Computational Fabrication

CS 491 and 591
Professor: Leah Buechley
https://handandmachine.org/classes/computational_fabrication/

No time for project demos :(.
If you didn’t present this assignment,
you’ll be first next time.

Late days = 1 day
due Thursday, day late = Friday
weekend days count as days

Some of you have used all of your late days on the first assignment.

questions?

Large Assignment 2

Due Tuesday 9/26 (one week from today)

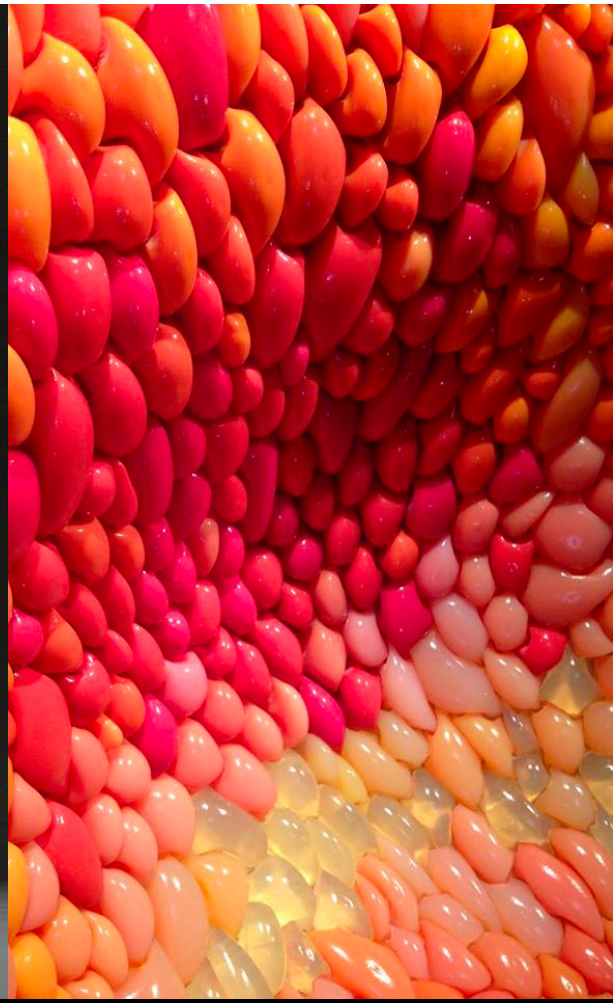
Three parametric 3D printed vessels

Comments and responses

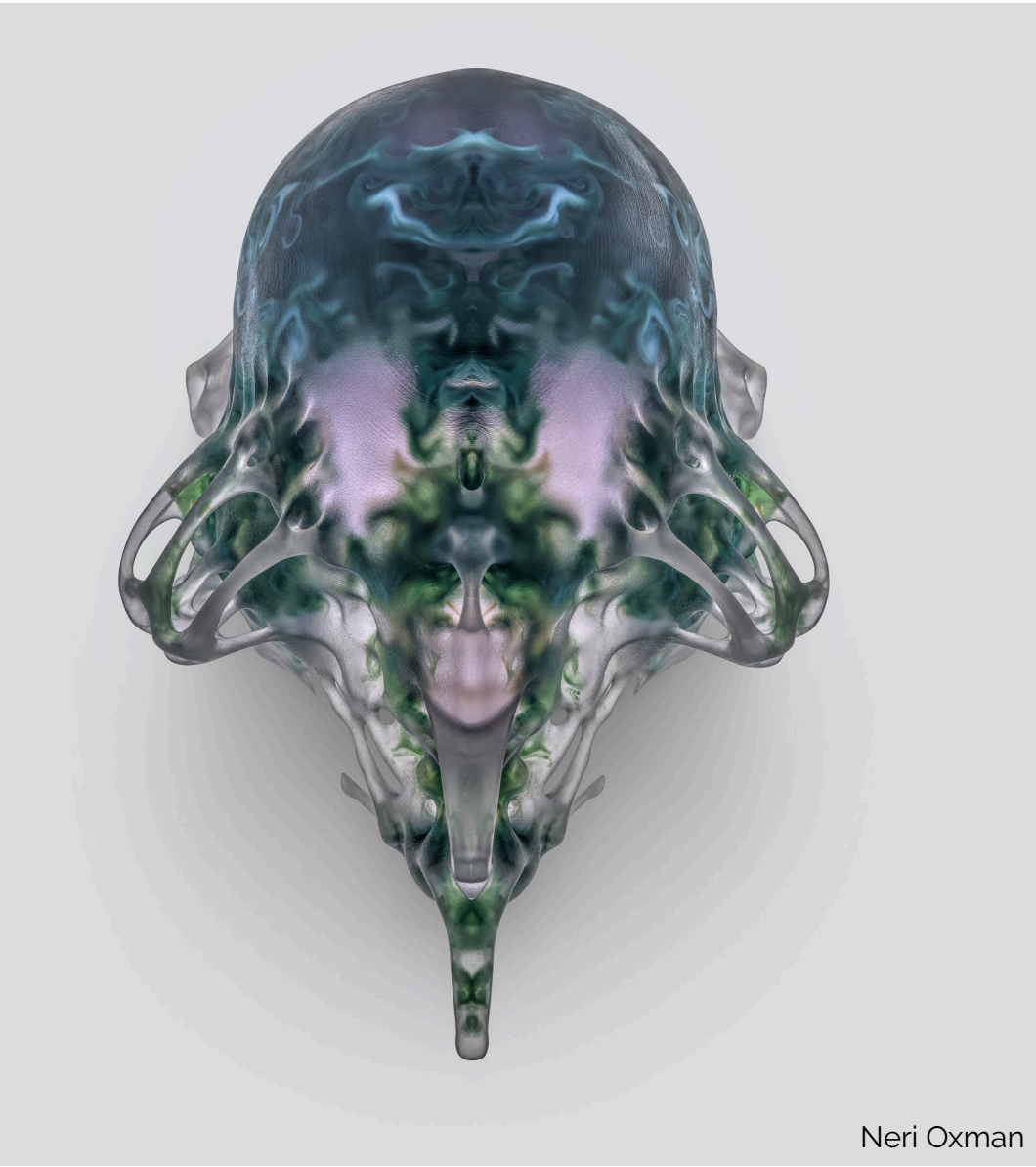
Weekly Designer: Neri Oxman

https://www.moma.org/calendar/exhibitions/5090

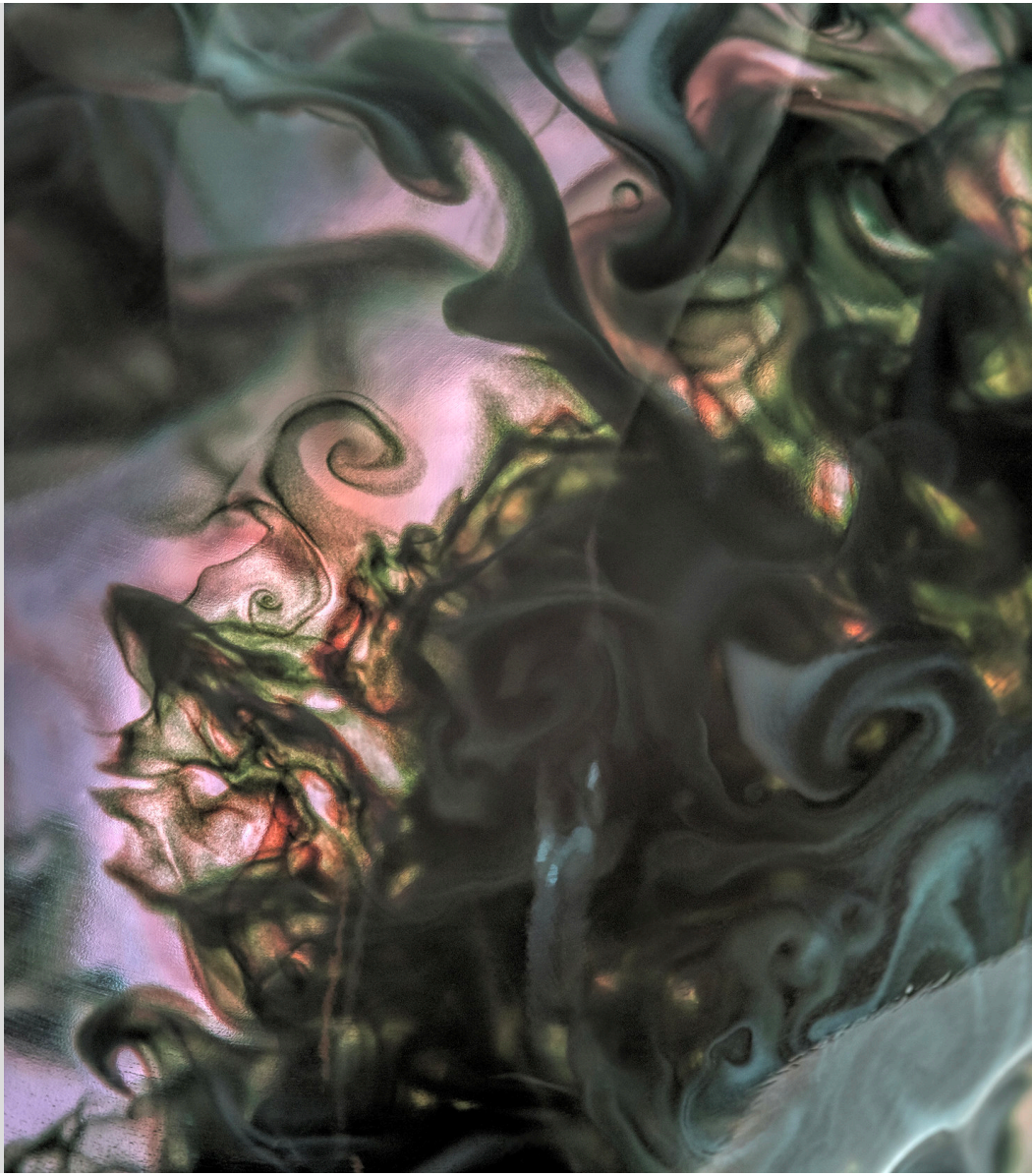
https://oxman.com/projects

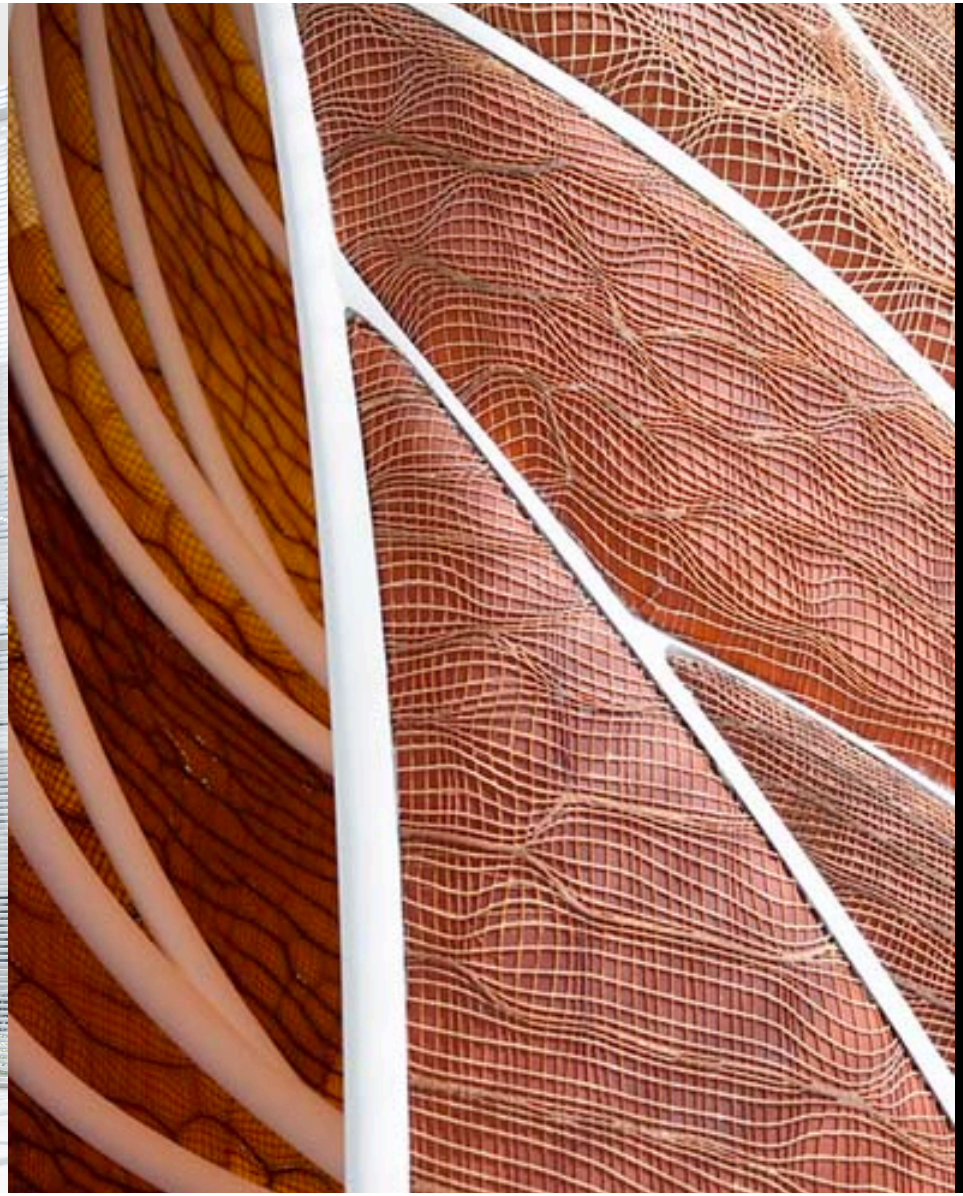


Neri Oxman



Neri Oxman





Neri Oxman

Rhino, Grasshopper, and Python

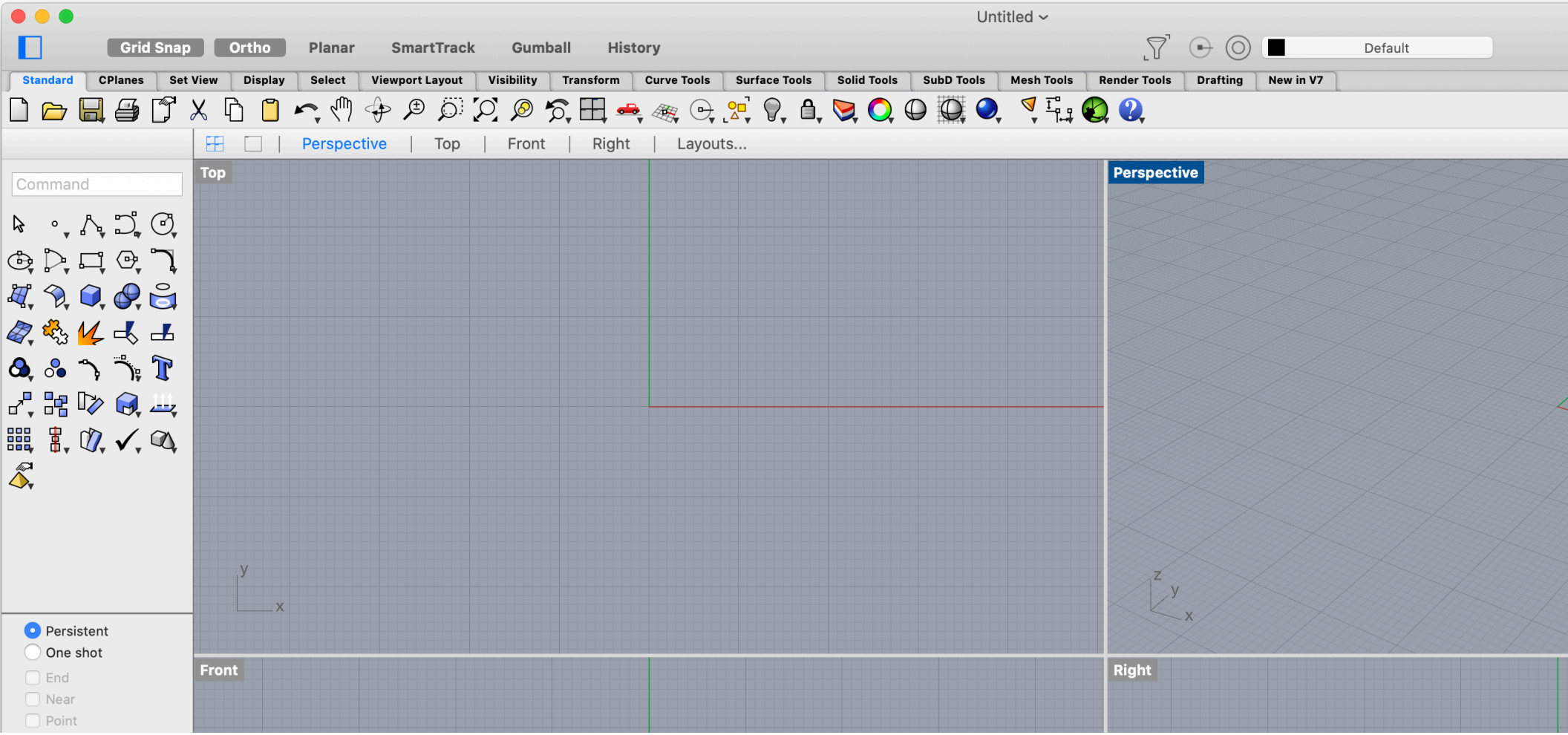
Reminder Tips

Extremely helpful to use a mouse instead of a trackpad! Bring a mouse to class if you can.

Very useful to have two screens,
one for scripting (Grasshopper) and one for visualizing (Rhino).

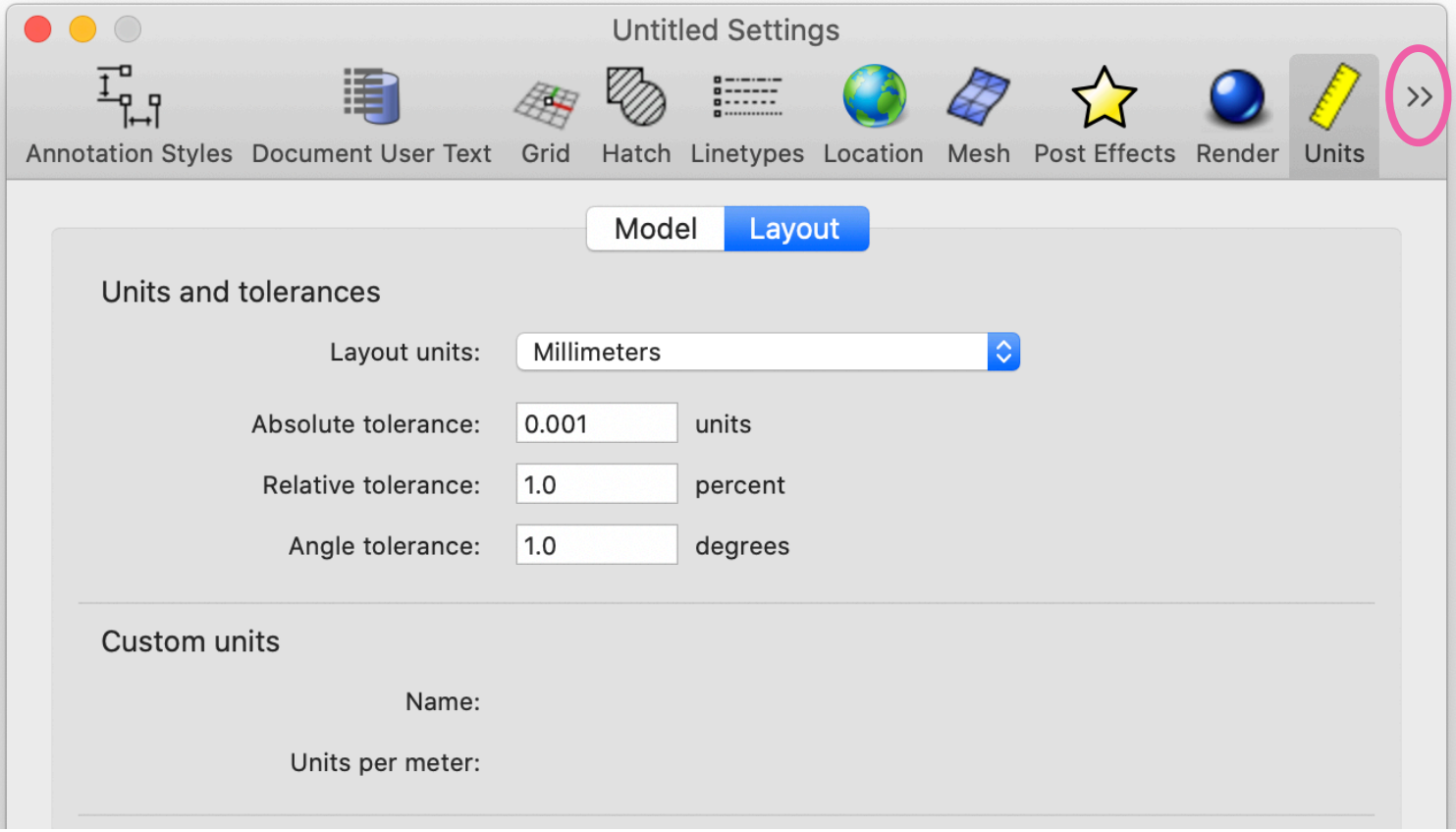
If you don't have two screens, use two side-by side windows,
one for scripting (Grasshopper) and one for visualizing (Rhino).

open up Rhino



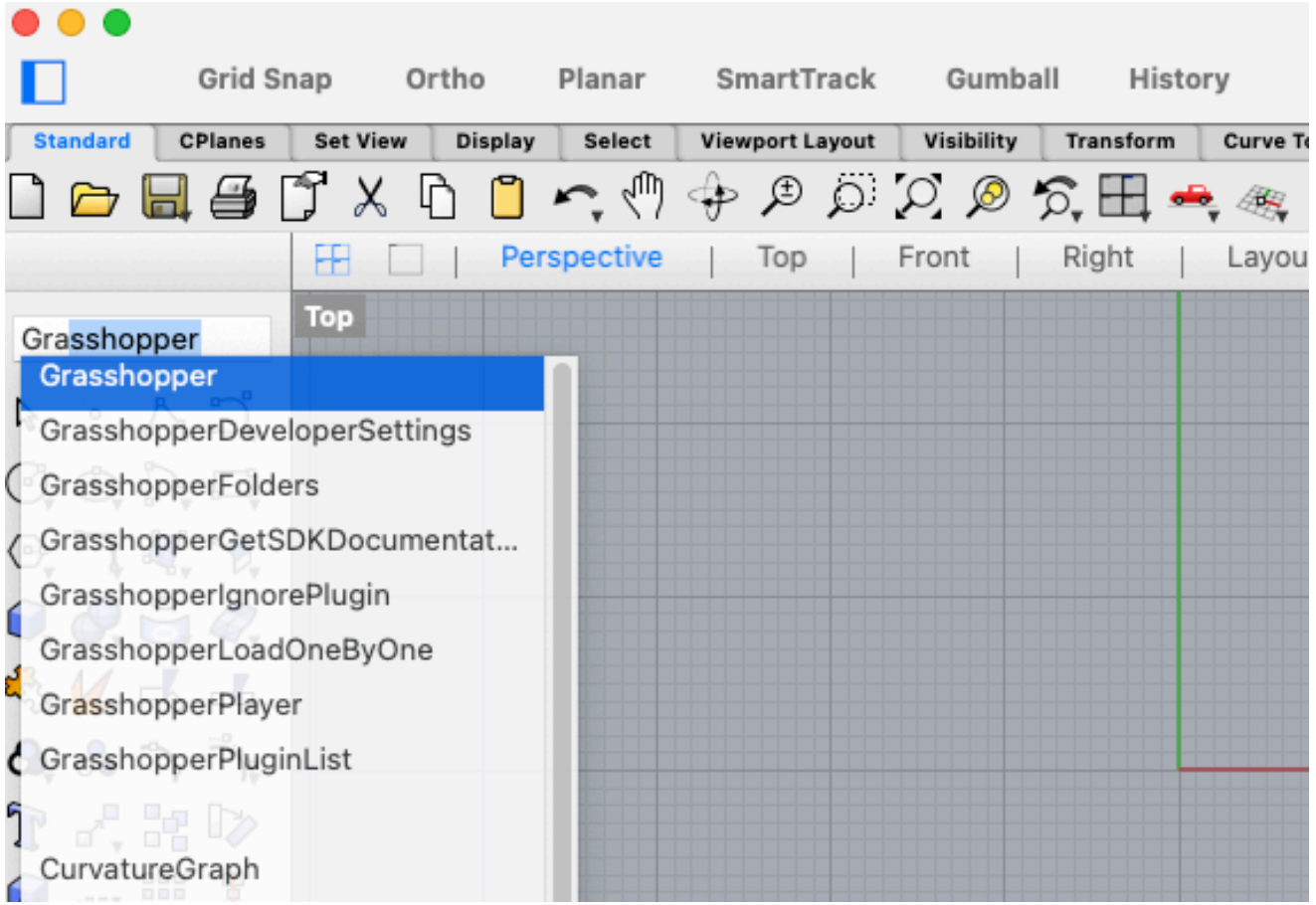
Units and Grid Settings

File—> Settings—>Units

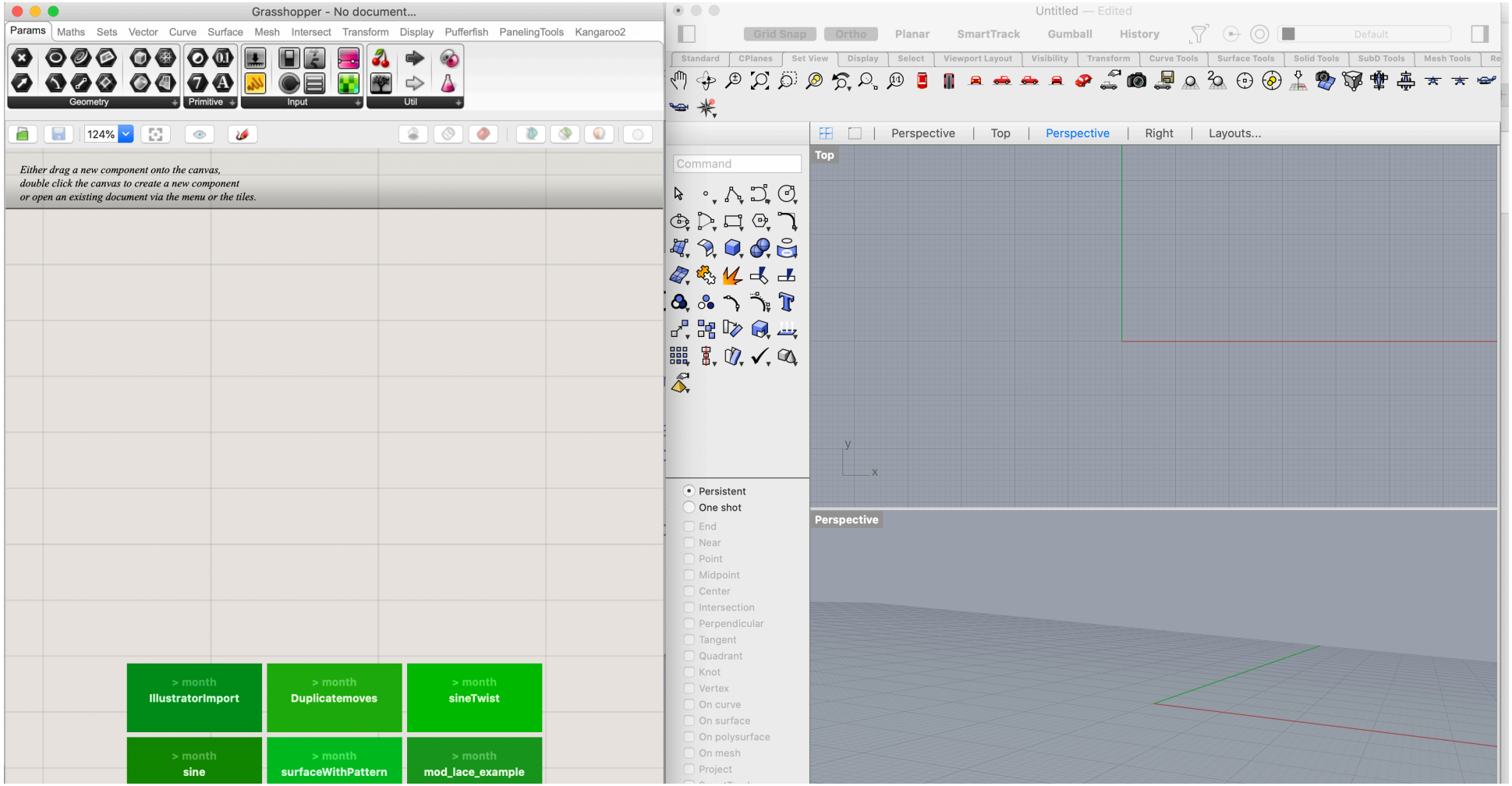


if you don't see units

Open Grasshopper



Set up so you can see both applications



RhinoScript for Python Documentation

Open and bookmark:

https://developer.rhino3d.com/api/RhinoScriptSyntax

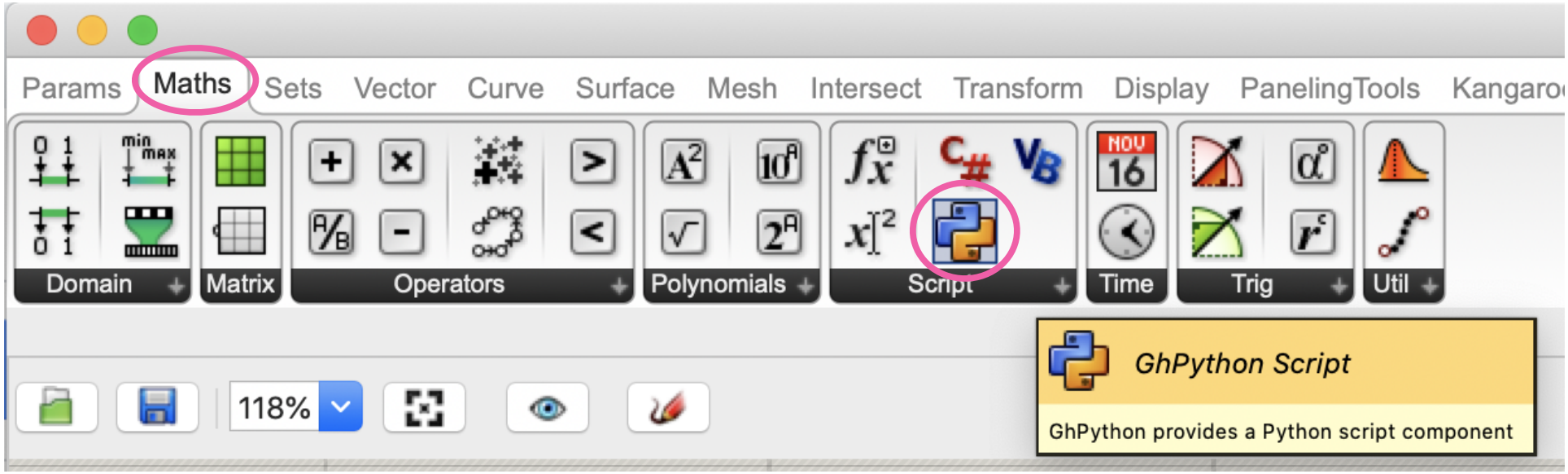
Grasshopper Documentation

Open and bookmark:

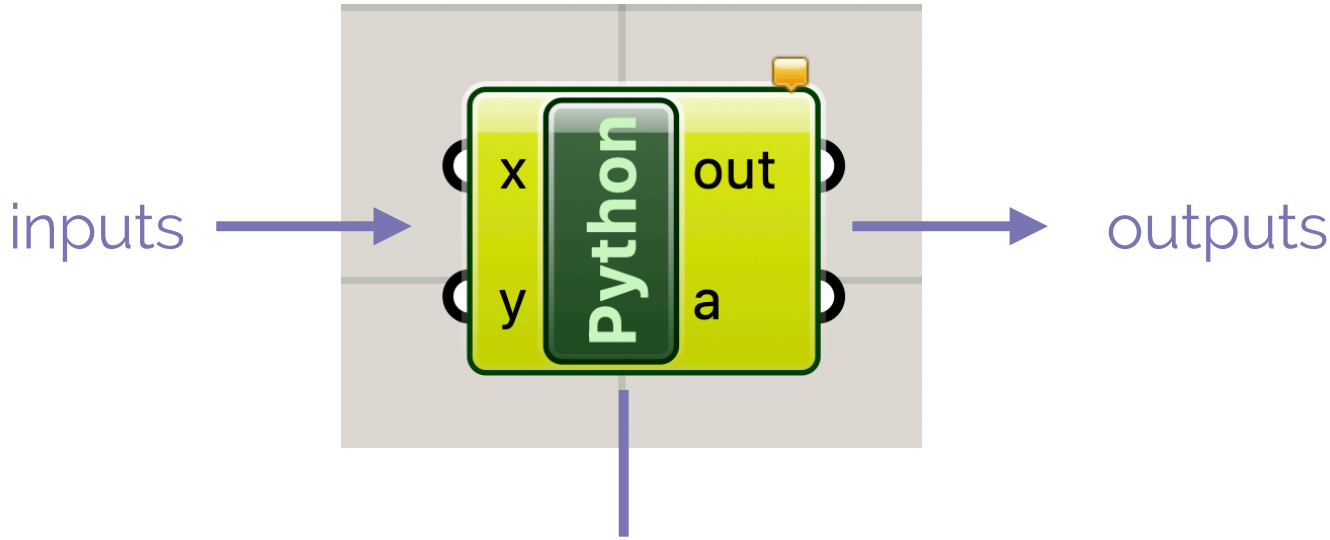
https://grasshopperdocs.com/

Points, Lines, and Surfaces

Python Code Block



Python Code Block



Grasshopper Python Script Editor

```
"""Provides a scripting component.
    Inputs:
        x: The x script variable
        y: The y script variable
    Output:
        a: The a output variable"""

__author__ = "Leah"

import rhinoscriptsyntax as rs

```

inside block: Python code

Points: rs.CreatePoint (x,y,z)

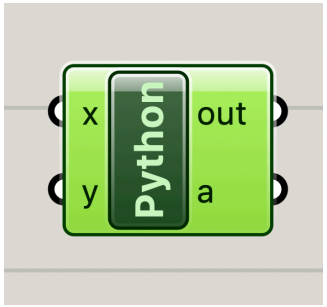
Python

Grasshopper Python Script Editor

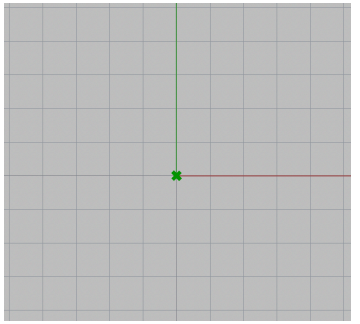
```
1 import rhinoscriptsyntax as rs
2
3 point = rs.CreatePoint(0,0,0)
4 a = point
```

Help Output

Grasshopper



Rhino



CreatePoint Documentation

CreatePoint

```
CreatePoint(point, y=None, z=None)
```

Converts 'point' into a Rhino.Geometry.Point3d if possible. If the provided object is already a point, it value is copied. In case the conversion fails, an error is raised. Alternatively, you can also pass two coordinates singularly for a point on the XY plane, or three for a 3D point.

Parameters:

```
point (Point3d|Vector3d|Point3f|Vector3f|str|guid|[number, number, number])
```

Returns:

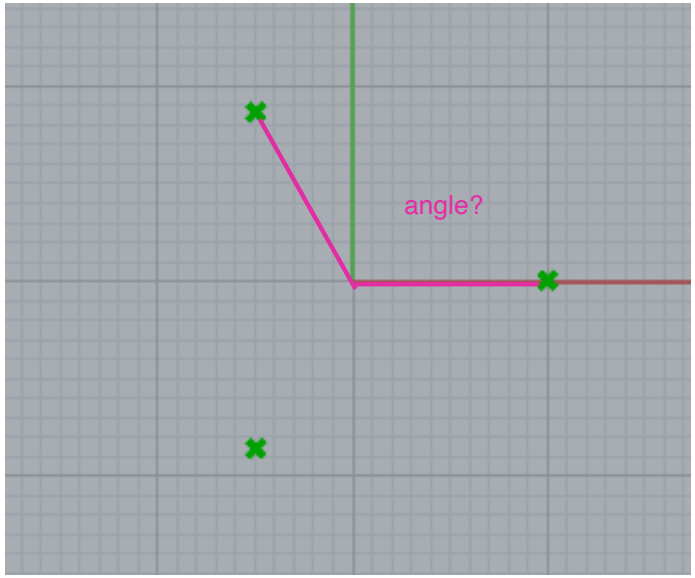
```
point: a Rhino.Geometry.Point3d. This can be seen as an object with three indices:
  [0]  X coordinate
  [1]  Y coordinate
  [2]  Z coordinate.
```

https://developer.rhino3d.com/api/RhinoScriptSyntax/#utility-CreatePoint

Points of a triangle

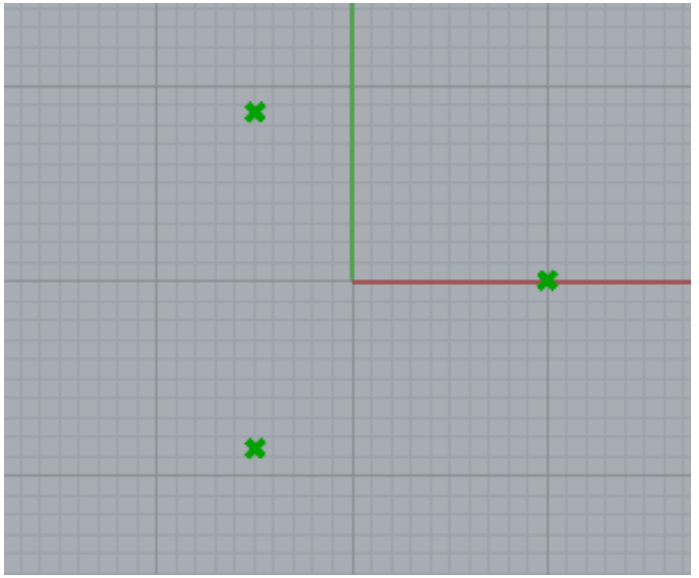
How can we calculate the corner points of a triangle?

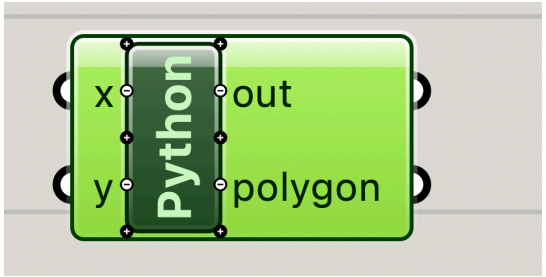
Let's use polar coordinates: a radius and an angle



Points of a triangle

```python
__author__ = "Leah"

import rhinoscriptsyntax as rs
import math

points = []
r = 10

for i in range(0,3):
    angle = 360/3
    x = r*math.cos(math.radians(angle*i))
    y = r*math.sin(math.radians(angle*i))
    points.append(rs.CreatePoint(x,y,0))

polygon = points
```





Draw the triangle using: rs.AddPolyline (list of points)

AddPolyline

```
AddPolyline(points, replace_id=None)
```

Adds a polyline curve to the current model

Parameters:

```
points ([guid|point, guid|point, ...]): list of 3D points. Duplicate, consecutive points will be
         removed. The list must contain at least two points. If the
         list contains less than four points, then the first point and
         last point must be different.
replace_id (guid, optional): If set to the id of an existing object, the object
         will be replaced by this polyline
```

creates a line between a list of points

Returns:

```
guid: id of the new curve object if successful
```

Example:

```
import rhinoscriptsyntax as rs
points = rs.GetPoints(True)
if points: rs.AddPolyline(points)
```

See Also:

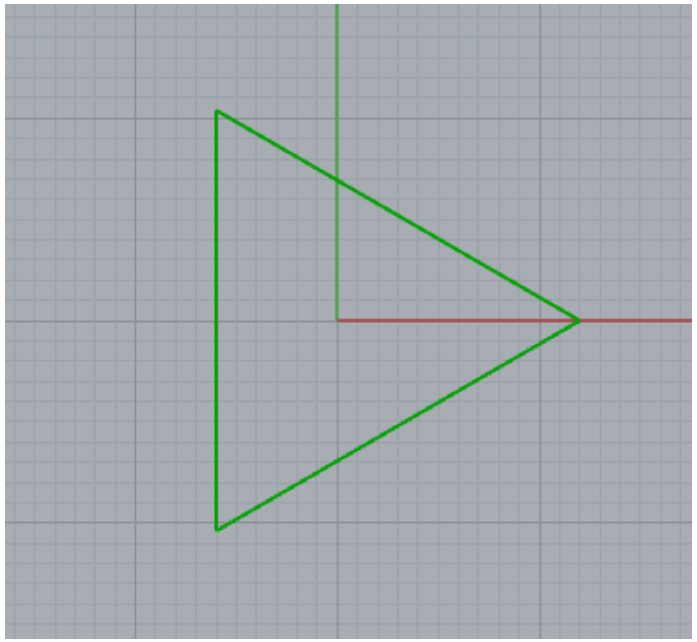
IsPolyline

Draw the triangle using: rs.AddPolyline (list of points)

```
__author__ = "Leah"

import rhinoscriptsyntax as rs
import math

r = 10
points = []

for i in range (0,3):
    angle = i*360/3
    x = r*math.cos(math.radians(angle))
    y = r*math.sin(math.radians(angle))
    points.append(rs.CreatePoint(x,y,0))

polygon = rs.AddPolyline(points)

```

Closed polygon: add one more point

```
__author__ = "Leah"

import rhinoscriptsyntax as rs
import math

r = 10
points = []

for i in range (0,3+1):
    angle = i*360/3
    x = r*math.cos(math.radians(angle))
    y = r*math.sin(math.radians(angle))
    points.append(rs.CreatePoint(x,y,0))

polygon = rs.AddPolyline(points)

```



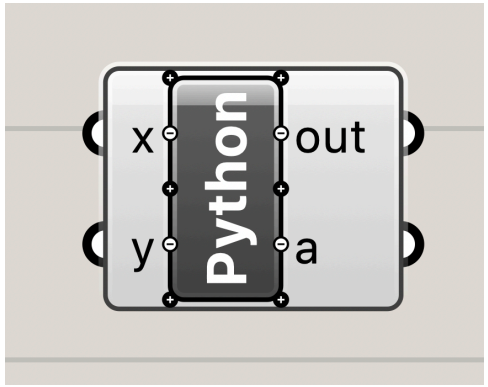
Creating a closed polygon

1. Create points at corners of polygon & add to a list using rs.CreatePoint()

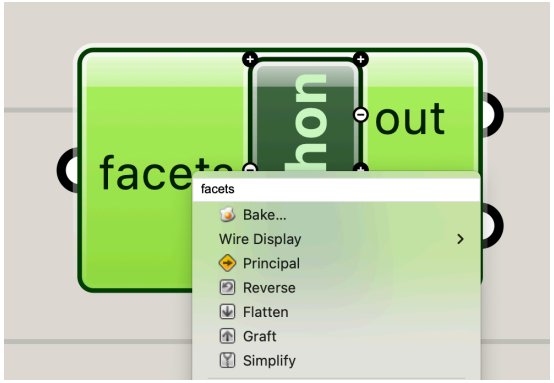
2. Draw lines between points to create a closed line using rs.AddPolyline()

questions?

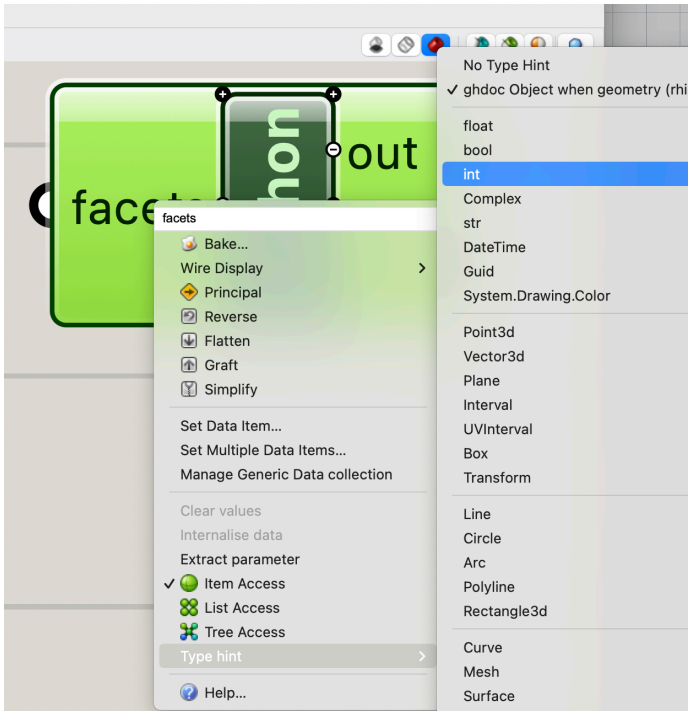
Adding an input to the Python block



zoom in on block to add or remove inputs

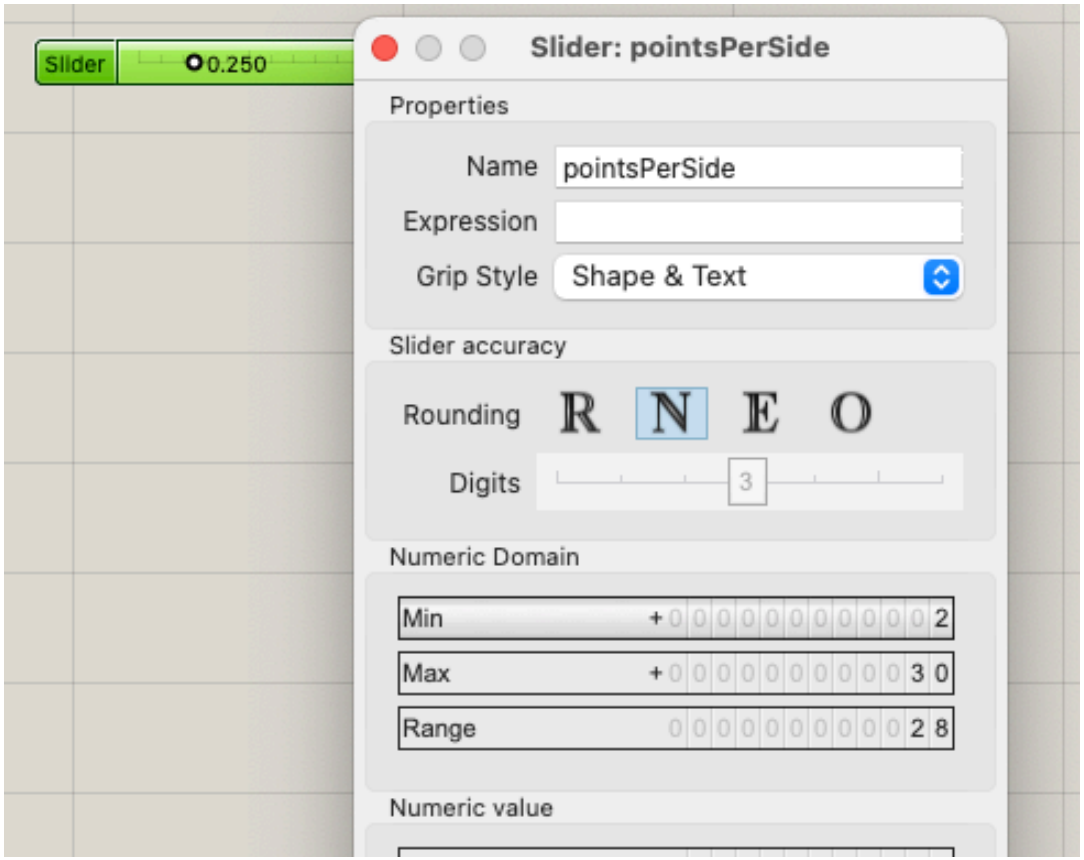


right click on input to rename **facets**

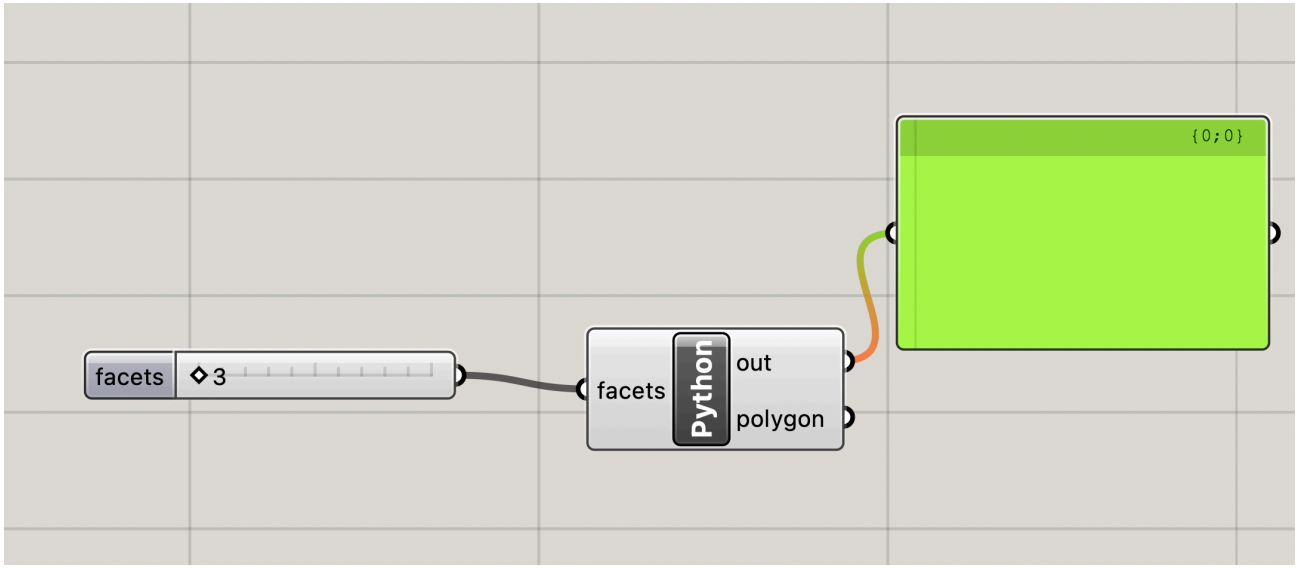


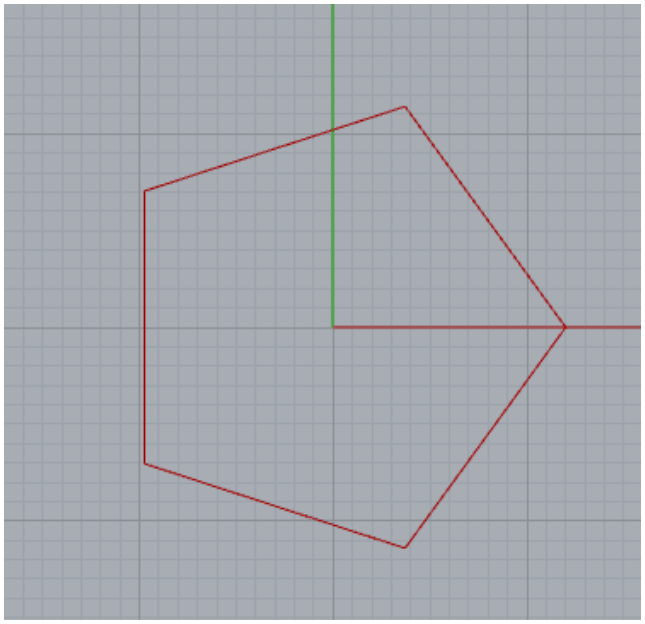
all inputs need Type hints select **int**

Create a Number Slider: Integers from 3-30

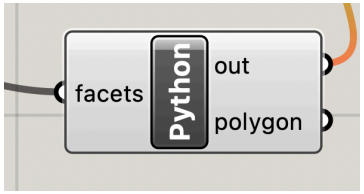


Adding an input: add facets slider

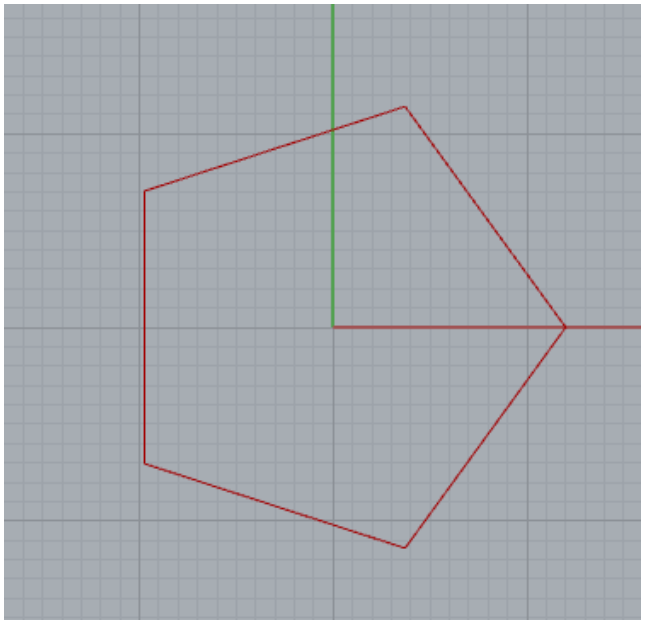




Adding an input: add facets slider

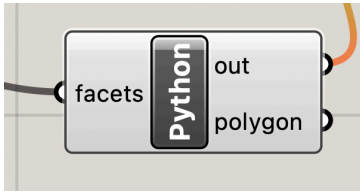



```python
import rhinoscriptsyntax as rs
import math

points = []
r = 20

for i in range (0,facets+1):
    angle = 360/facets
    x = r*math.cos(math.radians(angle*i))
    y = r*math.sin(math.radians(angle*i))
    point = rs.CreatePoint(x,y,0)
    points.append(point)

polygon = rs.AddPolyline(points)
```

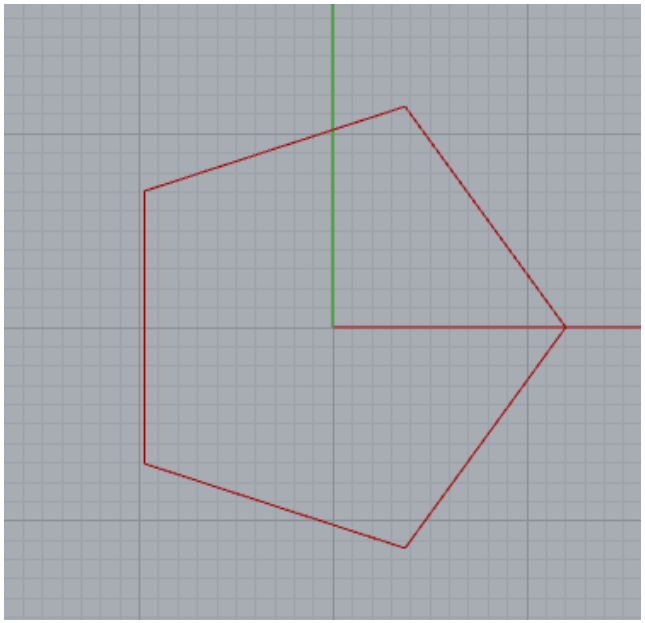


questions?

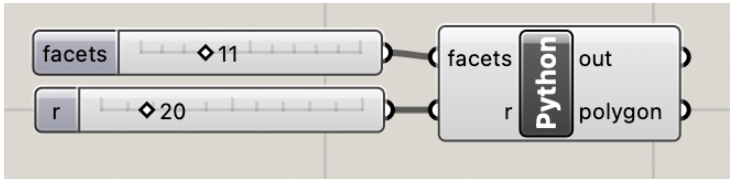
Clean up code: define a function

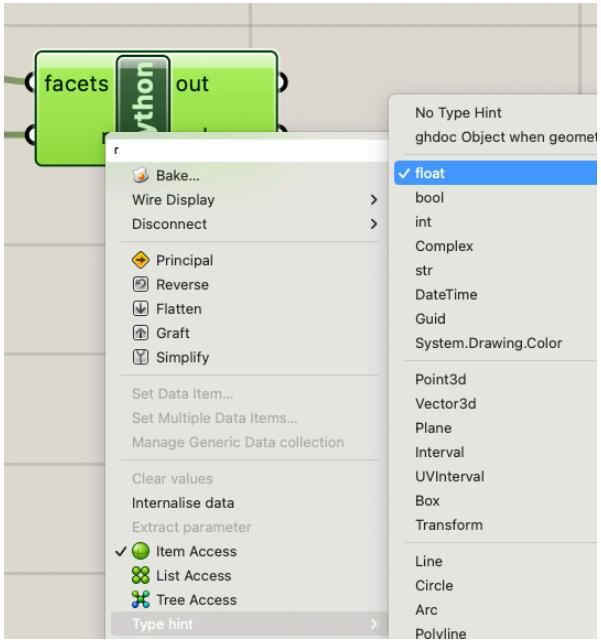



```python
import rhinoscriptsyntax as rs
import math

def polygonPoints(facets, r):
    points = []
    for i in range (0,facets+1):
        angle = 360/facets
        x = r*math.cos(math.radians(angle*i))
        y = r*math.sin(math.radians(angle*i))
        point = rs.CreatePoint(x,y,0)
        points.append(point)
    return points

points = polygonPoints(facets,20)
polygon = rs.AddPolyline(points)
```



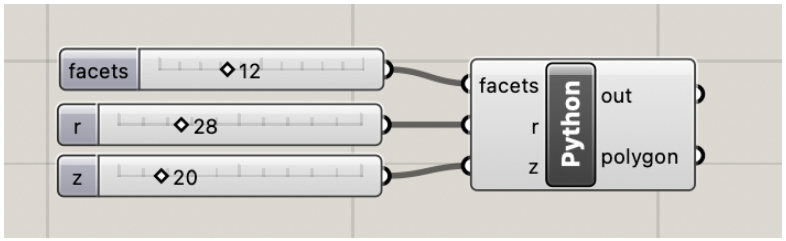
Add an r input slider

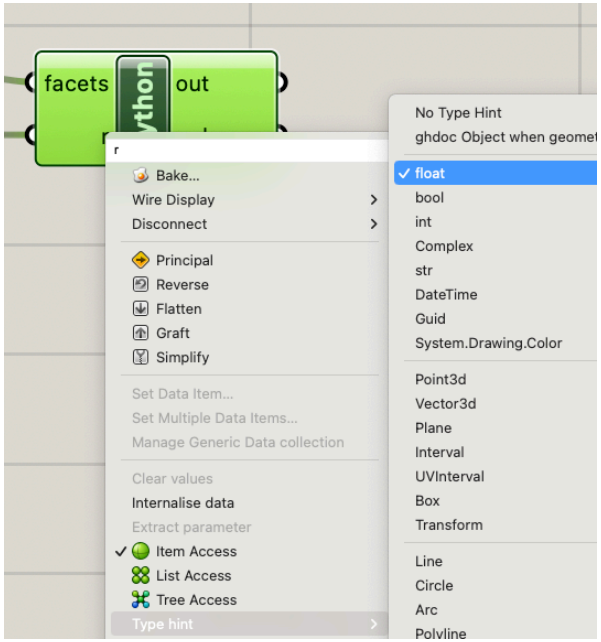




remember Type hint
select **float**

Add a z input slider

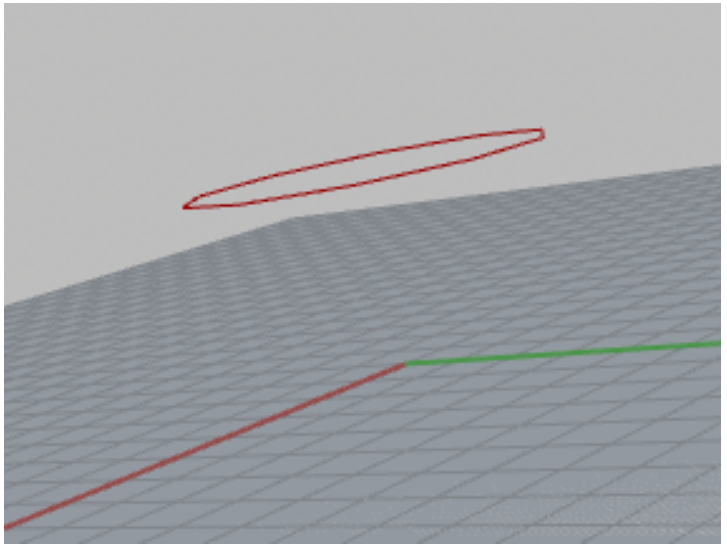




remember Type hint
select **float**

Add a z input slider

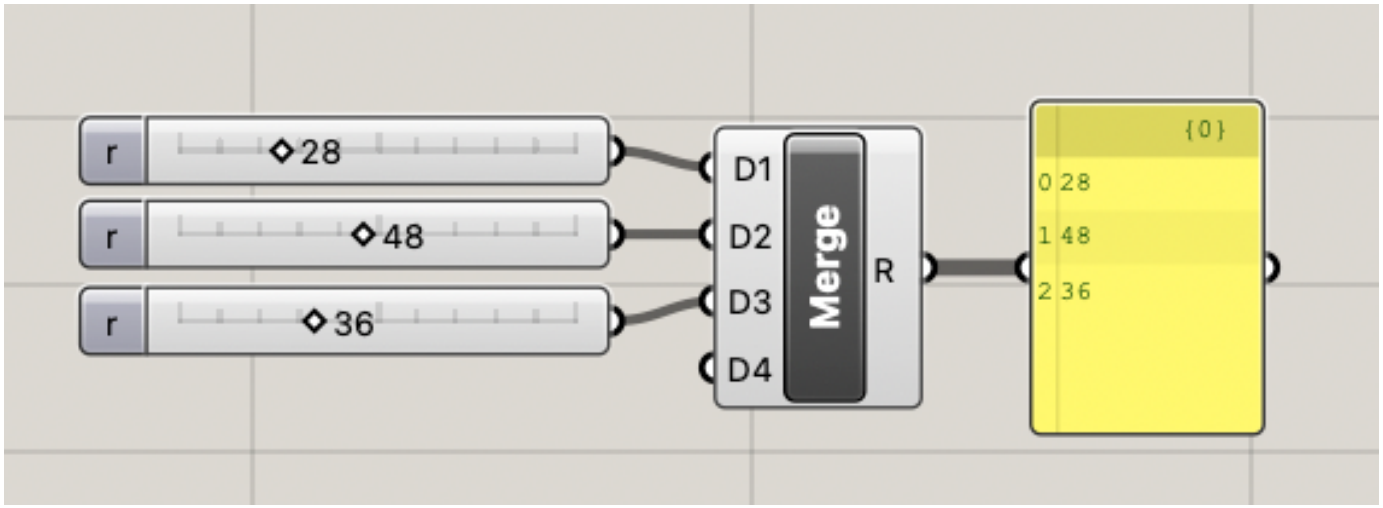
```
1 import rhinoscriptsyntax as rs
2 import math
3
4 def polygonPoints(facets, r, z):
5     points = []
6     for i in range (0,facets+1):
7         angle = 360/facets
8         x = r*math.cos(math.radians(angle*i))
9         y = r*math.sin(math.radians(angle*i))
10        point = rs.CreatePoint(x,y,z)
11        points.append(point)
12    return points
13
14 polygon = rs.AddPolyline(polygonPoints(facets, r, z))
15
```



questions?

Now we’re going to create several polygons with different rs and zs

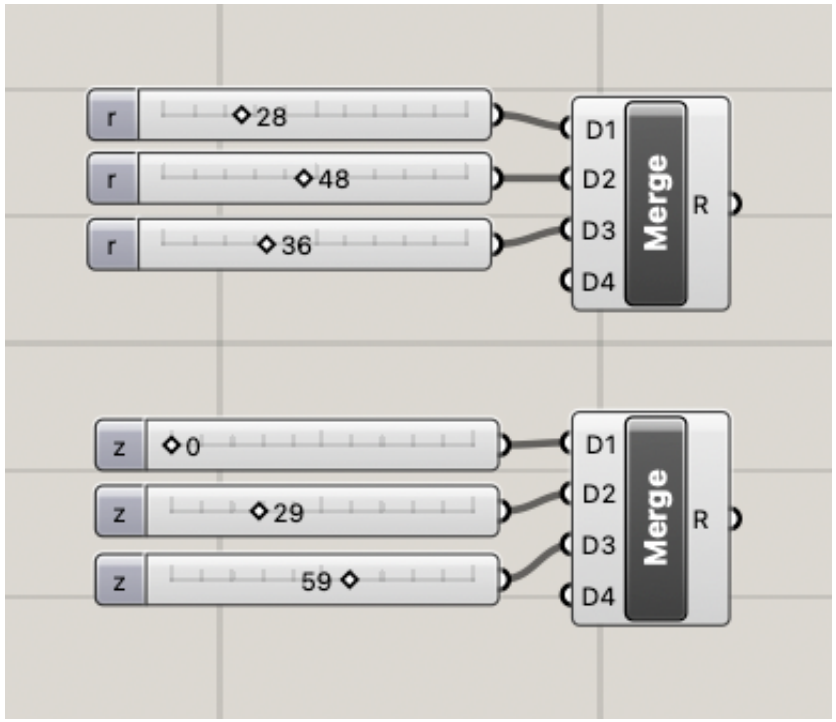
Lists in Grasshopper using Merge

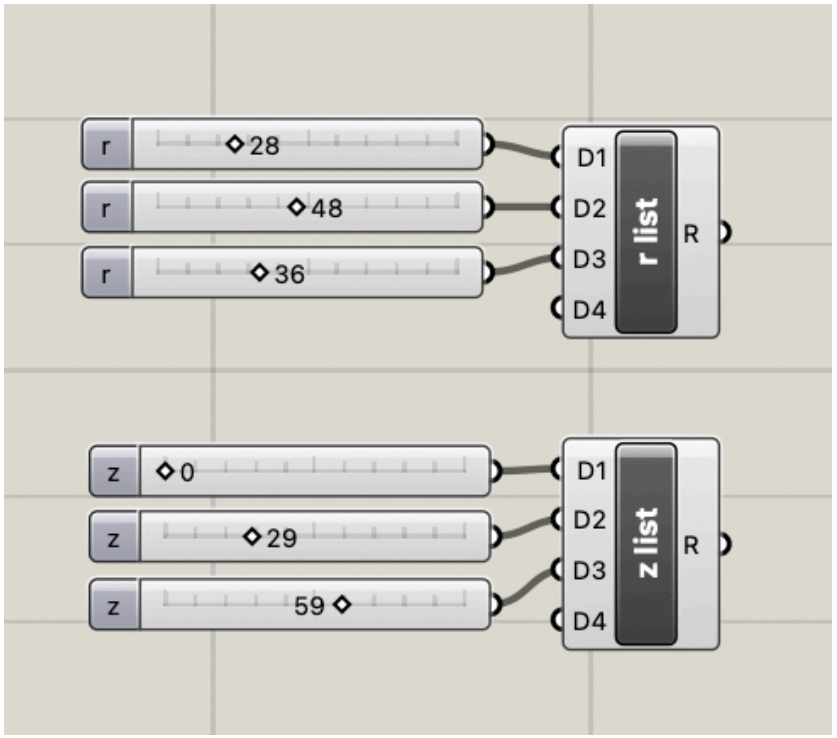


The Merge block is a good way to create lists with specific entries.

https://grasshopperdocs.com/components/grasshoppersets/merge.html

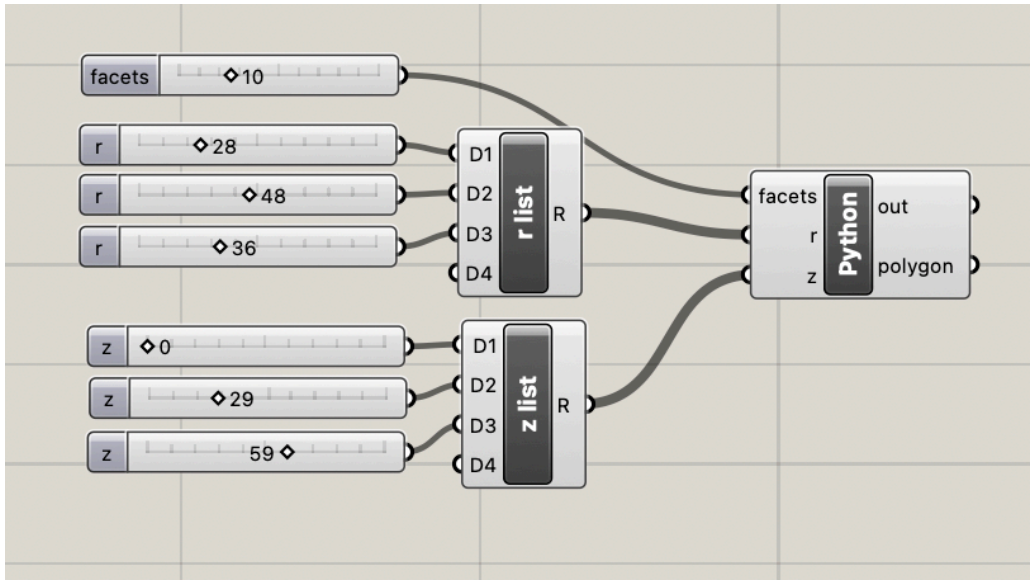
Create Merge blocks for r and z with 3 entries each

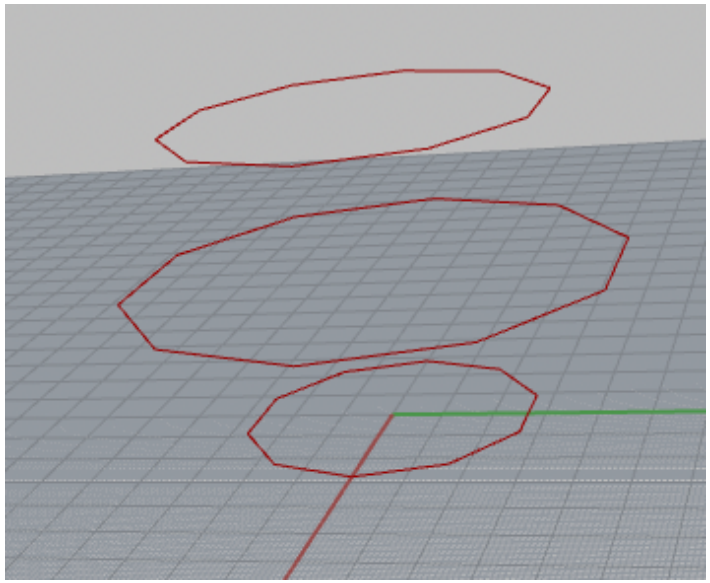




rename them

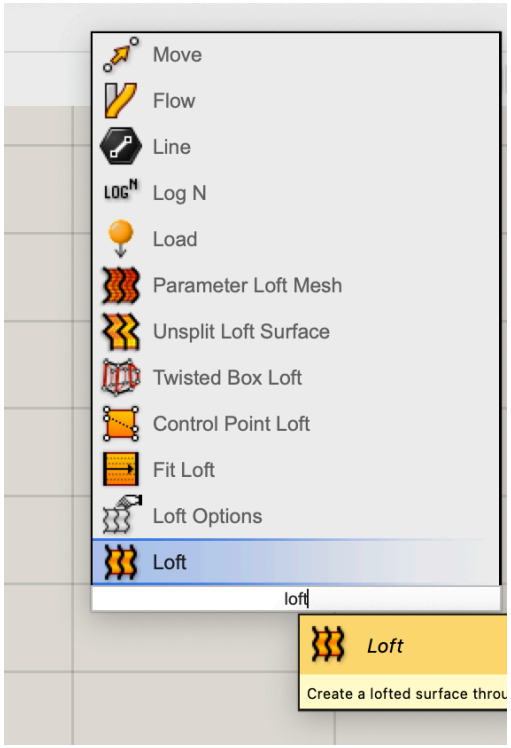
Connect to Python block for r and z

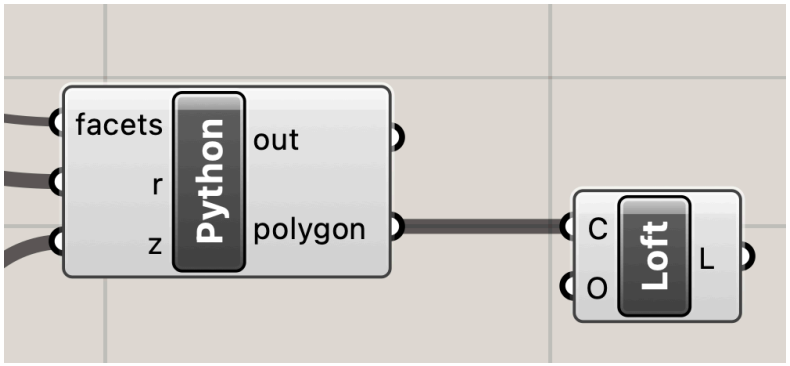


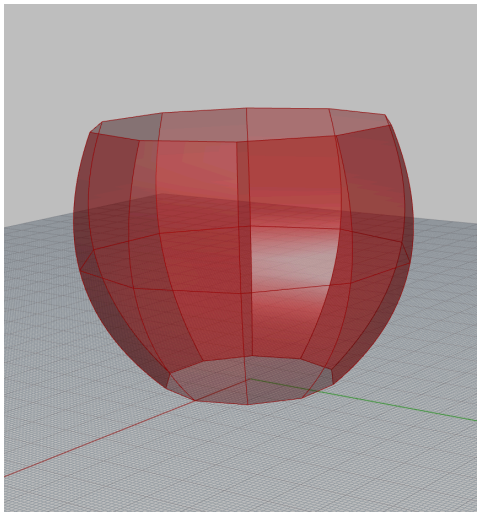


questions?

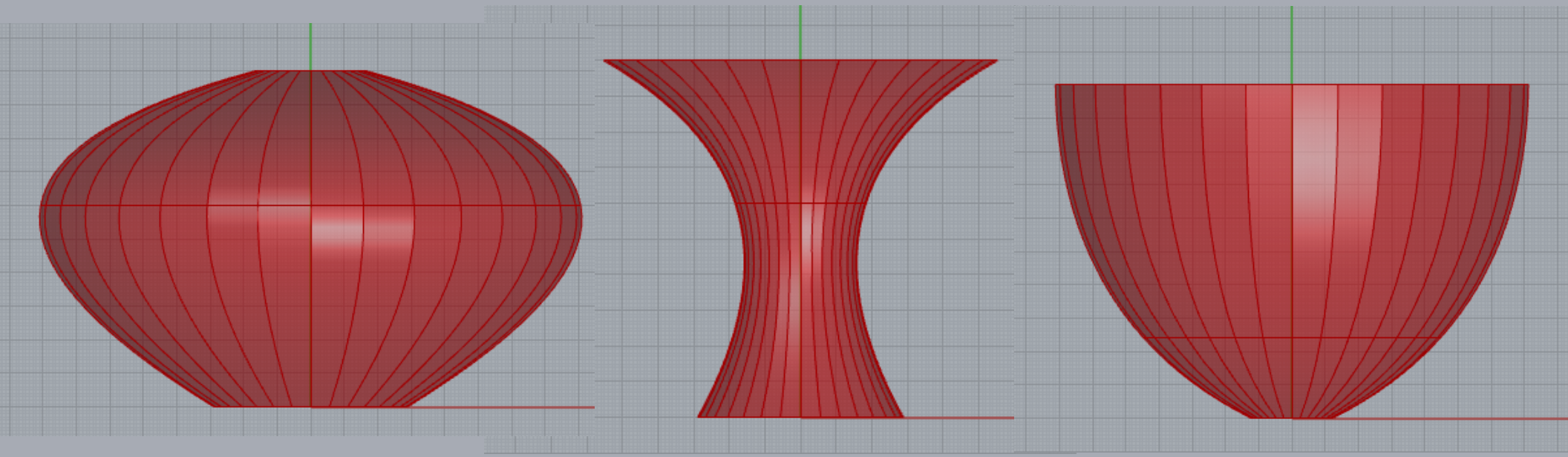
Create a surface using Loft



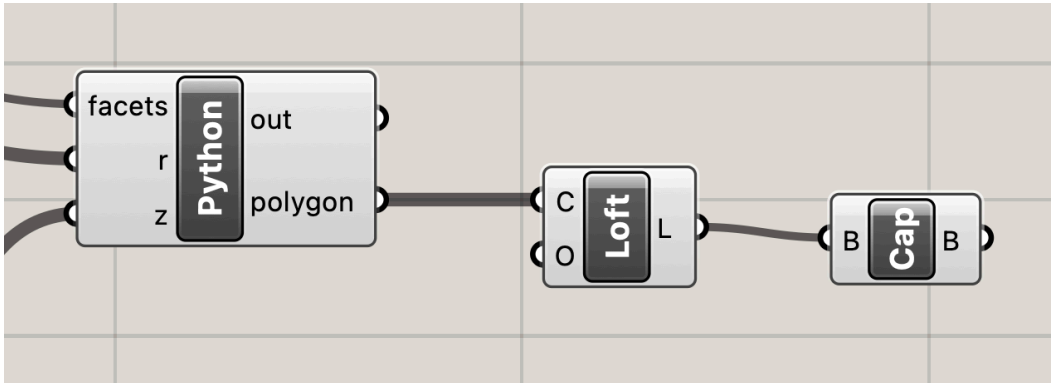


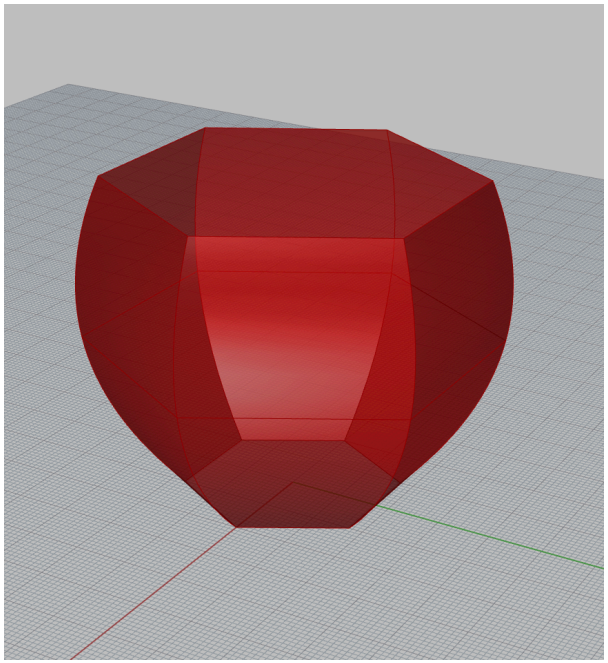


Play with sliders



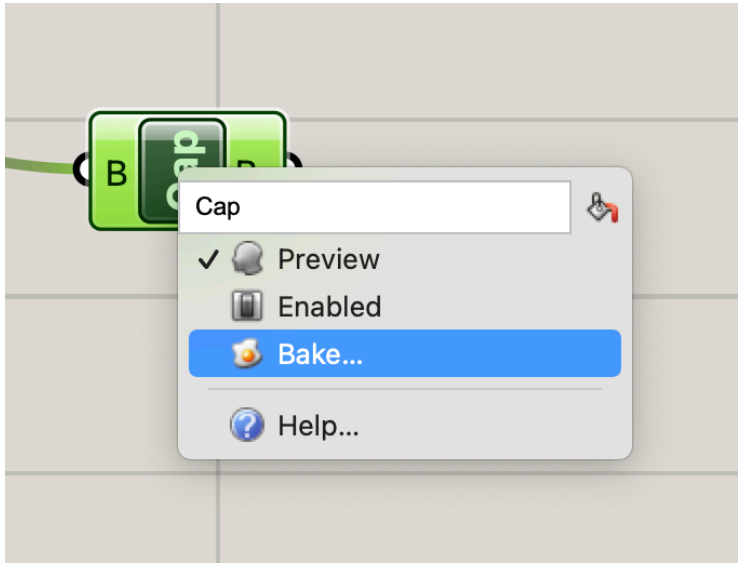
Create a solid with Cap



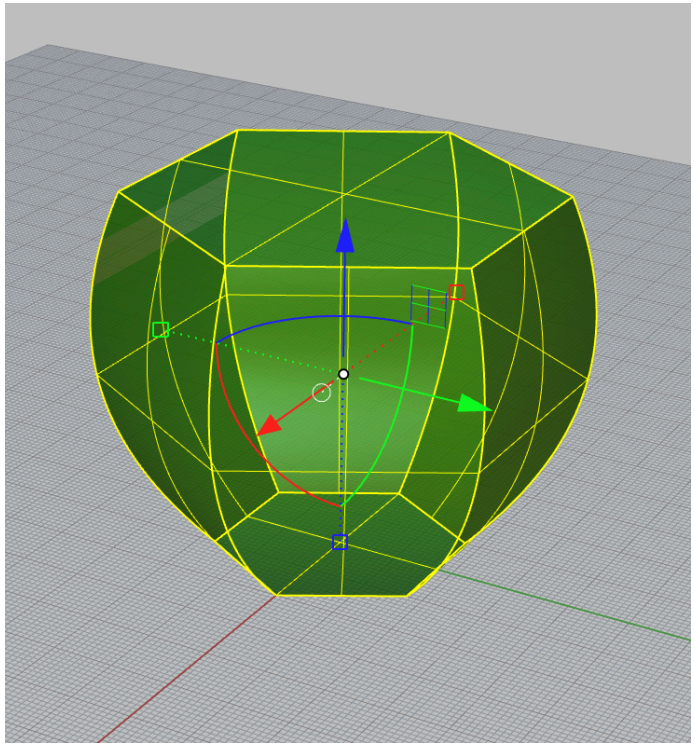


questions?

Bake your shape



right click on Cap



now exists as permanent, fixed Rhino Geometry

Baking

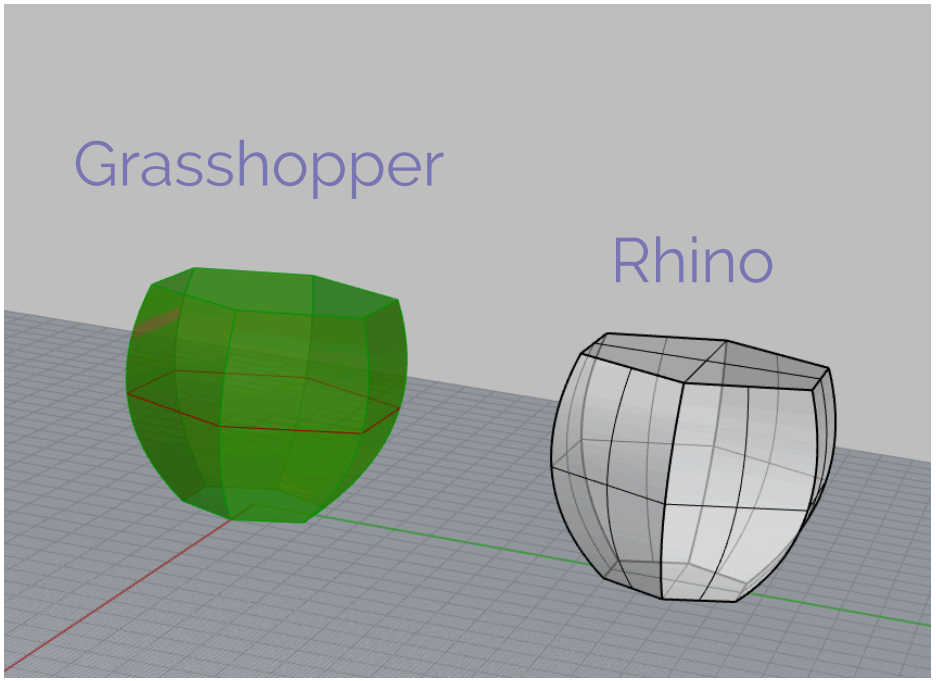
Transforms Grasshopper geometry, which is dynamic and parametric

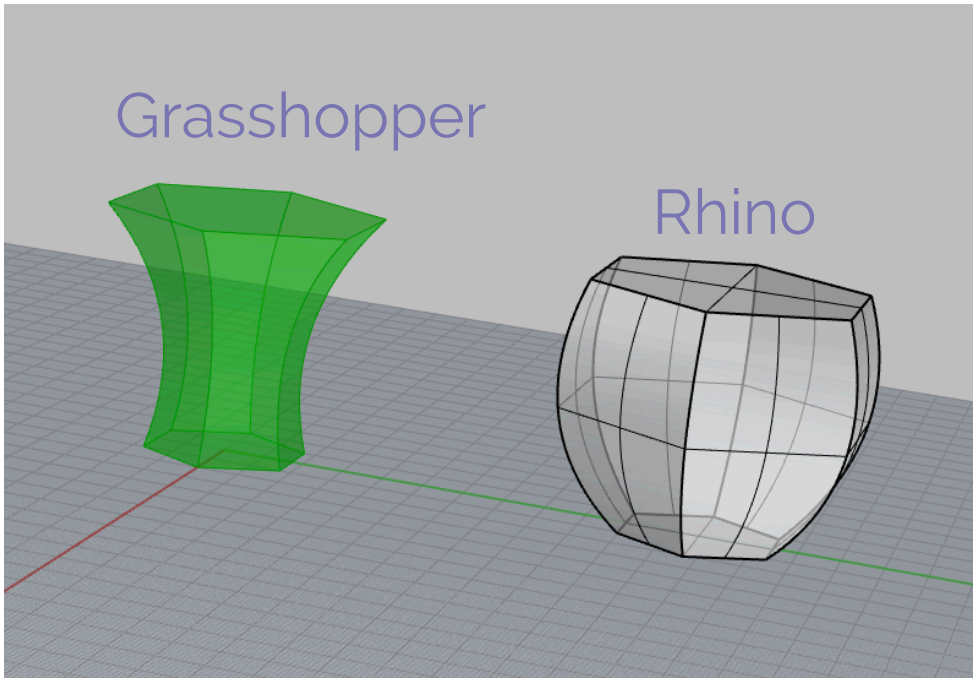
Into Rhino geometry, which is fixed. Rhino geometry cannot be changed by Grasshopper programs.

One you bake something you are done. The baked form remains fixed in Rhino as you continue to work in Grasshopper.

Geometry must be baked into Rhino before 3D printing.

Baking in the workflow: save a shape and keep working





continuing to edit in Grasshopper