points of both lines The 2 tangent lines will be at each side of axis\_c. The smaller than 90 degree angle between axis\_side and the 2 possible axis\_p

### Returns

- \* *If axis\_side is given –*
  - Returns a list of lists (matrix)
    - \* Element [0][0] is the point tangent to circle 1 at side axis\_side
    - \* Element [0][1] is the point tangent to circle 2 at side axis\_side
    - \* Element [1][0] is the point tangent to circle 1 at opposite side of direction of axis\_side
    - \* Element [1][1] is the point tangent to circle 2 at opposite side of direction of axis\_side
- \* *If axis\_side is not given, the order of the list of the lines is – arbitrary*
- \* *If there is an error it will return 0*

### Notes

#### Interesting variables

axis\_p (*FreeCAD.Vector*)

Vector of the circle plane, perpendicular to axis\_d. It can have to possible directions. If parameter axis\_side is defined, it will have the direction that has less than 90 degrees related to axis\_side

`fcfun.get_tangent_circle_pt` (*ext\_pt, center\_pt, rad, axis\_n, axis\_side=None*)

Get the point of the tangent to the circle

(difficult to draw **in** using ASCII text)

```
external point
:      tangent point 1
*---
 \ /  \
  \ (  ) circle
tangent \ _ /
point 2
```

The 3 points: center(C), ext\_pt(E) **and** tangent\_pt(T) form a

(continues on next page)

(continued from previous page)

```

rectangle triangle

        axis_p
        :      axis_side
        :      /
        : <90 /
        T      /
        . *. /
        . 90 . rad

        . alpha      beta .
*-----*      ---- axis_c (axis going thru centers)
E          :      C
:          :      :
:.....:      :
:      +      :
:  axis_c_ET_d      :
:          :      :
:.....:      :
:      +
      EC_d (hypotenuse)

```

### Parameters

- **ext\_pt** (*FreeCAD.Vector*) – External point
- **center\_pt** (*FreeCAD.Vector*) – Center of the circle
- **rad** (*float*) – Radius of the circle
- **axis\_n** (*FreeCAD.Vector*) – Direction of the normal of the circle
- **axis\_side** (*FreeCAD.Vector*) – Direction to the side of the tangent point, if not given, it will return both points The 2 tangent points will be at each side of axis\_c. The smaller than 90 degree angle between axis\_side and the 2 possible axis\_p

### Returns

- *If axis\_side is not given* – returns a list with the 2 points that each point forms a line tangent to the circle. The 2 lines are defined by one of each point and the external point.
- *If axis\_side is given* – Only returns a point (*FreeCAD.Vector*) with the tangent point defined by the direction of axis\_side
- *If there is an error it will return 0*

### Notes

#### Interesting Parameters

axis\_p (*FreeCAD.Vector*)

Vector of the circle plane, perpendicular to axis\_d. It can have to possible directions. If parameter axis\_side is defined, it will have the direction that has less than 90 degrees related to axis\_side

fcfun.get\_vecname\_perpend1 (vecname)

Gets a perpendicular vecname

**Parameters** **vec** (*str*) – 'x', '-x', 'y', '-y', 'z', '-z'

**Returns** Perpendicular vector name

**Return type** str

`fcfun.get_vecname_perpend2 (vecname)`

Gets the other perpendicular vecname (see `get_vecname_perpend`)

**Parameters** **vec** (*str*) – 'x', '-x', 'y', '-y', 'z', '-z'

**Returns** Perpendicular vector name

**Return type** str

`fcfun.get_fcvecfname (axis)`

Returns the FreeCAD.Vector of the vector name given

`fcfun.getvecfname (axis)`

Get axis name renurns the vector

`fcfun.regpolygon_dir_vec1 (n_sides, radius, fc_normal, fc_verx1, pos)`

Similar to `regpolygon_vec1` but in any place and direction of the space calculates the vertexes of a regular polygon. Returns a list of FreeCAD vectors with the vertexes. The first vertex will be repeated at the end, this is needed to close the wire to make the shape The polygon will have the center in pos. The normal on `fc_normal` The direction of the first vertex on `fc_verx_1`

**Parameters**

- **n\_sides** (*int*) – Number of sides of the polygon
- **radius** (*float*) – Circumradius of the polygon
- **fc\_normal** (*FreeCAD.Vector*) – Direction of the normal
- **fc\_verx1** (*FreeCAD.Vector*) – Direction of the first vertex
- **pos** (*FreeCAD.Vector*) – Position of the center

**Returns** List of FreeCAD.Vector of the vertexes

**Return type** List

`fcfun.regpolygon_vec1 (n_sides, radius, x_angle=0)`

Calculates the vertexes of a regular polygon. Returns a list of FreeCAD vectors with the vertexes. The first vertex will be repeated at the end, this is needed to close the wire to make the shape The polygon will be on axis XY (z=0).

**Parameters**

- **n\_sides** (*int*) – Number of sides of the polygon
- **radius** (*float*) – Circumradius of the polygon
- **x\_angle** (*float*) – If zero, the first vertex will be on axis x (y=0) if `x_angle != 0`, it will rotated some angle

**Returns** List of FreeCAD.Vector of the vertexes

**Return type** List

`fcfun.rotateview (axisX=1.0, axisY=0.0, axisZ=0.0, angle=45.0)`

Rotate the camara

`fcfun.shpRndRectWire (x=1, y=1, r=0.5, zpos=0)`

Creates a wire (shape), that is a rectangle with rounded edges. if `r== 0`, it will be a rectangle The wire will be centered

## Parameters

- **x** (*float*) – Dimension of the base, on the X axis
- **y** (*float*) – Dimension of the height, on the Y axis
- **r** (*float*) – Radius of the rounded edge.
- **zpos** (*float*) – Position on the Z axis

**Returns** FreeCAD Wire of a rounded edges rectangle

**Return type** Shape Wire

```
fcfun.shp_2stadium_dir (length,    r_s,    r_l,    h_tot,    h_rl,    fc_axis_h=FreeCAD.Vector,
                        fc_axis_l=FreeCAD.Vector,    ref_l=1,    rl_h0=1,    xtr_h=0,    xtr_nh=0,
                        pos=FreeCAD.Vector)
```

Makes to concentric stadiums, useful for making rails for bolts the length is the same for both. Changes the radius and the height The smaller radius will have the largest length

```

rl_h0 = 1: the large stadium is at h=0

      fc_axis_h
      _____:_____.....
      |      :      :      |      :
      |      :      :      |      :
      ____|      :      :      |_____ + h_tot
      |      :      :      |      : h_rl :
      |_____ :_____ * _____| .....> fc_axis_l
:      :      :      :
:.....+:.....:
:      r_s:      lenght
:      :
:      :
:..r_l...:

rl_h0 = 0 : the large stadium is at the end of h

      fc_axis_h
      _____:_____.....
      |      :      :      |      : h_rl :
      |_____ :      :      |_____ :
      |      :      :      |      :
      |      :      :      |      : + h_tot
      :      :      :      :
      :      |_____ :_____ * _____| .....> fc_axis_l
      :      :      :      :

```

(continues on next page)



(continued from previous page)

```

:..:..+:.....:
:   r_s:   length
:       :
:       :
:       :
:..r_l...:

```

on the axis\_h, the h\_rl stadium can be at the reference, or at the end of the reference (rl\_h0 =0):

```

ref_l points:

fc_axis_s
:
:
:-----
( 2   1 ) -----> fc_axis_l

```

### Parameters

- **length** (*float*) – Length of the parallels, from one semicircle center to the other
- **r\_s** (*float*) – Smaller radius of the semicircles
- **r\_l** (*float*) – Larger radius of the semicircles
- **h\_tot** (*float*) – Total height
- **h\_rl** (*float*) – Height of the larger radius stadium
- **fc\_axis\_h** (*FreeCAD.Vector*) – Vector on the direction of the height
- **fc\_axis\_l** (*FreeCAD.Vector*) – Vector on the direction of the parallels,
- **ref\_l** (*int*) – Reference (zero) of the fc\_axis\_l
  - 1: reference on the center (makes axis\_s symmetrical)
  - 2: reference at one of the semicircle centers (point 2) the other circle center will be on the direction of fc\_axis\_l
- **rl\_h0** (*int*) –
  - 1: if the larger radius stadium is at the beginning of the axis\_h
  - 0: at the end of axis\_h
- **xtr\_h** (*float*) – If >0 it will be that extra height on the direction of fc\_axis\_h
- **xtr\_nh** (*float*) – If >0 it will be that extra height on the opposite direction of fc\_axis\_h
- **xtr\_nh** –
- **pos** (*FreeCAD.Vector*) – Position of the reference

**Returns** FreeCAD Shape of a two stadiums

**Return type** Shape

`fcfun.shp_aluwire_dir` (*width*, *thick*, *slot*, *insquare*, *fc\_axis\_x=FreeCAD.Vector*,  
*fc\_axis\_y=FreeCAD.Vector*, *ref\_x=1*, *ref\_y=1*, *pos=FreeCAD.Vector*)

Creates a wire (shape), that is an approximation of a generic alum profile extrusion. Creates it in any position and any direction

```

          Y
          |
          |_ X
:----- width -----:
:       slot         :
:       :--:         :
:-----: :-----:
|   _  |   _  |
|  \| \   /  /  |
|_ \| \   /  /  | .....
    |   |   | ..... insquare
    |   ( ) | .....indiam :
    _  |   _  | .....:
|  \| /  \  \  |
|  \| /  \  \  | ....
|_____|   |_____| ....thick

          Y values:
:  3   _   4
:  |_1 7| ..... 1,2: width/2 - thick
:  2 / /|_ | .....7: width/2- (thick+thick*cos45)
:___/ / 6 5 ..... 5,6: slot/2.
:  0 |8      :8:insquare/2-thick*cos45  0:insquare/2 :
:.....|.....:.....:.....:

ref_x= 1 ; ref_y = 1
      fc_axis_w
      :
      :
      _ : _
      |_|:|_|
.....|.:|..... fc_axis_p
      _|_:|_
      |_| : |_|
      :
      :
      :

ref_x= 2 ; ref_y = 1 (the zero of axis_y is at the center)
                    (the zero of axis_x is at one side)

      fc_axis_y
      :
      :
      :
      :_ _
      |_|_|_|
.....:|...|..... fc_axis_x
      :_|_|_|
      |_|   |_|
      :
      :
      :

```

### Parameters

- **width** (*float*) – Total width of the profile, it is a square
- **thick** (*float*) – Thickness of the side

- **slot** (*float*) – Width of the rail
- **insquare** (*float*) – Width of the inner square
- **indiam** (*float*) – Diameter of the inner hole
- **fc\_axis\_x** (*int*) – Is a generic X axis, can be any
  - 1: reference (zero) at the center
  - 2: reference (zero) at the side, the other end side will be on the direction of fc\_axis\_x
- **fc\_axis\_y** (*int*) – Is a generic Y axis, can be any perpendicular to fc\_axis\_x
  - 1: reference (zero) at the center
  - 2: reference (zero) at the side, the other end side will be on the direction of fc\_axis\_y
- **ref\_x** (*float*) – Reference (zero) on the fc\_axis\_x
- **ref\_y** (*float*) – Reference (zero) on the fc\_axis\_y
- **pos** (*FreeCAD.Vector*) – Position of the center

**Returns** FreeCAD Shape Wire of a aluminium profile

**Return type** Shape Wire

```
fcfun.shp_belt_dir(center_sep, rad1, rad2, height, fc_axis_h=FreeCAD.Vector,
                  fc_axis_l=FreeCAD.Vector, ref_l=1, ref_h=1, xtr_h=0, xtr_nh=0,
                  pos=FreeCAD.Vector)
```

Makes a shape of 2 tangent circles (like a belt joining 2 circles). check shp\_belt\_wire\_dir

#### Parameters

- **center\_sep** (*float*) – Separation of the circle centers
- **rad1** (*float*) – Radius of the first circle, on the opposite direction of fc\_axis\_l
- **rad2** (*float*) – Radius of the second circle, on the direction of fc\_axis\_l
- **height** (*float*) – Height of the shape
- **fc\_axis\_l** (*FreeCAD.Vector*) – Vector on the direction circle centers, pointing to rad2
- **fc\_axis\_h** (*FreeCAD.Vector*) – Vector on the height direction
- **ref\_l** (*int*) – Reference (zero) of the fc\_axis\_l
  - 1: reference on the center
  - 2: reference at rad1 semicircle centers (point 2) the other circle center will be on the direction of fc\_axis\_l
  - 3: reference at the end of rad1 circle the other end will be on the direction of fc\_axis\_l
- **ref\_h** (*int*) –
  - 1: reference is at the center of the height
  - 2: reference is at the bottom
- **xtr\_h** (*float*) – If >0 it will be that extra height on the direction of fc\_axis\_h
- **xtr\_nh** (*float*) – If >0 it will be that extra height on the opposite direction of fc\_axis\_h
- **pos** (*FreeCAD.Vector*) – Position of the reference

**Returns** FreeCAD Shape of a belt

### Return type Shape

`fcfun.shp_belt_wire_dir`(*center\_sep*, *rad1*, *rad2*, *fc\_axis\_l*=FreeCAD.Vector,  
*fc\_axis\_s*=FreeCAD.Vector, *ref\_l*=1, *ref\_s*=1, *pos*=FreeCAD.Vector)

Makes a shape of a wire with 2 circles and exterior tangent lines check [here](#) It is not easy to draw it well rad1 and rad2 can be exchanged, rad1 doesnt have to be larger:

```

    ....
    :      ( \ tangent
rad1 :      ( \ .. rad2
    :      .--( + +)--
    :      ( / :
    :      ( / :
    :      : :
    :      :...:
    :      + center_sep

    ....
    :      ( \ tangent
rad1 :      ( \ .. rad2
    :      --( + +)--
    :      ( / :
    :      ( / :
    :      : :
    :      :...:
ref_l: 3 2 1

```

### Parameters

- **center\_sep** (*float*) – Separation of the circle centers
- **rad1** (*float*) – Radius of the first circle, on the opposite direction of *fc\_axis\_l*
- **fc\_axis\_l** (*FreeCAD.Vector*) – Vector on the direction circle centers, pointing to rad2
- **fc\_axis\_s** (*FreeCAD.Vector*) – Vector on the direction perpendicular to *fc\_axis\_l*, on the plane of the wire
- **ref\_l** (*int*) – Reference (zero) of the *fc\_axis\_l*
  - 1: reference on the center
  - 2: reference at one of the semicircle centers (point 2) the other circle center will be on the direction of *fc\_axis\_l*
  - 3: reference at the end of rad1 circle the other end will be on the direction of *fc\_axis\_l*
- **pos** (*FreeCAD.Vector*) – Position of the reference

**Returns** FreeCAD Wire of a belt

### Return type Shape Wire

`fcfun.shp_bolt`(*r\_shank*, *l\_bolt*, *r\_head*, *l\_head*, *hex\_head*=0, *xtr\_head*=1, *xtr\_shank*=1, *support*=1,  
*axis*='z', *hex\_ref*='x', *hex\_rot\_angle*=0, *pos*=FreeCAD.Vector)

Similar to addBolt, but creates a shape instead of a FreeCAD Object Creates a shape of the bolt shank and head or the nut Tolerances have to be included if you want it for making a hole

It is referenced at the end of the head

### Parameters

- ```
axis = '-z':  
      Z  
      :  
      :  
.....|_____|  
xtr_head=1 .....| :....|..... X  
          |     |  
          |_____|  
          |   |  
          |   |  
          |___|  
  
axis = 'z':  
      Z  
      :  
      :  
      :  
      :  
      :  
      :  
      :  
.....|_____|  
xtr_head=1 .....| :....|..... X  
          |     |  
          |_____|
```

- Return type** Shape

```
fcfun.shp_bolt_dir(r_shank, l_bolt, r_head, l_head, hex_head=0, xtr_head=1, xtr_shank=1,
support=1, fc_normal=FreeCAD.Vector, fc_verx1=FreeCAD.Vector, pos_n=0,
pos=FreeCAD.Vector)
```

Similar to shp\_bolt, but it can be done in any direction Creates a shape, not a of a FreeCAD Object Creates a shape of the bolt shank and head or the nut Tolerances have to be included if you want it for making a hole

It is referenced at the end of the head

## Parameters

- **r\_shank** (*float*) – Radius of the shank (tolerance included)
- **l\_bolt** (*float*) – Total length of the bolt: head & shank
- **r\_head** (*float*) – Radius of the head (tolerance included)
- **l\_head** (*float*) – Length of the head
- **hex\_head** (*int*) – Indicates if the head is hexagonal or rounded
  - 1: hexagonal
  - 0: rounded
- **h\_layer3d** (*float*) – Height of the layer for printing, if 0, means that the support is not needed
- **xtr\_head** (*int*) – 1 if you want 1 mm on the head to avoid cutting on the same plane pieces after making cuts (boolean difference)
- **xtr\_shank** (*int*) – 1 if you want 1 mm at the opposite side of the head to avoid cutting on the same plane pieces after making cuts (boolean difference)
- **support** (*int*) – 1 if you want to include a triangle between the shank and the head to support the shank and not building the head on the air using kcomp.LAYER3D\_H
- **fc\_normal** (*FreeCAD.Vector*) – Defines the orientation. For example:

```

fc_normal = (0,0,-1):      Z
                            :
                            :_____
                .....|_____|_____
..xtr_head=1   .....|_____|..... X   pos_n = 0
:               l_head+:|_____|_____
:               :.....|_____|_____          pos_n = 1
:+ l_bolt              |_____|_____
:                       |_____|_____
:.....pos_n = 2
:.....|_____xtr_shank
:
:
fc_normal

fc_normal = (0,0,1):      Z
                            :
                            :_____
                .....|_____|_____
xtr_head=1     .....|_____|..... X
                .....|_____|_____

```

- **fc\_verx1** (*FreeCAD.Vector*) – In case of a hexagonal head, this will indicate the axis that the first vertex of the nut will point it has to be perpendicular to fc\_normal,
- **pos\_n** (*int*) – Location of pos along the normal, at the cylinder center
  - 0: at the top of the head (excluding xtr\_head)
  - 1: at the union of the head and the shank
  - 2: at the end of the shank (excluding xtr\_shank)
- **pos** (*FreeCAD.Vector*) – Position of the center of the head of the bolt

**Returns** FreeCAD Shape of a bolt

**Return type** Shape

```
fcfun.shp_boltnut_dir_hole(r_shank, l_bolt, r_head, l_head, r_nut, l_nut, hex_head=0,
                           xtr_head=1, xtr_nut=1, supp_head=1, supp_nut=1, head-
                           start=1, fc_normal=FreeCAD.Vector, fc_verx1=FreeCAD.Vector,
                           pos=FreeCAD.Vector)
```

Similar to addBoltNut\_hole, but in any direction and creates shapes, not FreeCAD Objects Creates the hole for the bolt shank, the head and the nut. The bolt head will be at the botton, and the nut will be on top Tolerances have to be already included in the arguments values

#### Parameters

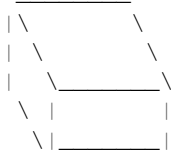
- **r\_shank** (*float*) – Radius of the shank (tolerance included)
- **l\_bolt** (*float*) – Total length of the bolt: head & shank
- **r\_head** (*float*) – Radius of the head (tolerance included)
- **l\_head** (*float*) – Length of the head
- **r\_nut** (*float*) – Radius of the nut (tolerance included)
- **l\_nut** (*float*) – Length of the nut. It doesn't have to be the length of the nut but how long you want the nut to be inserted
- **hex\_head** (*int*) – Indicates if the head is hexagonal or rounded
  - 1: hexagonal
  - 0: rounded
- **xtr\_head** (*int*) – 1 if you want an extra size on the side of the head to avoid cutting on the same plane pieces after making differences
- **xtr\_nut** (*int*) – 1 if you want an extra size on the side of the nut to avoid cutting on the same plane pieces after making differences
- **supp\_head** (*int*) – 1 if you want to include a triangle between the shank and the head to support the shank and not building the head on the air using kcomp.LAYER3D\_H
- **supp\_nut** (*int*) – 1 if you want to include a triangle between the shank and the nut to support the shank and not building the nut on the air using kcomp.LAYER3D\_H
- **headstart** (*int*) – If on pos you have the head, or if you have it on the other end
- **fc\_normal** (*FreeCAD.Vector*) – Direction of the bolt
- **fc\_verx1** (*FreeCAD.Vector*) – Direction of the first vertex of the hexagonal nut. Perpendicular to fc\_normal. If not perpendicular or zero, means that it doesn't matter which direction and the function will obtain one perpendicular direction
- **pos** (*FreeCAD.Vector*) – Position of the head (if headstart) or of the nut

**Returns** FreeCAD Object of a Nut Hole

**Return type** FreeCAD Object

`fcfun.shp_box_dir(box_w, box_d, box_h, fc_axis_w=FreeCAD.Vector, fc_axis_h=FreeCAD.Vector, fc_axis_d=FreeCAD.Vector, cw=1, cd=1, ch=1, pos=FreeCAD.Vector)`

Makes a shape of a box given its 3 dimensions: width, depth and height and the direction of the height and depth dimensions. The position of the box is given and also if the position is given by a corner or its center



Example of **not** centered on origin

```
Z=fc_axis_h      . Y = fc_axis_d
:
:
: /: . / |
: / : . / | h
: /_____ / |
| :.....|...| 3
| / 4 | /
| /      | / d
| /_____ / .....X
1      2
w
```

Example of centered on origin

```
Z=fc_axis_h      Y = fc_axis_d
:
:
: /: : / | .
: / : : / | h
: /___:___ / |
| :.....|...| 3
| / 4 :..|.. / .....X
| /      | / d
| /_____ /
1      2
w
```

### Parameters

- **box\_w** (*float*) – Width of the box
- **box\_d** (*float*) – Depth of the box
- **box\_h** (*float*) – Height of the box
- **fc\_axis\_w** (*FreeCAD.Vector*) – Direction of the width
- **fc\_axis\_d** (*FreeCAD.Vector*) – Direction of the depth
- **fc\_axis\_h** (*FreeCAD.Vector*) – Direction of the height



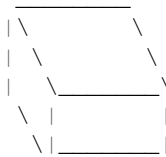
- **cw**(*int*) –
  - 1: the width dimension is centered
  - 0: it is not centered
- **cd**(*int*) –
  - 1: the depth dimension is centered
  - 0: it is not centered
- **ch**(*int*) –
  - 1: the height dimension is centered
  - 0: it is not centered
- **pos**(*FreeCAD.Vector*) – Position of the box, it can be the center one corner, or a point centered in the dimensions given by cw, cd, ch

**Returns** Shape of a box

**Return type** TopoShape

`fcfun.shp_box_dir_xtr`(*box\_w*, *box\_d*, *box\_h*, *fc\_axis\_h*=*FreeCAD.Vector*,  
*fc\_axis\_d*=*FreeCAD.Vector*, *fc\_axis\_w*=*FreeCAD.Vector*, *cw*=1, *cd*=1,  
*ch*=1, *xtr\_h*=0, *xtr\_nh*=0, *xtr\_d*=0, *xtr\_nd*=0, *xtr\_w*=0, *xtr\_nw*=0,  
*pos*=*FreeCAD.Vector*)

Makes a shape of a box given its 3 dimensions: width, depth and height and the direction of the height and depth dimensions. The position of the box is given and also if the position is given by a corner or its center. Extra mm to make cuts



Example of **not** centered on origin

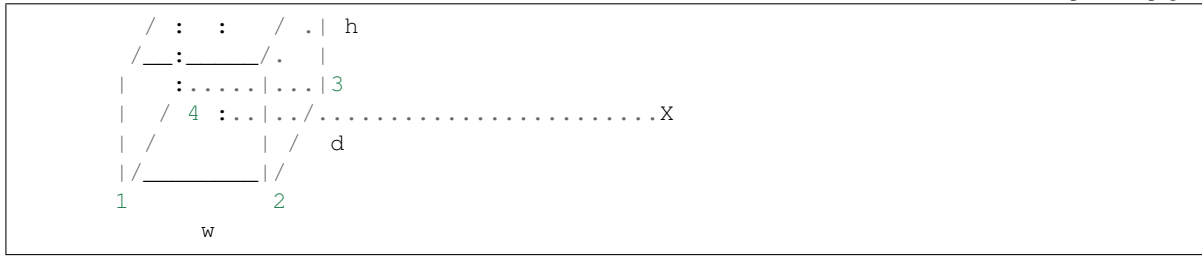
```
Z=fc_axis_h      . Y = fc_axis_d
:                .
:                .
: /: . / |
: / : . / | h
: /_____ / |
| :.....|...|3
| / 4 | /
| / | / d
| /_____ / .....X
1      2
      w
```

Example of centered on origin

```
Z=fc_axis_h      Y = fc_axis_d
:                .
:                .
: /: : / |.
```

(continues on next page)

(continued from previous page)



### Parameters

- **box\_w**(*float*) – Width of the box
- **box\_d**(*float*) – Depth of the box
- **box\_h**(*float*) – Height of the box
- **fc\_axis\_h**(*FreeCAD.Vector*) – Direction of the height
- **fc\_axis\_d**(*FreeCAD.Vector*) – Direction of the depth
- **fc\_axis\_w**(*FreeCAD.Vector*) – Direction of the width
- **cw**(*int*) –
  - 1 the width dimension is centered
  - 0 it is not centered
- **cd**(*int*) –
  - 1 the depth dimension is centered
  - 0 it is not centered
- **ch**(*int*) –
  - 1 the height dimension is centered
  - 0 it is not centered
- **xtr\_w**(*float*) – If an extra mm will be added, the number will determine the size useful to make cuts
- **xtr\_nw**(*float*) – If an extra mm will be added, the number will determine the size useful to make cuts
- **xtr\_d**(*float*) – If an extra mm will be added, the number will determine the size useful to make cuts
- **xtr\_nd**(*float*) – If an extra mm will be added, the number will determine the size useful to make cuts
- **xtr\_h**(*float*) – If an extra mm will be added, the number will determine the size useful to make cuts
- **xtr\_nh**(*float*) – If an extra mm will be added, the number will determine the size useful to make cuts
- **pos**(*FreeCAD.Vector*) – Position of the box, it can be the center one corner, or a point centered in the dimensions given by cw, cd, ch

**Returns** FreeCAD.Object with a shape of a box

**Return type** TopoShape

## Notes

**fc\_axis\_w** not necessary, unless **cw=0**, then it indicates the direction of **w**, it has to be perpendicular to the previous

`fcfun.shp_box_rot(box_w, box_d, box_h, axis_w='x', axis_nh='-z', cw=1, cd=1, ch=1)`

Makes a box with width, depth, height and then rotation will be referred to **axis\_w** = (1,0,0) and **axis\_nh** = (0,0,-1). Can be centered on any of the dimensions.

### Parameters

- **box\_w** (*float*) – The width is X
- **box\_d** (*float*) – The depth is Y
- **box\_h** (*float*) – The height is Z
- **cw** (*int*) – If 1 is centered
- **cd** (*int*) – If 1 is centered
- **ch** (*int*) – If 1 is centered
- **axis\_w** (*str*) – Can be: x, -x, y, -y, z, -z
- **axis\_nh** (*str*) – Can be: x, -x, y, -y, z, -z

## Notes

Check if it makes sense to have this small function

`fcfun.shp_boxcen(x, y, z, cx=False, cy=False, cz=False, pos=FreeCAD.Vector)`

Adds a shape of box, referenced on the specified axis, with its Placement and Rotation at zero. So it can be referenced absolutely from its given position

### Parameters

- **x** (*float*) – Length
- **y** (*float*) – Width
- **z** (*float*) – Height
- **name** (*str*) – Object Name
- **cx** (*boolean*) – Center in the length or not
- **cy** (*boolean*) – Center in the or width not
- **cz** (*boolean*) – Center in the height or not
- **pos** (*FreeCAD.Vector*) – Placement

**Returns** Shape of a box

**Return type** TopoShape

`fcfun.shp_boxcenchmf(x, y, z, chmfrad, fx=False, fy=False, fz=True, cx=False, cy=False, cz=False, pos=FreeCAD.Vector)`

Same as **shp\_boxcen** but with a chamfered dimension

### Parameters

- **x** (*float*) – Length
- **y** (*float*) – Width

- **z** (*float*) – Height
- **fillrad** (*float*) – Fillet size
- **fx** (*boolean*) – Fillet in x dimension
- **fy** (*boolean*) – Fillet in y dimension
- **fz** (*boolean*) – Fillet in z dimension
- **cx** (*boolean*) – Center in the length or not
- **cy** (*boolean*) – Center in the or width not
- **cz** (*boolean*) – Center in the height or not
- **pos** (*FreeCAD.Vector*) – Placement

**Returns** Shape of a box

**Return type** TopoShape

`fcfun.shp_boxcenfill` (*x*, *y*, *z*, *fillrad*, *fx=False*, *fy=False*, *fz=True*, *cx=False*, *cy=False*, *cz=False*,  
*pos=FreeCAD.Vector*)

Same as `shp_boxcen` but with a filleted dimension

#### Parameters

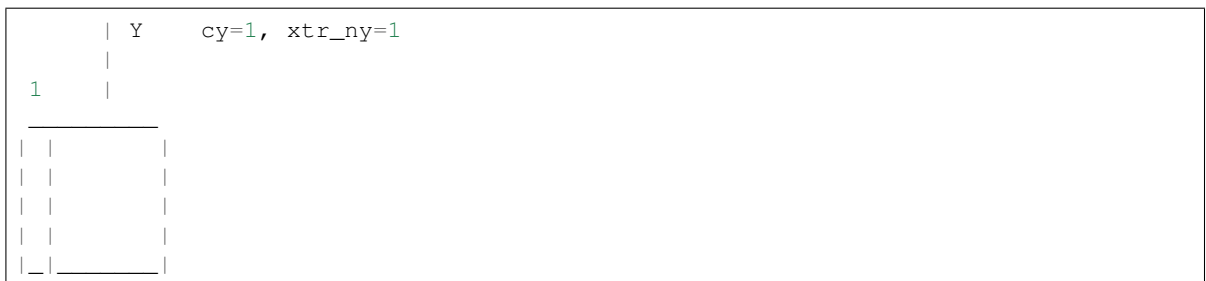
- **x** (*float*) – Length
- **y** (*float*) – Width
- **z** (*float*) – Height
- **fillrad** (*float*) – Fillet size
- **fx** (*boolean*) – Fillet in x dimension
- **fy** (*boolean*) – Fillet in y dimension
- **fz** (*boolean*) – Fillet in z dimension
- **cx** (*boolean*) – Center in the length or not
- **cy** (*boolean*) – Center in the or width not
- **cz** (*boolean*) – Center in the height or not
- **pos** (*FreeCAD.Vector*) – Placement

**Returns** Shape of a box

**Return type** TopoShape

`fcfun.shp_boxcenxtr` (*x*, *y*, *z*, *cx=False*, *cy=False*, *cz=False*, *xtr\_nx=0*, *xtr\_x=0*, *xtr\_ny=0*, *xtr\_y=0*,  
*xtr\_nz=0*, *xtr\_z=0*, *pos=FreeCAD.Vector*)

The same as `shp_boxcen`, but when it is used to cut. So sometimes it is useful to leave an extra 1mm on some sides to avoid making cuts sharing faces. The extra part is added but not influences on the reference



### Parameters

- **x** (*float*) – Length
- **y** (*float*) – Width
- **z** (*float*) – Height
- **cx** (*int*) – Center in the length or not
- **cy** (*int*) – Center in the or width not
- **cz** (*int*) – Center in the height or not
- **xtr\_x** (*float*) – Extra mm to add in positive axis of length
- **xtr\_nx** (*float*) – Extra mm to add in negative axis of length
- **xtr\_y** (*float*) – Extra mm to add in positive axis of width
- **xtr\_ny** (*float*) – Extra mm to add in negative axis of width
- **xtr\_z** (*float*) – Extra mm to add in positive axis of height
- **xtr\_nz** (*float*) – Extra mm to add in negative axis of height
- **pos** (*FreeCAD.Vector*) – Placement

**Returns** Shape of a box

**Return type** TopoShape

```
fcfun.shp_boxdir_fillchmfplane(box_w, box_d, box_h, axis_d=FreeCAD.Vector,
                                axis_w=FreeCAD.Vector, axis_h=FreeCAD.Vector,
                                cw=1, cd=1, ch=1, xtr_d=0, xtr_nd=0, xtr_w=0,
                                xtr_nw=0, xtr_h=0, xtr_nh=0, fillet=1, radius=1.0,
                                plane_fill=FreeCAD.Vector, both_planes=1,
                                edge_dir=FreeCAD.Vector, pos=FreeCAD.Vector)
```

Creates a box shape (cuboid) along 3 axis.

The shape will be filleted or chamfered on the edges of the plane defined by the plane\_fill vector. If both\_planes == 1, both faces will be filleted/chamfered if both\_planes == 0, only the face that plane\_fill is normal and goes outwards if edge\_dir has an edge direction, only those edges in that direction will be filleted/chamfered

Example of **not** centered on origin: cd=0, cw=0, ch=0

```
axis_h      . axis_d
:
:
:  _____
: /: . / | :
: /: . / | :h
: /_____ / | :
| :.....|...|...
| /      | / .
| /      | / . d
| /_____ /.....> axis_w
:
:
:.....w.....
```

Example of centered on origin: cd=1, cw=1, ch=1

```
axis_h      axis_d
```

(continues on next page)

(continued from previous page)

```

      :
      :
      : _____
      /:  :  / |
      /:  :  / | h
      /__:_/ |
      | :....|...|
      | /  :..|..|.....> axis_w
      | /      | /
      |/_      |/_
      w

```

Example of parameter both\_planes and edge\_dir

```

if both_planes == 1:
    if edge_dir == V0:
        edges_to_chamfer = [1,2,3,4,5,6,7,8]
    elif edge_dir == axis_w:
        edges_to_chamfer = [2,4,6,8]
    elif edge_dir == axis_d:
        edges_to_chamfer = [1,3,5,7]
elif both_planes == 0:
    if edge_dir == V0:
        edges_to_chamfer = [1,2,3,4]
    elif edge_dir == axis_w:
        edges_to_chamfer = [2,4]
    elif edge_dir == axis_d:
        edges_to_chamfer = [1,3]

```

```

axis_h=plane_fill
:
: _____2_____
: /:  :  / |
: 1 :      3 |
: /____4____/ |
| :...6. |...|
| /      | /
| 5      | 7
|/_8____|/.....> axis_w

```

Another example of parameter both\_planes

```

if both_planes == 1:
    edges_to_chamfer = [1,2,3,4,5,6,7,8]
elif both_planes == 0:
    edges_to_chamfer = [5,6,7,8]

axis_h
:
: _____2_____
: /:  :  / |
: 1 :      3 |
: /____4____/ |
| :...6. |...|
| /      | /
| 5      | 7
|/_8____|/.....> axis_w

```

(continues on next page)

(continued from previous page)

```

:
:
:
V
plane_fill = axis_h.negative()

```

### Parameters

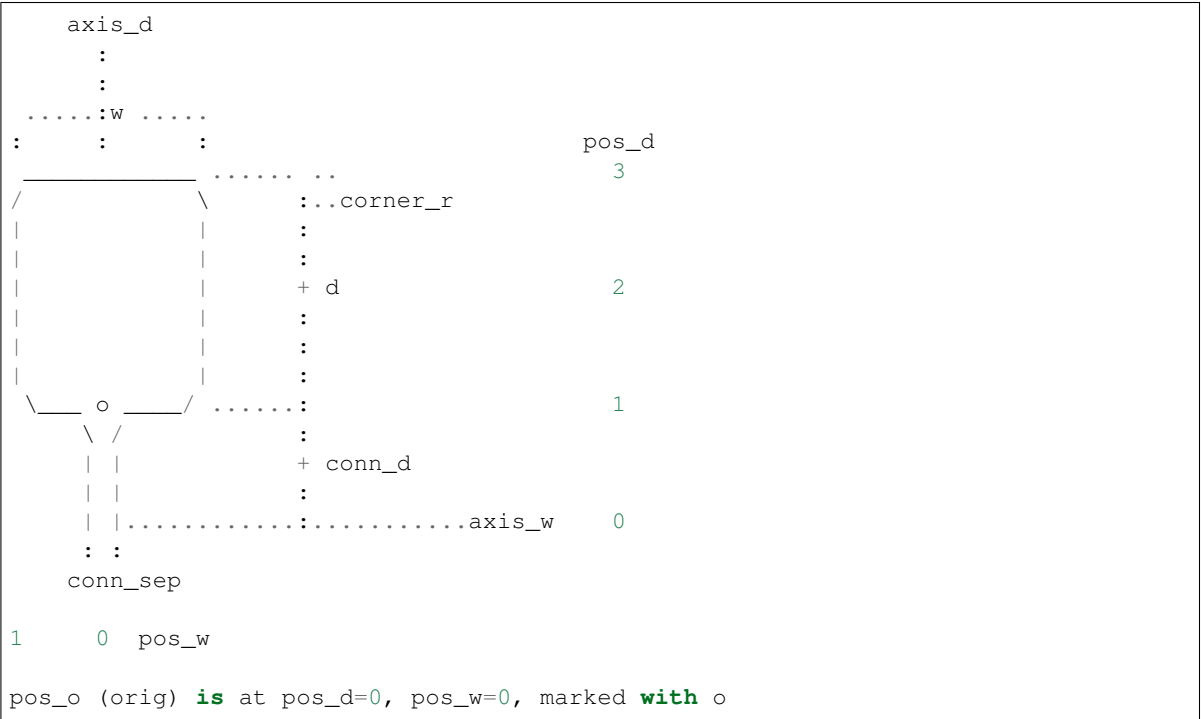
- **box\_d** (*positive float*) – Depth of the box
- **box\_w** (*positive float*) – Width of the box
- **box\_h** (*positive float*) – Height of the box
- **axis\_d** (*FreeCAD.Vector*) – Depth vector of the coordinate system
- **axis\_w** (*FreeCAD.Vector*) – Width vector of the coordinate system, can be V0 if centered and will be perpendicular to axis\_d and axis\_w
- **axis\_h** (*FreeCAD.Vector*) – Height vector of the coordinate system
- **cw** (*int*) – 1: centered along axis\_w
- **cd** (*int*) – 1: centered along axis\_d
- **ch** (*int*) – 1: centered along axis\_h
- **xtr\_d** (*float, >= 0*) – Extra depth, if there is an extra depth along axis\_d
- **xtr\_nd** (*float, >= 0*) – Extra depth, if there is an extra depth along axis\_d.negative
- **xtr\_w** (*float, >= 0*) – Extra width, if there is an extra width along axis\_w
- **xtr\_nw** (*float, >= 0*) – Extra width, if there is an extra width along axis\_w.negative
- **xtr\_h** (*float, >= 0*) – Extra height, if there is an extra height along axis\_h
- **xtr\_nh** (*float, >= 0*) – Extra height, if there is an extra height along axis\_h.negative
- **fillet** (*int*) –
  - 1: to fillet the edges
  - 0: to chamfer the edges
- **radius** (*float >= 0*) – radius of the fillet/chamfer
- **plane\_fill** (*FreeCAD.Vector*) – Vector perpendicular to the face that is going to be filleted/chamfered
- **both\_planes** (*int*) –
  - 0: fillet/chamfer only the edges on the face perpendicular to plane\_fill and on the face that plane\_fill goes outwards. See drawing
  - 1: fillet/chamfer the edges on both faces perpendicular to plane\_fill
- **edge\_dir** (*FreeCAD.Vector*) –
  - V0: fillet/chamfer all the edges of that/those faces
  - Axis: fillet/chamfer only the edges of that/those faces that are parallel to this axis
- **pos** (*FreeCAD.Vector*) – Position of the box

**Returns** Shape of the filleted/chamfered box

### Return type TopoShape

`fcfun.shp_cableturn(d, w, thick_d, corner_r, conn_d, conn_sep, xtr_conn_d=0, closed=0, axis_d=FreeCAD.Vector, axis_w=FreeCAD.Vector, pos_d=0, pos_w=0, pos=FreeCAD.Vector)`

Creates a shape of an electrical cable turn, in any direction But it is a shape in FreeCAD See function `wire_cableturn`



### Parameters

- **d** (*float*) – Depth/length of the turn
- **w** (*float*) – Width of the turn
- **thick\_d** (*float*) – Diameter of the wire
- **corner\_r** (*float*) – Radius of the corners
- **conn\_d** (*float*) – Depth/length of the connector part
  - 0: there is no connecting wire
- **xtr\_conn\_d** (*float*) – If `conn_d > 0`, there can be an extra length of connector to make unions, it will not be counted as `pos_d = 0`. It will not work well if it is closed
- **conn\_sep** (*float*) – Separation of the connectors
- **closed** (*boolean*) –
  - 0 : the ends are not closed
  - 1 : the ends are closed
- **axis\_d** (*FreeCAD.Vector*) – Coordinate System Vector along the depth
- **axis\_w** (*FreeCAD.Vector*) – Coordinate System Vector along the width
- **pos\_d** (*int*) – Location of pos along the axis\_d (0,1,2,3), see drawing



- 0: reference at the beginning of the connector
- 1: reference at the beginning of the turn, at the side of the connector
- 2: reference at the middle of the turn
- 3: reference at the end of the turn
- **pos\_w** (*int*) – Location of pos along the axis\_w (0,1), see drawing
  - 0: reference at the center of symmetry
  - 1: reference at the end of the turn
- **pos** (*FreeCAD.Vector*) – Position of the reference

**Returns** FreeCAD Shape of a electrical wire

**Return type** Shape

`fcfun.shp_cir_fillchmf` (*shp, circen\_pos=FreeCAD.Vector, fillet=1, radius=1*)

Fillet or chamfer edges that is a circle, the shape has to be a cylinder

**Parameters**

- **shp** (*Shape*) – Original cylinder shape we want to fillet or chamfer
- **circen\_pos** (*FreeCAD.Vector*) – Center of the circle
- **fillet** (*int*) –
  - 1 if we are doing a fillet
  - 0 if it is a chamfer
- **radius** (*float*) – Radius of the fillet or chamfer

**Returns** FreeCAD Shape with fillet/chamfer made

**Return type** Shape

`fcfun.shp_cyl` (*r, h, normal=FreeCAD.Vector, pos=FreeCAD.Vector*)

Same as addCylPos, but just creates the shape

**Parameters**

- **r** (*float*) – Radius,
- **h** (*float*) – Height
- **normal** (*FreeCAD.Vector*) – FreeCAD.Vector pointing to the normal (if its module is not one, the height will be larger than h)
- **pos** (*FreeCAD.Vector*) – Position of the cylinder

**Returns** FreeCAD Shape of a cylinder

**Return type** Shape

`fcfun.shp_cyl_gen` (*r, h, axis\_h=FreeCAD.Vector, axis\_ra=None, axis\_rb=None, pos\_h=0, pos\_ra=0, pos\_rb=0, xtr\_top=0, xtr\_bot=0, xtr\_r=0, pos=FreeCAD.Vector*)

This is a generalization of shp\_cylcenxtr. Makes a cylinder in any position and direction, with optional extra heights and radius, and various locations in the cylinder

## Chapter 1. Table Of Contents

```
axis_rb
:
```

(continues on next page)

(continued from previous page)

```

: . .
: . . .
( . . . )
x . . . . .> axis_ra

axis_h
:
:
: .....
: ____:____:.....: xtr_top
||      ||
||      ||
||      ||
|x      |.....>axis_ra
||      ||
||      ||
||_____|.....
:.....o.....:.....: xtr_bot
:;
xtr_r

```

### Parameters

- **r** (*float*) – Radius of the cylinder
- **h** (*float*) – Height of the cylinder
- **axis\_h** (*FreeCAD.Vector*) – Vector along the cylinder height
- **axis\_ra** (*FreeCAD.Vector*) – Vector along the cylinder radius, a direction perpendicular to axis\_h only make sense if pos\_ra = 1. It can be None.
- **axis\_rb** (*FreeCAD.Vector*) – Vector along the cylinder radius, a direction perpendicular to axis\_h and axis\_rb only make sense if pos\_rb = 1 It can be None
- **pos\_h** (*int*) – Location of pos along axis\_h (0, 1)
  - 0: the cylinder pos is centered along its height
  - 1: the cylinder pos is at its base (not considering xtr\_h)
- **pos\_ra** (*int*) – Location of pos along axis\_ra (0, 1)
  - 0: pos is at the circumference center
  - 1: pos is at the circumsference, on axis\_ra, at r from the circle center (not at r + xtr\_r)
- **pos\_rb** (*int*) – Location of pos along axis\_rb (0, 1)
  - 0: pos is at the circumference center
  - 1: pos is at the circumsference, on axis\_rb, at r from the circle center (not at r + xtr\_r)
- **xtr\_top** (*float*) – Extra height on top, it is not taken under consideration when calculating the cylinder center along the height
- **xtr\_bot** (*float*) – Extra height at the bottom, it is not taken under consideration when calculating the cylinder center along the height or the position of the base
- **xtr\_r** (*float*) – Extra length of the radius, it is not taken under consideration when calculating pos\_ra or pos\_rb

- **pos** (*FreeCAD.Vector*) – Position of the cylinder, taking into account where the center is

**Returns** FreeCAD Shape of a cylinder

**Return type** Shape

`fcfun.shp_cylcenxtr(r, h, normal=FreeCAD.Vector, ch=1, xtr_top=0, xtr_bot=0, pos=FreeCAD.Vector)`

Add cylinder, can be centered on the position, and also can have an extra mm on top and bottom to make cuts

**Parameters**

- **r** (*float*) – Radius
- **h** (*float*) – Height
- **normal** (*FreeCAD.Vector*) – FreeCAD.Vector pointing to the normal
- **ch** (*int*) – Centered on the middle, of the height
- **xtr\_top** (*float*) – Extra on top (but does not influence the centering)
- **xtr\_bot** (*float*) – Extra on bottom (but does not influence the centering)
- **pos** (*FreeCAD.Vector*) – Position of the cylinder

**Returns** FreeCAD Shape of a cylinder

**Return type** Shape

`fcfun.shp_cylfilletchamfer(shp, fillet=1, radius=1)`

Fillet or chamfer all edges of a cylinder

**Parameters**

- **shp** (*Shape*) – Original cylinder shape we want to fillet or chamfer
- **fillet** (*int*) –
  - 1 if we are doing a fillet
  - 0 if it is a chamfer
- **radius** (*float*) – Radius of the fillet or chamfer

**Returns** FreeCAD Shape with fillet/chamfer made

**Return type** Shape

`fcfun.shp_cylhole(r_ext, r_int, h, axis='z', h_disp=0.0)`

Same as addCylHole, but just a shape

Add cylinder, with inner hole:

**Parameters**

- **r\_ext** (*float*) – External radius,
- **r\_int** (*float*) – Internal radius,
- **h** (*float*) – Height
- **axis** (*str*) – 'x', 'y' or 'z'
  - 'x' will along the x axis
  - 'y' will along the y axis
  - 'z' will be vertical

- **h\_disp** (*int*) – Displacement on the height.
  - if 0, the base of the cylinder will be on the plane
  - if -h/2: the plane will be cutting h/2

**Returns** FreeCAD Shape of a cylinder with hole

**Return type** Shape

```
fcfun.shp_cylhole_arc(r_out, r_in, h, axis_h=FreeCAD.Vector, axis_ra=None, axis_rb=None,
                     end_angle=360, pos_h=0, pos_ra=0, pos_rb=0, xtr_top=0, xtr_bot=0,
                     xtr_r_out=0, xtr_r_in=0, pos=FreeCAD.Vector)
```

This is similar to make shp\_cylhole\_gen but not for a whole, just an arc. I don't know how where makeCircle starts its startangle and end angle That is why I use this way

Makes a hollow cylinder in any position and direction, with optional extra heights, and inner and outer radius, and various locations in the cylinder

```
pos_h = 1, pos_ra = 0, pos_rb = 0
pos at 1:
    axis_rb
    :
    :
    . .
    . . . .
    ( ( 0 ) ) ---- axis_ra
    . . . .
    . .

    axis_h
    :
    :

    .....
:____:____:....: xtr_top
| :      : |
| :      : |
| :      : |
| :  0  : |      0: pos would be at 0, if pos_h == 0
| :      : |
| :      : |
|_:_1_:_|....>axis_ra
:..o...:.....: xtr_bot      This o will be pos_o (orig)
: : :
: : :
: + :
:r_in:
:   :
:..:
+
r_out
```

Values **for** pos\_ra (similar to pos\_rb along it axis)

```
axis_h
:
:
.....
```

(continues on next page)

(continued from previous page)

```

:____:____:....: xtr_top
| :      : |
| :      : |
| :      : |
2 1 0 : |....>axis_ra      (if pos_h == 0)
| :      : |
| :      : |
|_:____:_:|.....
:....o.....: xtr_bot      This o will be pos_o (orig)
: : :
: : :
: + :
:r_in:
: : :
:....:
+
r_out

```

### Parameters

- **r\_out** (*float*) – Radius of the outside cylinder
- **r\_in** (*float*) – Radius of the inner hole of the cylinder
- **h** (*float*) – Height of the cylinder
- **axis\_h** (*FreeCAD.Vector*) – Vector along the cylinder height
- **axis\_ra** (*FreeCAD.Vector*) – Vector along the cylinder radius, a direction perpendicular to axis\_h it is not necessary if pos\_ra == 0 It can be None, but if None, axis\_rb has to be None Defines the starting angle
- **axis\_rb** (*FreeCAD.Vector*) – Vector along the cylinder radius, a direction perpendicular to axis\_h and axis\_ra it is not necessary if pos\_ra == 0 It can be None
- **end\_angle** (*float (in degrees)*) – Rotating from axis\_ra in the direction determined by axis\_h
- **pos\_h** (*int*) – Location of pos along axis\_h (0, 1)
  - 0: the cylinder pos is centered along its height, not considering xtr\_top, xtr\_bot
  - 1: the cylinder pos is at its base (not considering xtr\_h)
- **pos\_ra** (*int*) – Location of pos along axis\_ra (0, 1)
  - 0: pos is at the circumference center
  - 1: pos is at the inner circumsference, on axis\_ra, at r\_in from the circle center (not at r\_in + xtr\_r\_in)
  - 2: pos is at the outer circumsference, on axis\_ra, at r\_out from the circle center (not at r\_out + xtr\_r\_out)
- **pos\_rb** (*int*) – Location of pos along axis\_ra (0, 1)
  - 0: pos is at the circumference center
  - 1: pos is at the inner circumsference, on axis\_rb, at r\_in from the circle center (not at r\_in + xtr\_r\_in)
  - 2: pos is at the outer circumsference, on axis\_rb, at r\_out from the circle center (not at r\_out + xtr\_r\_out)

- Returns** FreeCAD Shape of a arc of the cylinder
- Return type** Shape

This is a generalization of `shp_cylholedir` and `shp_cylhole`. Makes a hollow cylinder in any position and direction, with optional extra heights, and inner and outer radius, and various locations in the cylinder.

```

pos_h = 1, pos_ra = 0, pos_rb = 0
pos at 1:
    axis_rb
    :
    :
    . .      o: are n_bolt(4) holes
    .o. .o.
( ( 0 ) ) ---- axis_ra
    .o. .o.
    . .

    axis_h
    :
    :
    .....
:___:___:....: xtr_top
| :      : |
| :      : |
| :      : |
| :  0   : |      0: pos would be at 0, if pos_h == 0
| :      : |
| :      : |
|_:_1_:_|....>axis_ra
:....o...:.....: xtr_bot      This o will be pos_o (orig)
: : :
: : :
: + :
:r_in:
: :
:....:
+

```

123

(continued from previous page)

```

r_out

Values for pos_ra (similar to pos_rb along it axis)

    axis_h
    :
d_bolt :
    :.....
    :_:~_:~_:~_:~_:~_: xtr_top
    | : : : : : |
    | : : : : : |
    | : : : : : |
    3 2 1 0 : : |....>axis_ra (if pos_h == 0)
    | : : : : : |
    | : : : : : |
    |_:~_:~_:~_:~_:~_:
    :..:~:~:~:~:~:~: xtr_bot      This o will be pos_o (orig)
    : : : :
    : : :~:~:
    : : + :
    : : r_in
    : :~:~:~:~:
    : +
    : r_bolt2cen:
    : :
    : :~:~:~:~:
    +
    r_out

```

### Parameters

- **r\_out** (*float*) – Radius of the outside cylinder
- **r\_in** (*float*) – Radius of the inner hole of the cylinder
- **h** (*float*) – Height of the cylinder
- **n\_bolt** (*int*) – Number of bolt holes, if zero no bolt holes
- **d\_bolt** (*float*) – Diameter of the bolt holes
- **r\_bolt2cen** (*float*) – Distance (radius) from the cylinder center to the bolt hole centers
- **bolt\_axis\_ra** (*int*) –
  - 1: the first bolt will be on axis ra
  - 0: the first bolt will be rotated half of the angle between to bolt holes -> centered on the side
- **axis\_h** (*FreeCAD.Vector*) – Vector along the cylinder height
- **axis\_ra** (*FreeCAD.Vector*) – Vector along the cylinder radius, a direction perpendicular to axis\_h it is not necessary if pos\_ra == 0 It can be None, but if None, axis\_rb has to be None
- **axis\_rb** (*FreeCAD.Vector*) – Vector along the cylinder radius, a direction perpendicular to axis\_h and axis\_ra it is not necessary if pos\_ra == 0 It can be None
- **pos\_h** (*int*) – Location of pos along axis\_h (0, 1)



- 0: the cylinder pos is centered along its height, not considering xtr\_top, xtr\_bot
- 1: the cylinder pos is at its base (not considering xtr\_h)
- **pos\_ra** (*int*) – Location of pos along axis\_ra (0, 1)
  - 0: pos is at the circumference center
  - 1: pos is at the inner circumsference, on axis\_ra, at r\_in from the circle center (not at r\_in + xtr\_r\_in)
  - 2: pos is at the center of the bolt hole (one of them)
  - 3: pos is at the outer circumsference, on axis\_ra, at r\_out from the circle center (not at r\_out + xtr\_r\_out)
- **pos\_rb** (*int*) – Location of pos along axis\_rb (0, 1)
  - 0: pos is at the circumference center
  - 1: pos is at the inner circumsference, on axis\_rb, at r\_in from the circle center (not at r\_in + xtr\_r\_in)
  - 2: pos is at the center of the bolt hole (one of them)
  - 3: pos is at the outer circumsference, on axis\_rb, at r\_out from the circle center (not at r\_out + xtr\_r\_out)
- **xtr\_top** (*float*) – Extra height on top, it is not taken under consideration when calculating the cylinder center along the height
- **xtr\_bot** (*float*) – Extra height at the bottom, it is not taken under consideration when calculating the cylinder center along the height or the position of the base
- **xtr\_r\_in** (*float*) – Extra length of the inner radius (hollow cylinder), it is not taken under consideration when calculating pos\_ra or pos\_rb. It can be negative, so this inner radius would be smaller
- **xtr\_r\_out** (*float*) – Extra length of the outer radius it is not taken under consideration when calculating pos\_ra or pos\_rb. It can be negative, so this outer radius would be smaller
- **pos** (*FreeCAD.Vector*) – Position of the cylinder, taking into account where the center is

**Returns** FreeCAD Shape of a cylinder with hole

**Return type** Shape

`fcfun.shp_cylhole_gen(r_out, r_in, h, axis_h=FreeCAD.Vector, axis_ra=None, axis_rb=None, pos_h=0, pos_ra=0, pos_rb=0, xtr_top=0, xtr_bot=0, xtr_r_out=0, xtr_r_in=0, pos=FreeCAD.Vector)`

This is a generalization of `shp_cylholedir`. Makes a hollow cylinder in any position and direction, with optional extra heights, and inner and outer radius, and various locations in the cylinder

```
pos_h = 1, pos_ra = 0, pos_rb = 0
pos at 1:
    axis_rb
    :
    :
    . .
    . . . .
    ( ( 0 ) ) ---- axis_ra
    . . . .
```

(continues on next page)

(continued from previous page)

```

    . .
    axis_h
      :
      :
    .....
:____:____:....: xtr_top
| :      : |
| :      : |
| :      : |
| :  0  : |      0: pos would be at 0, if pos_h == 0
| :      : |
| :      : |
|_:_1_:_|....>axis_ra
:....o.....: xtr_bot      This o will be pos_o (orig)
: : :
: :..:
: + :
:r_in:
:   :
:....:
+
r_out

```

Values **for** pos\_ra (similar to pos\_rb along it axis)

```

    axis_h
      :
      :
    .....
:____:____:....: xtr_top
| :      : |
| :      : |
| :      : |
2 1  0 : |....>axis_ra      (if pos_h == 0)
| :      : |
| :      : |
|_:_:_:_|.....
:....o.....: xtr_bot      This o will be pos_o (orig)
: : :
: :..:
: + :
:r_in:
:   :
:....:
+
r_out

```

### Parameters

- **r\_out** (*float*) – Radius of the outside cylinder
- **r\_in** (*float*) – Radius of the inner hole of the cylinder
- **h** (*float*) – Height of the cylinder

- **axis\_h** (*FreeCAD.Vector*) – Vector along the cylinder height
- **axis\_ra** (*FreeCAD.Vector*) – Vector along the cylinder radius, a direction perpendicular to axis\_h it is not necessary if pos\_ra == 0 It can be None, but if None, axis\_rb has to be None
- **axis\_rb** (*FreeCAD.Vector*) – Vector along the cylinder radius, a direction perpendicular to axis\_h and axis\_ra it is not necessary if pos\_ra == 0 It can be None
- **pos\_h** (*int*) – Location of pos along axis\_h (0, 1)
  - 0: the cylinder pos is centered along its height, not considering xtr\_top, xtr\_bot
  - 1: the cylinder pos is at its base (not considering xtr\_h)
- **pos\_ra** (*int*) – Location of pos along axis\_ra (0, 1)
  - 0: pos is at the circumference center
  - 1: pos is at the inner circumference, on axis\_ra, at r\_in from the circle center (not at r\_in + xtr\_r\_in)
  - 2: pos is at the outer circumference, on axis\_ra, at r\_out from the circle center (not at r\_out + xtr\_r\_out)
- **pos\_rb** (*int*) – Location of pos along axis\_rb (0, 1)
  - 0: pos is at the circumference center
  - 1: pos is at the inner circumference, on axis\_rb, at r\_in from the circle center (not at r\_in + xtr\_r\_in)
  - 2: pos is at the outer circumference, on axis\_rb, at r\_out from the circle center (not at r\_out + xtr\_r\_out)
- **xtr\_top** (*float*) – Extra height on top, it is not taken under consideration when calculating the cylinder center along the height
- **xtr\_bot** (*float*) – Extra height at the bottom, it is not taken under consideration when calculating the cylinder center along the height or the position of the base
- **xtr\_r\_in** (*float*) – Extra length of the inner radius (hollow cylinder), it is not taken under consideration when calculating pos\_ra or pos\_rb. It can be negative, so this inner radius would be smaller
- **xtr\_r\_out** (*float*) – Extra length of the outer radius it is not taken under consideration when calculating pos\_ra or pos\_rb. It can be negative, so this outer radius would be smaller
- **pos** (*FreeCAD.Vector*) – Position of the cylinder, taking into account where the center is

**Returns** FreeCAD Shape of a cylinder with hole

**Return type** Shape

`fcfun.shp_cylholedir(r_out, r_in, h, normal=FreeCAD.Vector, pos=FreeCAD.Vector)`

Same as addCylHolePos, but just a shape Same as shp\_cylhole, but this one accepts any normal

#### Parameters

- **r\_out** (*float*) – Outside radius
- **r\_in** (*float*) – Inside radius
- **h** (*float*) – Height

- **normal** (*FreeCAD.Vector*) – FreeCAD.Vector pointing to the normal (if its module is not one, the height will be larger than h)
- **pos** (*FreeCAD.Vector*) – Position of the cylinder

**Returns** FreeCAD Shape of a cylinder with hole

**Return type** Shape

`fcfun.shp_extrud_face` (*face, length, vec\_extr\_axis, centered=0*)

Extrudes a face on any plane

#### Parameters

- **face** (*FreeCAD.Face*) – Face to be extruded.
- **length** (*float*) – Extrusion length
- **centered** (*int*) – 1 if the extrusion is centered (simetrical) 0 if it is not
- **vec\_extr\_axis** (*FreeCAD.Vector*) – Typically, it will be the same as `vec_facenormal`. by default, if it is 0, it will be equal to `vec_facenormal` It doesn't have to be on an axis, it can be diagonally

**Returns** FreeCAD Shape of the Face

**Return type** Shape

`fcfun.shp_extrud_face_rot` (*face, vec\_facenormal, vec\_edgx, length, centered=0, vec\_extr\_axis=0*)

Extrudes a face that is on plane XY, includes a rotation



#### Parameters

- **face** (*FreeCAD.Face*) – Face to be extruded. On plane XY
- **vec\_facenormal** (*FreeCAD.Vector*) – Indicates where the normal of the face will point. The normal of the original face is VZ, but this function may rotate it depending on this argument It has to be on an axis: 'x', 'y', ..
- **vec\_edgx** (*FreeCAD.Vector*) – Indicates where the edge X will be after the rotation It has to be on an axis: 'x', 'y', ..
- **length** (*float*) – Extrusion length
- **centered** (*int*) – 1 if the extrusion is centered (simetrical) 0 if it is not
- **vec\_extr\_axis** (*FreeCAD.Vector*) – Typically, it will be the same as `vec_facenormal`. by default, if it is 0, it will be equal to `vec_facenormal` It doesn't have to be on an axis, it can be diagonally

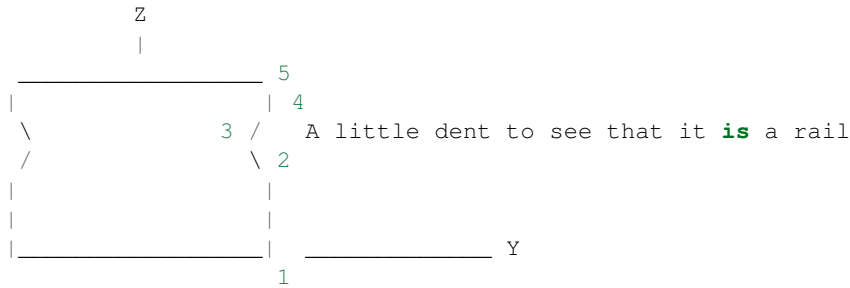
**Returns** FreeCAD Shape of a face

**Return type** Shape

`fcfun.shp_face_lgrail` (*rail\_w, rail\_h, axis\_l='x', axis\_b='-z'*)

Adds a shape of the profile (face) of a linear guide rail, the dent is just rough, to be able to see that it is a profile

It will be centered on the width axis, **and** zero on the length **and** height



#### Parameters

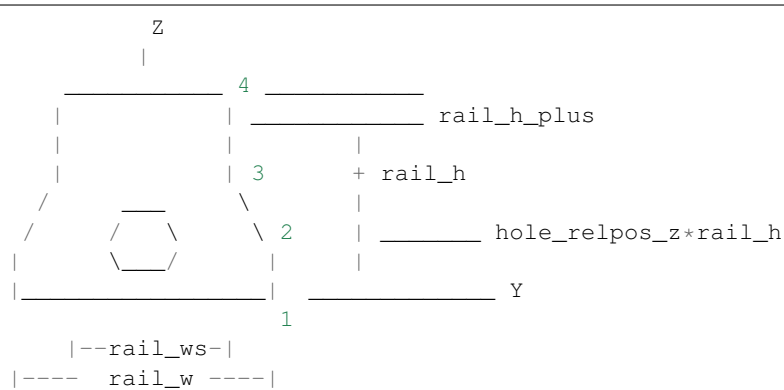
- **rail\_w**(*float*) – Width of the rail
- **rail\_h**(*float*) – Height of the rail
- **axis\_l**(*str*) – Axis where the length of the rail is: 'x', 'y', 'z'
- **axis\_b**(*str*) – Axis where the base of the rail is pointing: 'x', 'y', 'z', '-x', '-y', '-z',

**Returns** FreeCAD Shape Face of a rail

**Return type** Shape

`fcfun.shp_face_rail(rail_w, rail_ws, rail_h, rail_h_plus=0, offs_w=0, offs_h=0, axis_l='x', axis_b='-z', hole_d=0, hole_relpos_z=0.4)`

Adds a shape of the profile (face) of a rail



#### Parameters

- **rail\_w**(*float*) – Width of the rail
- **rail\_ws**(*float*) – Small width of the rail
- **rail\_h**(*float*) – Height of the rail
- **rail\_h\_plus**(*float*) – Above the rail can be some height to attach, o whatever it is not included on rail\_h
- **offs\_w**(*float*) – Offset on the width, to make the hole
- **offs\_h**(*float*) – Offset on the heigth, to make the hole
- **axis\_l**(*str*) – The axis where the length of the rail is: 'x', 'y', 'z'

- **axis\_b** (*str*) – The axis where the base of the rail is pointing: 'x', 'y', 'z', '-x', '-y', '-z', It will be centered on the width axis, and zero on the length and height
- **hole\_d** (*float*) – Diameter of a hole inside the rail. To have a leadscrew
- **hole\_relpos\_z** (*float*) – Relative position of the center of the hole, relative to the height (the rail\_h, not the total height (rail\_h+rail\_h\_plus))

**Returns** FreeCAD Shape Face of a rail

**Return type** Shape

`fcfun.shp_filletchamfer(shp, e_len, fillet=1, radius=1, axis='x', xpos_chk=0, ypos_chk=0, zpos_chk=0, xpos=0, ypos=0, zpos=0)`

Fillet or chamfer edges of a certain length, on a certain axis and a certain coordinate

#### Parameters

- **shp** (*Shape*) – Original shape we want to fillet or chamfer
- **fillet** (*int*) –
  - 1 if we are doing a fillet
  - 0 if it is a chamfer
- **e\_len** (*float*) – Length of the edges that we want to fillet or chamfer if e\_len == 0, chamfer/fillet any length
- **radius** (*float*) – Radius of the fillet or chamfer
- **axis** (*str*) – Axis where the fillet will be
- **xpos\_chk** (*int*) – If the position will be checked.
- **ypos\_chk** (*int*) – If the position will be checked.
- **zpos\_chk** (*int*) – If the position will be checked.
- **xpos** (*float*) – The X position
- **ypos** (*float*) – The Y position
- **zpos** (*float*) – The Z position

#### Notes

If axis = 'x', x\_pos\_check will not make sense

**Returns** FreeCAD Shape with fillet/chamfer made

**Return type** Shape

`fcfun.shp_filletchamfer_dir(shp, fc_axis=FreeCAD.Vector, fillet=1, radius=1)`

Fillet or chamfer edges on a certain axis

#### Parameters

- **shp** (*Shape*) – Original shape we want to fillet or chamfer
- **fillet** (*int*) –
  - 1 if we are doing a fillet
  - 0 if it is a chamfer
- **radius** (*float*) – The radius of the fillet or chamfer

- **fc\_axis** (*FreeCAD.Vector*) – Axis where the fillet will be

**Returns** FreeCAD Shape with fillet/chamfer made

**Return type** Shape

`fcfun.shp_filletchamfer_dirpt` (*shp, fc\_axis=FreeCAD.Vector, fc\_pt=FreeCAD.Vector, fillet=1, radius=1*)

Fillet or chamfer edges on a certain axis and a point contained in that axis

**Parameters**

- **shp** (*Shape*) – Original shape we want to fillet or chamfer
- **fc\_axis** (*FreeCAD.Vector*) – Axis where the fillet will be
- **fc\_pt** (*FreeCAD.Vector*) – Placement of the point
- **fillet** (*int*) –
  - 1 if we are doing a fillet
  - 0 if it is a chamfer
- **radius** (*float*) – Radius of the fillet or chamfer

**Returns** FreeCAD Shape with fillet/chamfer made

**Return type** Shape

`fcfun.shp_filletchamfer_dirpts` (*shp, fc\_axis, fc\_pts, fillet=1, radius=1*)

Fillet or chamfer edges on a certain axis and a list of point contained in that axis

**Parameters**

- **shp** (*Shape*) – Original shape we want to fillet or chamfer
- **fc\_axis** (*FreeCAD.Vector*) – Axis where the fillet will be
- **fc\_pts** (*FreeCAD.Vector*) – Vector list of the points
- **fillet** (*int*) –
  - 1 if we are doing a fillet
  - 0 if it is a chamfer
- **radius** (*float*) – Radius of the fillet or chamfer

**Returns** FreeCAD Shape with fillet/chamfer made

**Return type** Shape

`fcfun.shp_filletchamfer_dirs` (*shp, fc\_axis\_l, fillet=1, radius=1*)

Same as `shp_filletchamfer_dir`, but with a list of directions

**Parameters**

- **shp** (*Shape*) – Original shape we want to fillet or chamfer
- **fc\_axis\_l** (*list*) – List of *FreeCAD.Vector*. Each vector indicates the axis where the fillet/chamfer will be
- **fillet** (*int*) –
  - 1 if we are doing a fillet
  - 0 if it is a chamfer
- **radius** (*float*) – Radius of the fillet or chamfer

**Returns** FreeCAD Shape with fillet/chamfer made

**Return type** Shape

`fcfun.shp_hollowbelt_dir` (*center\_sep*, *rad1*, *rad2*, *rad\_thick*, *height*, *fc\_axis\_h*=FreeCAD.Vector,  
*fc\_axis\_l*=FreeCAD.Vector, *ref\_l*=1, *ref\_h*=1, *xtr\_h*=0, *xtr\_nh*=0,  
*pos*=FreeCAD.Vector)

Makes a shape of 2 tangent circles (like a belt joining 2 circles). check shp\_belt\_wire\_dir

**Parameters**

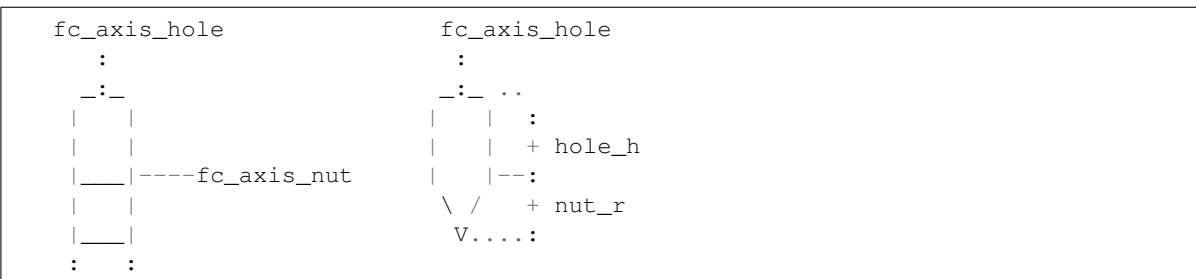
- **center\_sep** (*float*) – Separation of the circle centers
- **rad1** (*float*) – Internal radius of the first circle, on the opposite direction of *fc\_axis\_l*
- **rad2** (*float*) – Internal radius of the second circle, on the direction of *fc\_axis\_l*
- **rad\_thick** (*float*) – Increment to *rad1* and *rad2* to make the thickness.
- **height** (*float*) – Height of the shape
- **fc\_axis\_l** (FreeCAD.Vector) – Vector on the direction circle centers, pointing to *rad2*
- **fc\_axis\_h** (FreeCAD.Vector) – Vector on the hieght direction
- **ref\_l** (*int*) – Reference (zero) of the *fc\_axis\_l*
  - 1: reference on the center
  - 2: reference at one of the semicircle centers (point 2) the other circle center will be on the direction of *fc\_axis\_l*
  - 3: reference at the end of *rad1* circle the other end will be on the direction of *fc\_axis\_l*
- **ref\_h** (*int*) –
  - 1: reference is at the center of the height
  - 2: reference is at the bottom
- **xtr\_h** (*float*) – If >0 it will be that extra height on the direction of *fc\_axis\_h*
- **xtr\_nh** (*float*) – If >0 it will be that extra height on the opositve direction of *fc\_axis\_h*
- **pos** (FreeCAD.Vector) – Position of the reference

**Returns** FreeCAD Shape

**Return type** Shape

`fcfun.shp_nuthole` (*nut\_r*, *nut\_h*, *hole\_h*, *xtr\_nut*=1, *xtr\_hole*=1, *fc\_axis\_nut*=FreeCAD.Vector,  
*fc\_axis\_hole*=FreeCAD.Vector, *ref\_nut\_ax*=1, *ref\_hole\_ax*=1,  
*pos*=FreeCAD.Vector)

Similar to NutHole, but creates a shape, in any direction. Add a Nut hole (hexagonal) with a prism attached to introduce the nut tolerances are included



(continues on next page)



(continued from previous page)

```

:....:
+ nut_h

ref_nut:

fc_axis_hole          fc_axis_hole
:                      :
_:_-                  _:_-
|_|                  |_|
|_|                  |_|
2_1_|----fc_axis_nut  \ /
|_|                  V
|_|

ref_hole:

fc_axis_hole          fc_axis_hole
:                      :
_2_                  _2_
|_|                  |_|
|_|                  |_|
_1_|----fc_axis_nut  | 1 |
|_|                  \ /
|_|                  V
|_|

fc_axis_hole
:
_:_- ...
|.2.|...xtr_hole (but pos is not referenced on the xtr)
|_|
|_|
|_1_|----fc_axis_nut
|_|          ____ but pos is still referenced on the axis of
|_|.....    |_| the shank
                xtr_nut....|_|

```

### Parameters

- **nut\_r** (*float*) – Circumradius of the hexagon
- **nut\_h** (*float*) – Height of the nut, usually larger than the actual nut height, to be able to introduce it
- **hole\_h** (*float*) – The hole height, from the center of the hexagon to the side it will see light
- **xtr\_nut** (*int*) – 1 if you want 1 mm out of the hole, to cut
- **xtr\_hole** (*int*) – 1 if you want 1 mm out of the hole, to cut
- **fc\_axis\_nut** (*FreeCAD.Vector*) – Axis of the shank of the nut
- **fc\_axis\_hole** (*FreeCAD.Vector*) – Axis of the shank of the nut
- **ref\_nut\_ax** (*int*) – If it is referenced to the center, symmetrical point on the on the `fc_axis_nut`
- **ref\_hole\_ax** (*int*) – If it is referenced at the center of the shank, or at the end of the hole, not counting extra

- **pos** (*FreeCAD.Vector*) – Position

**Returns** FreeCAD Shape of a nut hole

**Return type** Shape

`fcfun.shp_regpolygon_dir_face(n_sides, radius, fc_normal=FreeCAD.Vector, fc_verx1=FreeCAD.Vector, pos=FreeCAD.Vector)`

Similar to `shp_regpolygon_face`, but in any direction of the space makes the shape of a face of a regular polygon

**Parameters**

- **n\_sides** (*int*) – Number of sides of the polygon
- **radius** (*float*) – Circumradius of the polygon
- **fc\_normal** (*FreeCAD.Vector*) – Direction of the normal
- **fc\_verx1** (*FreeCAD.Vector*) – Direction of the first vertex
- **pos** (*FreeCAD.Vector*) – Position of the center. Default (0,0,0)

**Returns** FreeCAD Face of a regular polygon

**Return type** Shape Face

`fcfun.shp_regpolygon_face(n_sides, radius, n_axis='z', v_axis='x', edge_rot=0, pos=FreeCAD.Vector)`

Makes the shape of a face of a regular polygon

**Parameters**

- **n\_sides** (*int*) – Number of sides of the polygon
- **radius** (*float*) – Circumradius of the polygon
- **n\_axis** (*str*) – Axis of the normal: 'x', '-x', 'y', '-y', 'z', '-z'
- **v\_axis** (*str*) – Perpendicular to `n_axis`, pointing to the first vertex, unless, `x_angle` is != 0. the vertex will be rotated `x_angle` degrees for `v_axis`
- **x\_angle** (*float*) – If zero, the first vertex will be on axis `v_axis` if `x_angle` != 0, it will rotated some angle
- **pos** (*FreeCAD.Vector*) – Position of the center. Default (0,0,0)

**Returns** FreeCAD Face of a regular polygon

**Return type** Shape Face

`fcfun.shp_regprism(n_sides, radius, length, n_axis='z', v_axis='x', centered=0, edge_rot=0, pos=FreeCAD.Vector)`

Makes a shape of a face of a regular polygon

**Parameters**

- **n\_sides** (*int*) – Number of sides of the polygon
- **radius** (*float*) – Circumradius of the polygon
- **length** (*float*) – Length of the polygon
- **n\_axis** (*str*) – Axis of the normal: 'x', '-x', 'y', '-y', 'z', '-z'
- **v\_axis** (*str*) – Perpendicular to `n_axis`, pointing to the first vertex, unless, `x_angle` is != 0. the vertex will be rotated `x_angle` degrees for `v_axis`
- **centered** (*int*) – 1 if the extrusion is centered on `pos` (symmetrical)

- **x\_angle** (*float*) – if zero, the first vertex will be on axis v\_axis if x\_angle != 0, it will rotated some angle
- **pos** (*FreeCAD.Vector*) – Position of the center. Default (0,0,0)

**Returns** FreeCAD Shape of a regular prism

**Return type** Shape

```
fcfun.shp_regprism_dirxtr(n_sides, radius, length, fc_normal=FreeCAD.Vector,
                           fc_verx1=FreeCAD.Vector, centered=0, xtr_top=0, xtr_bot=0,
                           pos=FreeCAD.Vector)
```

Similar to shp\_regprism\_xtr, but in any direction makes a shape of a face of a regular polygon. Includes the possibility to add extra length on top and bottom. On top is easy, but at the bottom, the reference will be no counting that extra lenght added. This is useful to make boolean difference

#### Parameters

- **n\_sides** (*int*) – Number of sides of the polygon
- **radius** (*float*) – Circumradius of the polygon
- **length** (*float*) – Length of the polygon
- **fc\_normal** (*FreeCAD.Vector*) – Direction of the normal
- **fc\_verx1** (*FreeCAD.Vector*) – Direction of the first vertex
- **centered** (*int*) – 1 if the extrusion is centered on pos (symmetrical)
- **xtr\_top** (*float*) – Add an extra lenght on top. If 0, nothing added
- **xtr\_bot** (*float*) – Add an extra lenght at the bottom. If 0, nothing added
- **pos** (*FreeCAD.Vector*) – Position of the center. Default (0,0,0)

**Returns** FreeCAD Shape of a regular prism

**Return type** Shape

```
fcfun.shp_regprism_xtr(n_sides, radius, length, n_axis='z', v_axis='x', centered=0, xtr_top=0,
                       xtr_bot=0, edge_rot=0, pos=FreeCAD.Vector)
```

makes a shape of a face of a regular polygon. Includes the possibility to add extra length on top and bottom. On top is easy, but at the bottom, the reference will be no counting that extra lenght added. This is useful to make boolean difference

#### Parameters

- **n\_sides** (*int*) – Number of sides of the polygon
- **radius** (*float*) – Circumradius of the polygon
- **length** (*float*) – Length of the polygon
- **n\_axis** (*str*) – Axis of the normal: 'x', '-x', 'y', '-y', 'z', '-z'
- **v\_axis** (*str*) – Perpendicular to n\_axis, pointing to the first vertex, unless, x\_angle is != 0. the vertex will be rotated x\_angle degrees for v\_axis
- **centered** (*int*) – 1 if the extrusion is centered on pos (symmetrical)
- **xtr\_top** (*float*) – Add an extra lenght on top. If 0, nothing added
- **xtr\_bot** (*float*) – Add an extra lenght at the bottom. If 0, nothing added
- **x\_angle** (*float*) – If zero, the first vertex will be on axis v\_axis if x\_angle != 0, it will rotated some angle

- **pos** (*FreeCAD.Vector*) – Position of the center. Default (0,0,0)

**Returns** FreeCAD Shape of a regular prism

**Return type** Shape

`fcfun.shp_rndrect_face` (*x*, *y*, *r=0.5*, *pos\_z=0*)

Same as `shpRndRectWire`

**Parameters**

- **x** (*float*) – Dimension of the base, on the X axis
- **y** (*float*) – Dimension of the height, on the Y axis
- **r** (*float*) – Radius of the rounded edge.
- **zpos** (*float*) – Position on the Z axis

**Returns** FreeCAD Face of a rounded edges rectangle

**Return type** Shape Face

`fcfun.shp_stadium_dir` (*length*, *radius*, *height*, *fc\_axis\_h=FreeCAD.Vector*,  
*fc\_axis\_l=FreeCAD.Vector*, *fc\_axis\_s=FreeCAD.Vector*, *ref\_l=1*, *ref\_s=1*,  
*ref\_h=1*, *xtr\_h=0*, *xtr\_nh=0*, *pos=FreeCAD.Vector*)

Makes a stadium shape in any direction

```
fc_axis_s
:
:_____ ref_l = 2, ref_s = 1
/         \
3 2   1   ) -----> fc_axis_l
5\_____4___/

fc_axis_h
_:_____
|         | :
|         | :
|   1     | ref_h=1 + h
|         | :
|_____2___| .....> fc_axis_l .....: ref_h=2
```

**Parameters**

- **length** (*float*) – Length of the parallels (distance between semicircle centers)
- **height** (*float*) – Height the stadium
- **fc\_axis\_s** (*FreeCAD.Vector*) – Direction on the short axis, not necessary if `ref_s == 1` it will be the perpendicular of the other 2 vectors
- **fc\_axis\_h** (*FreeCAD.Vector*) – Vector on the height direction
- **ref\_l** (*int*) – Reference (zero) of the `fc_axis_l`
  - 1: reference on the center (makes `axis_s` symmetrical)
  - 2: reference at one of the semicircle centers (point 2) the other circle center will be on the direction of `fc_axis_l`
  - 3: reference at the end (point 3) the other end will be on the direction of `fc_axis_l`
- **ref\_s** (*int*) – Reference (zero) of the `fc_axis_s`

- 1: reference at the center (makes axis\_l symmetrical): p 1,2,3
- 2: reference at the parallels lines: p: 4, 5 the other parallel will be on the direction of fc\_axis\_s
- **ref\_h**(*int*) –
  - 1: reference is at the center of the height
  - 2: reference is at the bottom
- **xtr\_h**(*float*) – If >0 it will be that extra height on the direction of fc\_axis\_h
- **xtr\_nh**(*float*) – If >0 it will be that extra height on the opposite direction of fc\_axis\_h
- **pos**(*FreeCAD.Vector*) – Placement

**Returns** FreeCAD Shape of a stadium

**Return type** Shape

`fcfun.shp_stadium_face(l, r, axis_rect='x', pos_z=0)`

Same as shp\_stadium\_wire, but returns a face

#### Parameters

- **l**(*float*) – Length of the parallels (from center to center)
- **r**(*float*) – Radius of the semicircles
- **axis\_rect**(*str*) – 'x' the parallels are on axis X (as in the drawing) 'y' the parallels are on axis Y
- **pos\_z**(*float*) – Position on the Z axis

**Returns** FreeCAD Face of a stadium

**Return type** Shape Face

`fcfun.shp_stadium_wire(l, r, axis_rect='x', pos_z=0)`

Creates a wire (shape), that is a rectangle with semicircles at a pair of opposite sides. Also called discorectangle it will be centered on XY



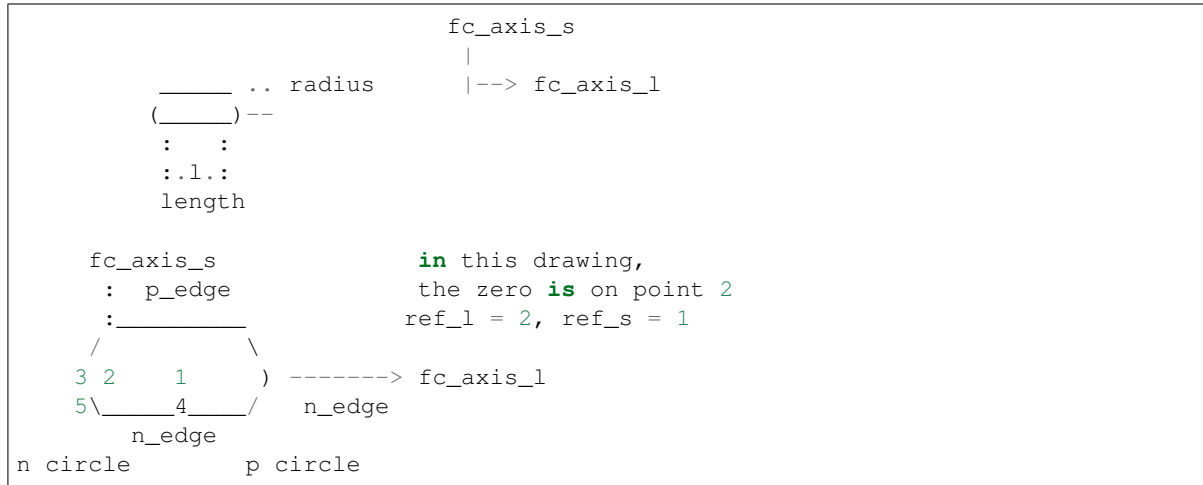
#### Parameters

- **l**(*float*) – Length of the parallels (from center to center)
- **r**(*float*) – Radius of the semicircles
- **axis\_rect**(*str*) – 'x' the parallels are on axis X (as in the drawing) 'y' the parallels are on axis Y
- **pos\_z**(*float*) – Position on the Z axis

**Returns** FreeCAD Wire of a stadium

**Return type** Shape Wire

`fcfun.shp_stadium_wire_dir` (*length*, *radius*, *fc\_axis\_l=FreeCAD.Vector*,  
*fc\_axis\_s=FreeCAD.Vector*, *ref\_l=1*, *ref\_s=1*, *pos=FreeCAD.Vector*)  
 Same as `shp_stadium_wire` but in any direction Also called discorectangle



### Parameters

- **length** (*float*) – Length of the parallels (distance between semicircle centers)
- **radius** (*float*) – Radius of the semicircles
- **fc\_axis\_l** (*FreeCAD.Vector*) – Vector on the direction of the parallels
- **fc\_axis\_s** (*FreeCAD.Vector*) – Vector on the direction perpendicular to the parallels
- **ref\_l** (*int*) – Reference (zero) of the `fc_axis_l`
  - 1 reference on the center (makes `axis_s` symmetrical)
  - 2 reference at one of the semicircle centers (point 2) the other circle center will be on the direction of `fc_axis_l`
  - 3 reference at the end (point 3) the other end will be on the direction of `fc_axis_l`
- **ref\_s** (*int*) – Reference (zero) of the `fc_axis_s`
  - 1 reference at the center (makes `axis_l` symmetrical): p 1,2,3
  - 2 reference at the parallels lines: p: 4, 5 the other parallel will be on the direction of `fc_axis_s`
- **pos** (*FreeCAD.Vector*) – FreeCAD vector of the position of the reference

**Returns** FreeCAD Wire of a stadium

**Return type** Shape Wire

`fcfun.vecname_parallel` (*vec1*, *vec2*)

Given to vectors by name 'x', '-x', ... indicates if they are parallel or not

`fcfun.wire_beltclamp` (*d*, *w*, *corner\_r*, *conn\_d*, *conn\_sep*, *xtr\_conn\_d=0*, *closed=0*,  
*axis\_d=FreeCAD.Vector*, *axis\_w=FreeCAD.Vector*, *pos\_d=0*, *pos\_w=0*,  
*pos=FreeCAD.Vector*)

Creates a wire following 2 pulleys and ending in a belt clamp But it is a wire in FreeCAD, has no volumen

```

axis_w
:
:
pulley1                pulley2

-----
( )                    ( )-----> axis_d
-----
      clamp1          clamp2

1 0      2 3    45 67    8 9      10 11    pos_d
:          :          :          :
:          :          :          :
:          :.....:
:          +
:          clamp_sep
:          :
:          :
:.....:
:          +
:          pull_sep_d

pos_w points:

axis_w
:
:                pull2
:      clamp1          clamp2
2_
3-
( 0 ) - - - - - ( 1 ) - - pull_sep_w (positive)
5-
6 - - - - - + clamp_pull1_w (neg)
4- 7 - - - - - < )      ( >      + clamp_w
8 - - - - -
:

axis_w
:
:                pull2
:      clamp1          clamp2
-
-                -
( ) - - - - - ( ) - - pull_sep_w (positive)
-
- - - - - + clamp_pull1_w (neg)
-      < )      ( >      + clamp_w
- - - - -
:
:      :      :      :      :
:      :      : cyl_r      :      :
:      :      :.....:      :
:      :      +      :      +
:      :      clamp_cyl_sep      :      +
:      :      :      :      clamp_pull2_d
:      :      :.....:
:      :      :      +
:      :      :.....: clamp_d
:      :      +
:      :      clamp_sep
:      :.....:
:      : +

```

(continues on next page)

(continued from previous page)

```

:      : clamp_d
:      :
:.....:
+
clamp_pull1_d

```

### Parameters

- **pull1\_d**(*float*) – Diameter of pulley 1
- **pull2\_d**(*float*) – Diameter of pulley 2
- **pull\_sep\_d**(*float*) – Separation between the 2 pulleys centers along axis\_d if positive, pulley 2 is further away in the direction of axis\_d if negative, pulley 2 is further away opposite to the direction of axis\_d
- **pull\_sep\_w**(*float*) – Separation between the 2 pulleys centers along axis\_w if positive, pulley 2 is further away in the direction of axis\_w if negative, pulley 2 is further away opposite to the direction of axis\_w
- **clamp\_pull1\_d**(*float*) – Separation between the clamp (side closer to the center) and the center of the pulley1
- **clamp\_pull1\_w**(*float*) – Separation between the center of the clamp and the center of the pulley1 if positive, the clamp is further away in the direction of axis\_w if negative, the clamp is further away opposite to the direction of axis\_w
- **clamp\_d**(*float*) – Length of the clamp (same for each clamp)
- **clamp\_w**(*float*) – Width of inner space (same for each clamp)
- **clamp\_sep**(*float*) – Separation between clamps, the closest ends
- **clamp\_cyl\_sep**(*float*) – Separation between clamp and the center of the cylinder (or the center) of the larger cylinder (when is a belt shape)
- **cyl\_r**(*float*) – Radius of the cylinder for the belt, if it is not a cylinder but a shape of 2 cylinders: < ), then the radius of the larger one
- **axis\_d**(*FreeCAD.Vector*) – Coordinate System Vector along the depth
- **axis\_w**(*FreeCAD.Vector*) – Coordinate System Vector along the width
- **pos\_d**(*int*) – Location of pos along the axis\_d, see drawing
  - 0: center of the pulley 1
  - 1: end of pulley 1
  - 2: end of clamp 1, closest end to pulley 1
  - 3: other end of clamp 1, closest to cylinder
  - 4: center of cylinder (or shape < ) 1
  - 5: external radius of cylinder 1
  - 6: external radius of cylinder 2
  - 7: center of cylinder (or shape > 2
  - 8: end of clamp 2, closest to cylinder
  - 9: other end of clamp 2, closest end to pulley 2



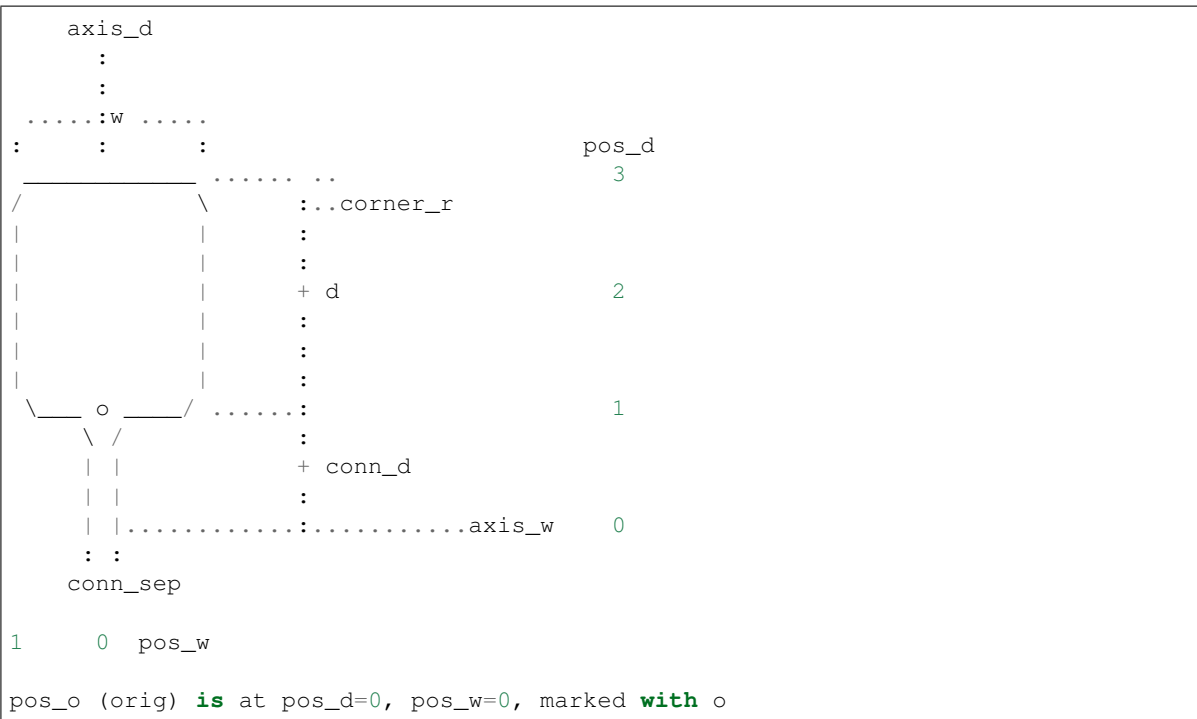
- 10: center of pulley 2
- 11: end of pulley 2
- **pos\_w** (*int*) – Location of pos along the axis\_w, see drawing
  - 0: center of pulley 1
  - 1: center of pulley 2
  - 2: end (radius) of pulley 1 along axis\_w
  - 3: end (radius) of pulley 2 along axis\_w
  - 4: other end (radius) of pulley 1 opposite to axis\_w
  - 5: other end (radius) of pulley 2 opposite to axis\_w
  - 6: clamp space, closest to the pulley
  - 7: center of clamp space
  - 8: clamp space, far away from the pulley
- **pos** (*FreeCAD.Vector*) – Position of the reference

**Returns** FreeCAD Wire of a belt clamped

**Return type** Shape Wire

`fcfun.wire_cableturn` (*d*, *w*, *corner\_r*, *conn\_d*, *conn\_sep*, *xtr\_conn\_d*=0, *closed*=0,  
*axis\_d*=FreeCAD.Vector, *axis\_w*=FreeCAD.Vector, *pos\_d*=0, *pos\_w*=0,  
*pos*=FreeCAD.Vector)

Creates a electrical wire turn, in any direction But it is a wire in FreeCAD, has no volumen



### Parameters

- **d** (*float*) – Depth/length of the turn

- **w** (*float*) – Width of the turn
- **corner\_r** (*float*) – Radius of the corners
- **conn\_d** (*float*) – Depth/length of the connector part
  - 0: there is no connecting wire
- **xtr\_conn\_d** (*float*) – If conn\_d > 0, there can be an extra length of connector to make unions, it will not be counted as pos\_d = 0. It will not work well if it is closed
- **conn\_sep** (*float*) – Separation of the connectors
- **closed** (*boolean*) –
  - 0: the ends are not closed
  - 1: the ends are closed
- **axis\_d** (*FreeCAD.Vector*) – Coordinate System Vector along the depth
- **axis\_w** (*FreeCAD.Vector*) – Coordinate System Vector along the width
- **pos\_d** (*int*) – Location of pos along the axis\_d (0,1,2,3), see drawing
  - 0: reference at the beginning of the connector
  - 1: reference at the beginning of the turn, at the side of the connector
  - 2: reference at the middle of the turn
  - 3: reference at the end of the turn
- **pos\_w** (*int*) – Location of pos along the axis\_w (0,1), see drawing
  - 0: reference at the center of symmetry
  - 1: reference at the end of the turn
- **pos** (*FreeCAD.Vector*) – Position of the reference

**Returns** FreeCAD Wire of a electrical wire

**Return type** Shape Wire

`fcfun.wire_lgrail(rail_w, rail_h, axis_w=FreeCAD.Vector, axis_h=FreeCAD.Vector, pos_w=0, pos_h=0, pos=FreeCAD.Vector)`

Creates a wire of a linear guide rail, the dent is just rough, to be able to see that it is a profile

```

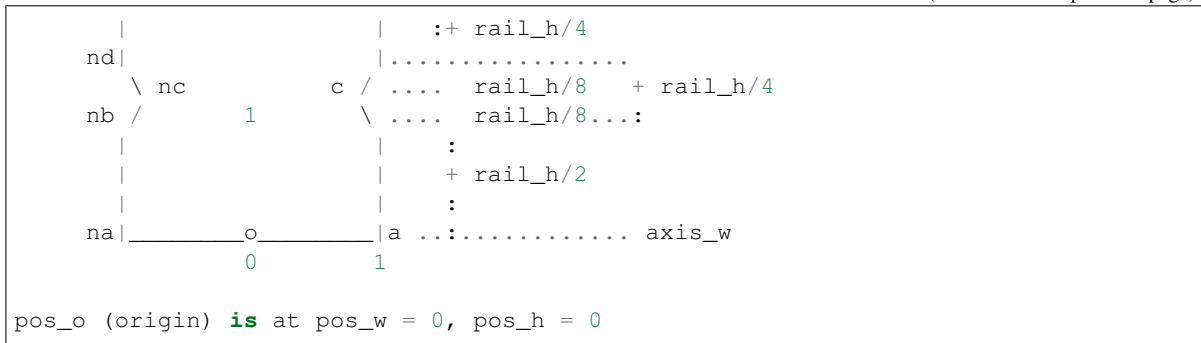
          axis_h
          :
ne  _____2_____ e
|
nd|                    | d
| \ nc                c / A little dent to see that it is a rail
nb / 1                \ b
|
|
|
na|_____o_____a ..... axis_w
      0          1

                      rail_h/8
                      :
ne  _____2_____ : : : :

```

(continues on next page)

(continued from previous page)



### Parameters

- **rail\_w** (*float*) – Width of the rail
- **rail\_h** (*float*) – Height of the rail
- **axis\_w** (*FreeCAD.Vector*) – The axis where the width of the rail is
- **axis\_h** (*FreeCAD.Vector*) – The axis where the height of the rail is
- **pos\_w** (*int*) – Location of pos along axis\_w \* 0 : center of symmetry \* 1 : end of the rail
- **pos\_h** (*int*) – Location of pos along axis\_h \* 0 : bottom \* 1 : middle point (this is kind of non-sense) \* 2 : top point
- **pos** (*FreeCAD.Vector*) – Position, at the point defined by pos\_w and pos\_h

**Returns** Wire of a rail

**Return type** FreeCAD Wire

`fcfun.wire_sim_xy` (*vecList*)

Creates a wire (shape), from a list of points on the positive quadrant of XY the wire is simmetrical to both X and Y



**Parameters** **vecList** (*list*) – List of FreeCAD Vectors, the have to be in order clockwise if the first or the last points are not on the axis, a new point will be created



## INDICES AND TABLES

- `genindex`
- `modindex`
- `search`



## PYTHON MODULE INDEX

### **c**

`comps`, [34](#)

### **f**

`fcfun`, [82](#)

`filter_holder_cls`, [56](#)

### **p**

`parts`, [37](#)





## A

add2CylsHole() (in module fcfun), 83  
 add3CylsHole() (in module fcfun), 84  
 add\_fcobj() (in module fcfun), 88  
 addBolt() (in module fcfun), 84  
 addBoltNut\_hole() (in module fcfun), 85  
 addBox() (in module fcfun), 85  
 addBox\_cen() (in module fcfun), 86  
 addCyl() (in module fcfun), 86  
 addCyl\_pos() (in module fcfun), 87  
 addCylHole() (in module fcfun), 86  
 addCylHolePos() (in module fcfun), 87  
 addCylPos() (in module fcfun), 87  
 aluprof\_vec() (in module fcfun), 88  
 AluProfBracketPerp (class in parts), 40  
 AluProfBracketPerpFlap (class in parts), 41  
 AluProfBracketPerpTwin (class in parts), 43  
 axis\_h (parts.ThinLinBearHouse attribute), 50  
 axis\_h (parts.ThinLinBearHouse1rail attribute), 48  
 axis\_h (parts.ThinLinBearHouseAsim attribute), 54

## B

BeltClamp (class in beltcl), 68  
 beltclamp\_blk\_t (filter\_holder\_cls.ShpFilterHolder attribute), 60  
 beltpost\_l (filter\_holder\_cls.ShpFilterHolder attribute), 60  
 bolt2bolt\_wid (parts.ThinLinBearHouseAsim attribute), 54  
 bolt2cen\_dep (parts.ThinLinBearHouseAsim attribute), 54  
 bolt2cen\_wid\_n (parts.ThinLinBearHouseAsim attribute), 54  
 bolt2cen\_wid\_p (parts.ThinLinBearHouseAsim attribute), 54  
 boltcen\_axis\_dist (parts.ThinLinBearHouse attribute), 51  
 boltcen\_axis\_dist (parts.ThinLinBearHouse1rail attribute), 48  
 boltcen\_perp\_dist (parts.ThinLinBearHouse attribute), 51

boltcen\_perp\_dist (parts.ThinLinBearHouse1rail attribute), 48

## C

calc\_desp\_ncen() (in module fcfun), 89  
 calc\_rot() (in module fcfun), 90  
 calc\_rot\_z() (in module fcfun), 90  
 clamp\_lrbeltpostcen\_dist (filter\_holder\_cls.ShpFilterHolder attribute), 60  
 comps module, 34

## D

d0\_cen (filter\_holder\_cls.ShpFilterHolder attribute), 61  
 d0\_cen (tensioner\_cls.TensionerSet attribute), 64  
 depth (parts.IdlePulleyHolder attribute), 38  
 Din125Washer (class in fc\_cls), 73  
 Din9021Washer (class in fc\_cls), 73  
 Din912Bolt (class in fc\_cls), 74  
 Din934Nut (class in fc\_cls), 72  
 DoubleBeltClamp (class in beltcl), 70

## E

edgeonaxis() (in module fcfun), 91  
 equ() (in module fcfun), 91

## F

f\_breadboard() (in module comp\_optic), 75  
 f\_cagecube() (in module comp\_optic), 76  
 f\_cagecubehalf() (in module comp\_optic), 76  
 fc\_calc\_desp\_ncen() (in module fcfun), 91  
 fc\_calc\_rot() (in module fcfun), 91  
 fc\_isonbase() (in module fcfun), 91  
 fc\_isparal() (in module fcfun), 91  
 fc\_isparal\_nrm() (in module fcfun), 91  
 fc\_isperp() (in module fcfun), 91  
 fcfun module, 82  
 fcoFat (parts.IdlePulleyHolder attribute), 38  
 fillet\_len() (in module fcfun), 92

filletchamfer() (in module fcfun), 92  
 filt\_hole\_d (filter\_holder\_cls.ShpFilterHolder attribute), 60  
 filt\_hole\_h (filter\_holder\_cls.ShpFilterHolder attribute), 60  
 filt\_hole\_w (filter\_holder\_cls.ShpFilterHolder attribute), 60  
 filter\_holder\_cls  
   module, 56  
 fuseshplist() (in module fcfun), 93

## G

get\_bolt\_bearing\_sep() (in module fcfun), 93  
 get\_bolt\_end\_sep() (in module fcfun), 93  
 get\_fc\_perpend1() (in module fcfun), 94  
 get\_fc\_list\_4perp2\_fcvec() (in module fcfun), 94  
 get\_fc\_list\_4perp2\_vecname() (in module fcfun), 95  
 get\_fc\_list\_4perp\_fcvec() (in module fcfun), 95  
 get\_fc\_list\_4perp\_vecname() (in module fcfun), 95  
 get\_fcvecup() (in module fcfun), 95  
 get\_nameofbasevec() (in module fcfun), 95  
 get\_positive\_vecname() (in module fcfun), 95  
 get\_rot() (in module fcfun), 96  
 get\_tangent\_2circles() (in module fcfun), 96  
 get\_tangent\_circle\_pt() (in module fcfun), 97  
 get\_vecname\_perpend1() (in module fcfun), 98  
 get\_vecname\_perpend2() (in module fcfun), 99  
 getfcvecfname() (in module fcfun), 99  
 getvecfname() (in module fcfun), 99

## H

h0\_cen (filter\_holder\_cls.ShpFilterHolder attribute), 61  
 h0\_cen (tensioner\_cls.TensionerSet attribute), 64  
 height (parts.IdlePulleyHolder attribute), 38

## I

IdlePulleyHolder (class in parts), 37

## L

Lb1cPlate (class in comp\_optic), 77  
 Lb2cPlate (class in comp\_optic), 77  
 lcp01m\_plate() (in module comp\_optic), 78  
 lcpb1m\_base() (in module comp\_optic), 79  
 LinBearHouse (class in parts), 52  
 lr\_beltpost\_r (filter\_holder\_cls.ShpFilterHolder attribute), 60

## M

metric (fc\_cls.Din125Washer attribute), 73

metric (fc\_cls.Din9021Washer attribute), 74  
 model\_type (fc\_cls.Din125Washer attribute), 73  
 model\_type (fc\_cls.Din9021Washer attribute), 74  
 module  
   comps, 34  
   fcfun, 82  
   filter\_holder\_cls, 56  
   parts, 37

## N

nl\_bot\_axis (parts.ThinLinBearHouse attribute), 50  
 nl\_bot\_axis (parts.ThinLinBearHouse1rail attribute), 48  
 nl\_perp (parts.ThinLinBearHouse attribute), 50  
 nl\_perp (parts.ThinLinBearHouse1rail attribute), 48  
 nl\_slide\_axis (parts.ThinLinBearHouse attribute), 50  
 nl\_slide\_axis (parts.ThinLinBearHouse1rail attribute), 48  
 nbot\_ax (parts.ThinLinBearHouseAsim attribute), 54  
 NemaMotorHolder (class in parts), 45  
 NemaMotorPulleySet (class in partset), 65  
 nfro\_ax (parts.ThinLinBearHouseAsim attribute), 54  
 nsid\_ax (parts.ThinLinBearHouseAsim attribute), 54  
 NutHole (class in fcfun), 82

## P

PartAluProf (class in comps), 35  
 PartFilterHolder (class in filter\_holder\_cls), 56  
 PartLinGuideBlock (class in comps), 36  
 parts  
   module, 37  
 PrizLed() (in module comp\_optic), 79  
 prnt\_ax (filter\_holder\_cls.ShpFilterHolder attribute), 61  
 prnt\_ax (tensioner\_cls.TensionerSet attribute), 64

## R

regpolygon\_dir\_vecl() (in module fcfun), 99  
 regpolygon\_vecl() (in module fcfun), 99  
 rotateview() (in module fcfun), 99

## S

shp\_2stadium\_dir() (in module fcfun), 100  
 shp\_aluwire\_dir() (in module fcfun), 101  
 shp\_belt\_dir() (in module fcfun), 103  
 shp\_belt\_wire\_dir() (in module fcfun), 104  
 shp\_bolt() (in module fcfun), 104  
 shp\_bolt\_dir() (in module fcfun), 105  
 shp\_boltnut\_dir\_hole() (in module fcfun), 107  
 shp\_box\_dir() (in module fcfun), 108  
 shp\_box\_dir\_xtr() (in module fcfun), 109  
 shp\_box\_rot() (in module fcfun), 111  
 shp\_boxcen() (in module fcfun), 111

shp\_boxcenchmf() (in module fcfun), 111  
shp\_boxcenfill() (in module fcfun), 112  
shp\_boxcenxtr() (in module fcfun), 112  
shp\_boxdir\_fillchmfplane() (in module fcfun),  
113  
shp\_cableturn() (in module fcfun), 116  
shp\_cir\_fillchmf() (in module fcfun), 117  
shp\_cyl() (in module fcfun), 117  
shp\_cyl\_gen() (in module fcfun), 117  
shp\_cylcenxtr() (in module fcfun), 120  
shp\_cylfilletchamfer() (in module fcfun), 120  
shp\_cylhole() (in module fcfun), 120  
shp\_cylhole\_arc() (in module fcfun), 121  
shp\_cylhole\_bolthole() (in module fcfun), 123  
shp\_cylhole\_gen() (in module fcfun), 125  
shp\_cylholedir() (in module fcfun), 127  
shp\_extrud\_face() (in module fcfun), 128  
shp\_extrud\_face\_rot() (in module fcfun), 128  
shp\_face\_lgrail() (in module fcfun), 128  
shp\_face\_rail() (in module fcfun), 129  
shp\_filletchamfer() (in module fcfun), 130  
shp\_filletchamfer\_dir() (in module fcfun), 130  
shp\_filletchamfer\_dirpt() (in module fcfun),  
131  
shp\_filletchamfer\_dirpts() (in module fcfun),  
131  
shp\_filletchamfer\_dirs() (in module fcfun),  
131  
shp\_hollowbelt\_dir() (in module fcfun), 132  
shp\_nuthole() (in module fcfun), 132  
shp\_regpolygon\_dir\_face() (in module fcfun),  
134  
shp\_regpolygon\_face() (in module fcfun), 134  
shp\_regprism() (in module fcfun), 134  
shp\_regprism\_dirxtr() (in module fcfun), 135  
shp\_regprism\_xtr() (in module fcfun), 135  
shp\_rndrect\_face() (in module fcfun), 136  
shp\_stadium\_dir() (in module fcfun), 136  
shp\_stadium\_face() (in module fcfun), 137  
shp\_stadium\_wire() (in module fcfun), 137  
shp\_stadium\_wire\_dir() (in module fcfun), 137  
ShpFilterHolder (class in filter\_holder\_cls), 56  
shpRndRectWire() (in module fcfun), 99  
SimpleEndstopHolder (class in parts), 38  
Sk\_dir (class in comps), 34  
SM1TubelensSm2() (in module comp\_optic), 80

## T

TensionerSet (class in tensioner\_cls), 61  
ThinLinBearHouse (class in parts), 49  
ThinLinBearHouseIrrail (class in parts), 47  
ThinLinBearHouseAsim (class in parts), 52  
ThLed30() (in module comp\_optic), 81  
tot\_d (tensioner\_cls.TensionerSet attribute), 64

tot\_d\_extend (tensioner\_cls.TensionerSet attribute), 64

## V

vecname\_paral() (in module fcfun), 138

## W

w0\_cen (filter\_holder\_cls.ShpFilterHolder attribute),  
61  
w0\_cen (tensioner\_cls.TensionerSet attribute), 64  
width (parts.IdlePulleyHolder attribute), 38  
wire\_beltclamp() (in module fcfun), 138  
wire\_cableturn() (in module fcfun), 141  
wire\_lgrail() (in module fcfun), 142  
wire\_sim\_xy() (in module fcfun), 143