Computational Fabrication

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

Large Assignment 2 due Tuesday

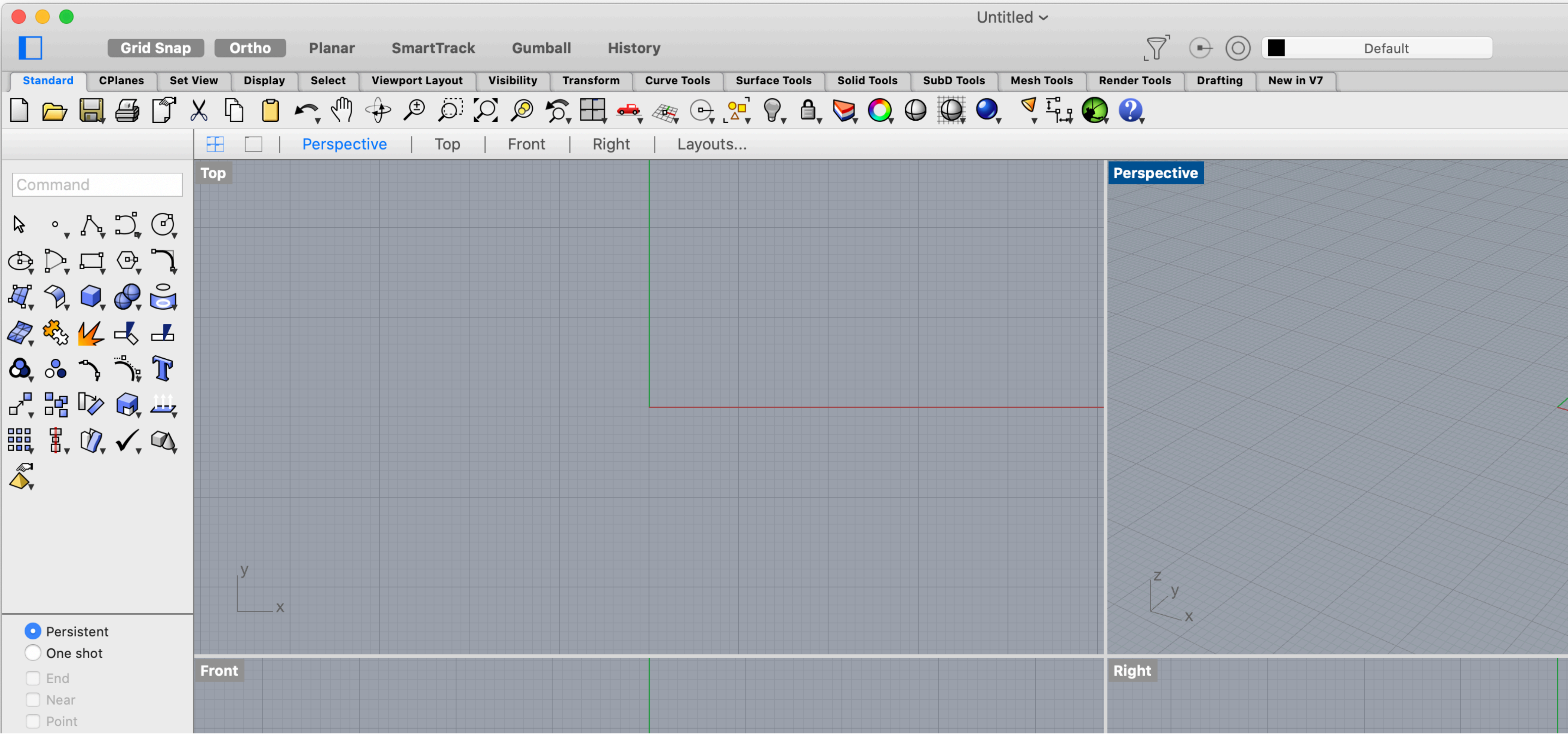
Due midnight Tuesday 9/24

Three parametric 3D printed vessels

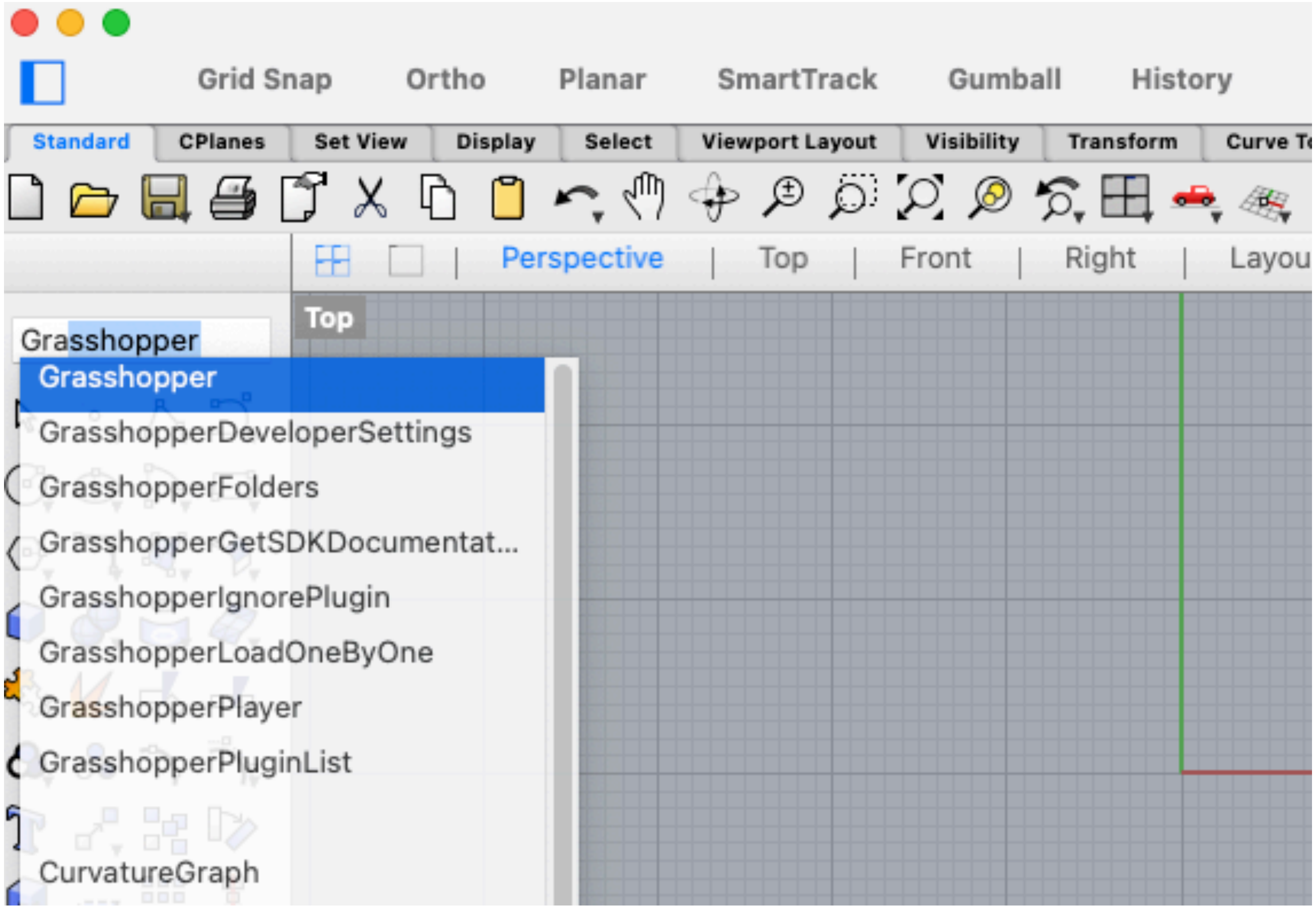
Comments and responses

Rhino, Grasshopper, and Python cont.

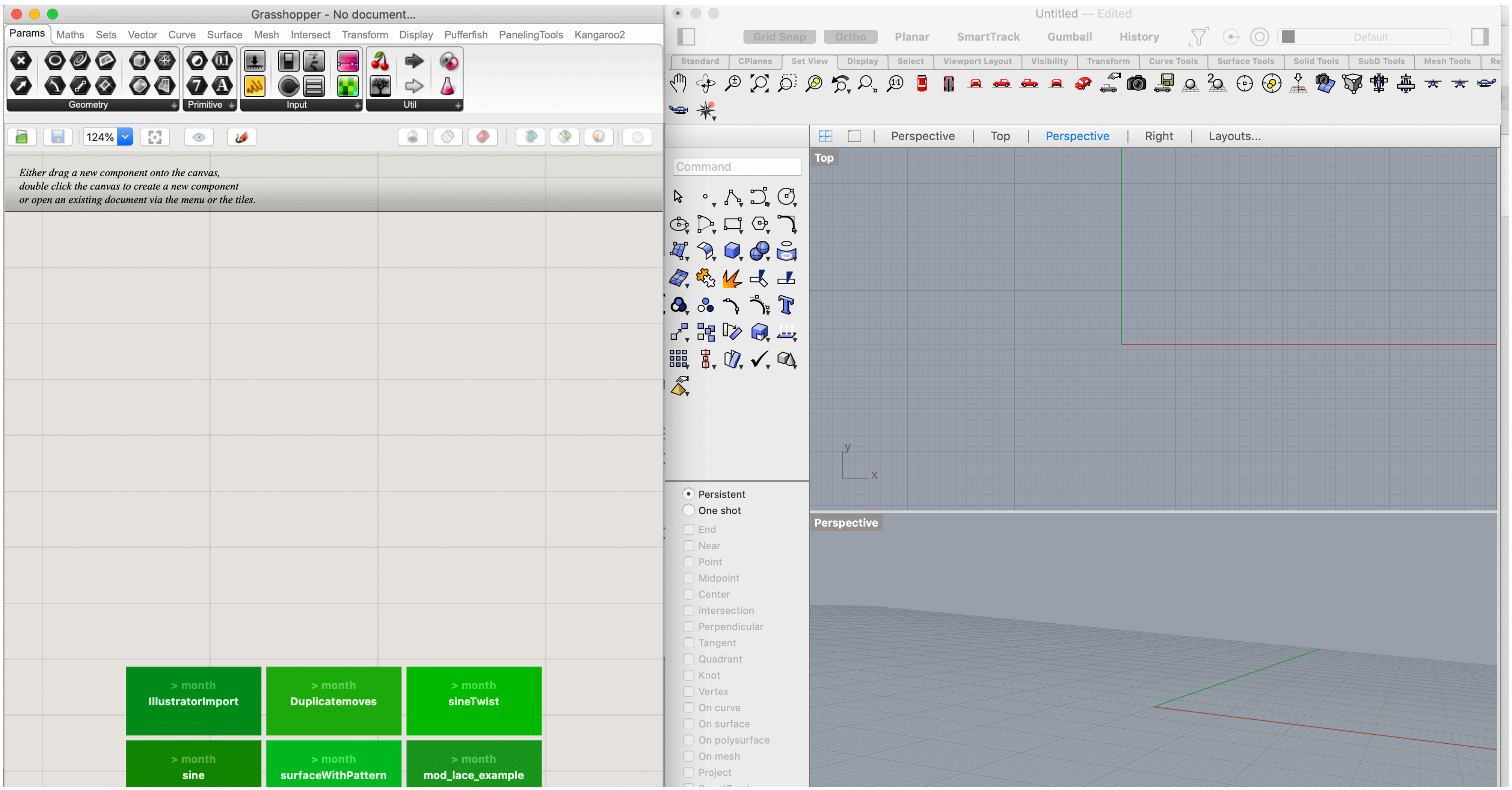
open up Rhino



Open Grasshopper

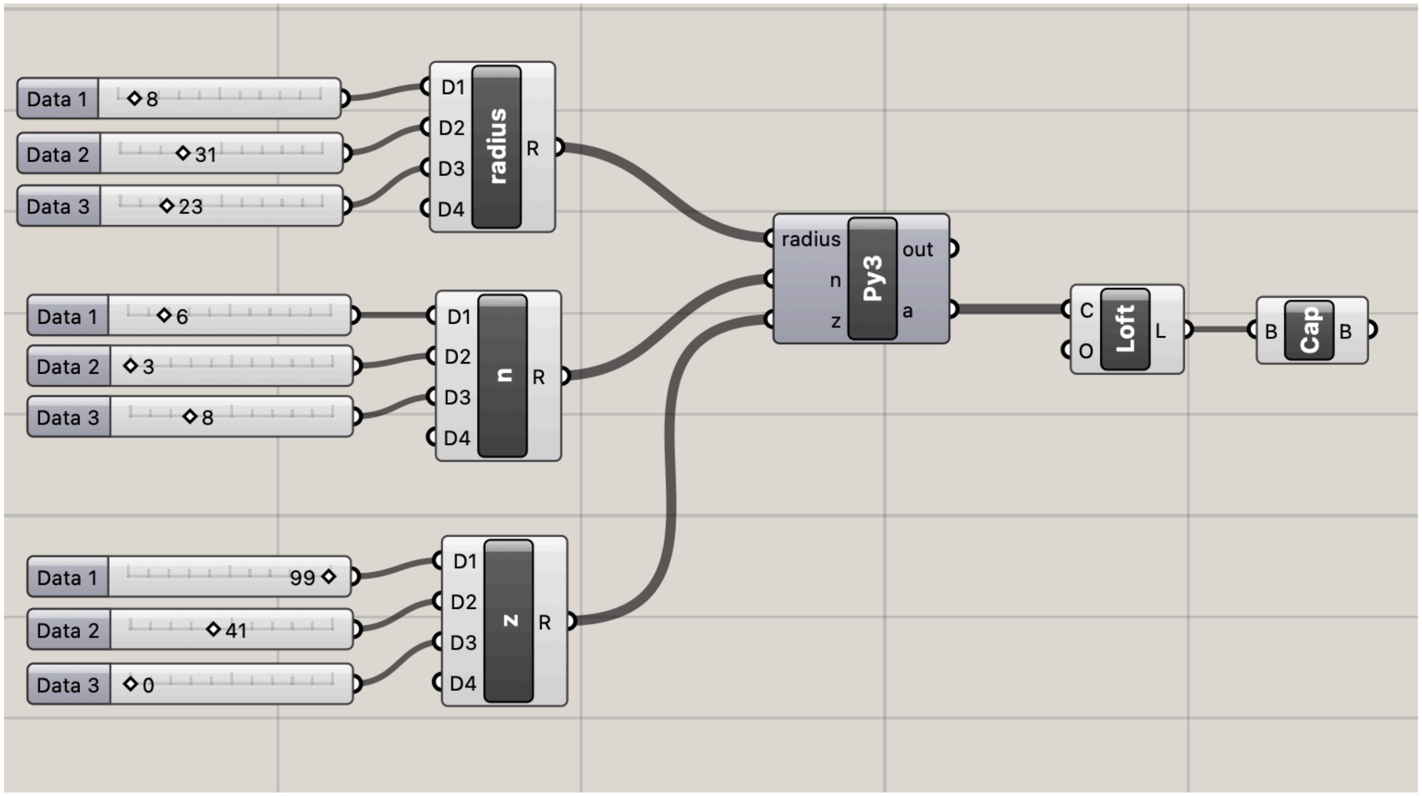


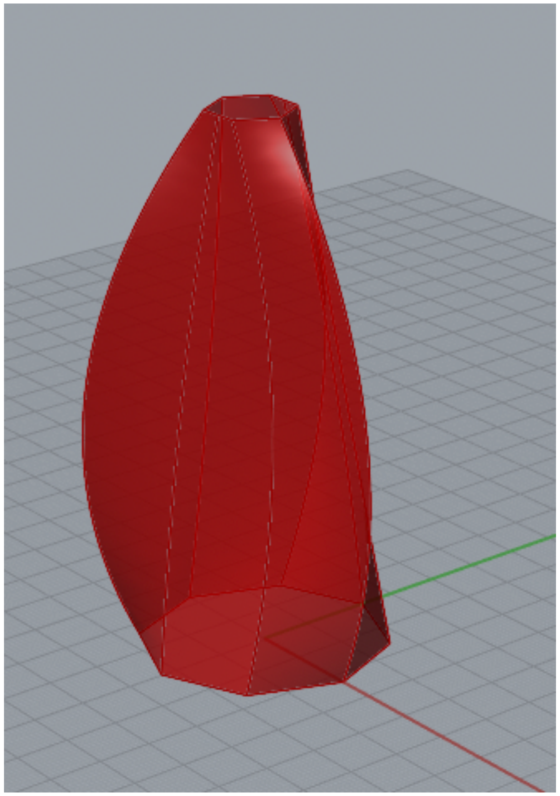
Set up so you can see both applications



Open up the program from last class

Program from last class





To create a vessel, we will:

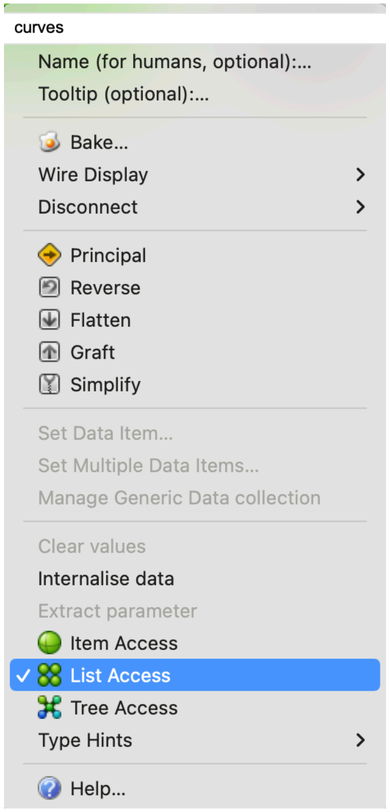
- Create a solid in Python that will define the outside of the vessel. Commands: AddLoftSrf, CapPlanarHoles
- Create a new solid that is offset a distance **d** from the current solid. This will define the inside of the vessel. New command: OffsetCurve
- Subtract the inside shape from the outside shape to create a vessel with walls of thickness **d**. Command: BooleanDifference
- Create a bottom. New commands: AddPlanarSrf, ExtrudeSurface
- Add the bottom to the walls to create a vessel with walls and a bottom. We'll do this in Grasshopper using the Solid Union block.

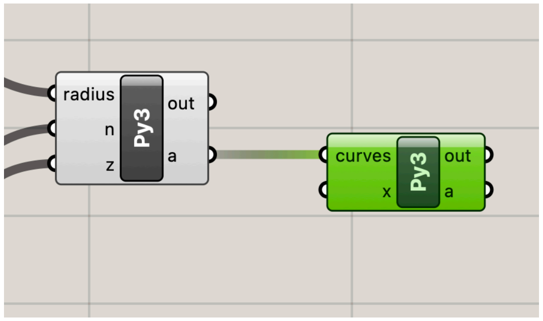
Delete Loft and Cap Grasshopper blocks

Create a solid that defines the outside of vessel

Curves as Input

- Create a new python block, name one of its inputs “curves”
- Select List access. We will work with a list of curves.
- Type hint should be Curve
- Connect the curves output from our first block to this new block

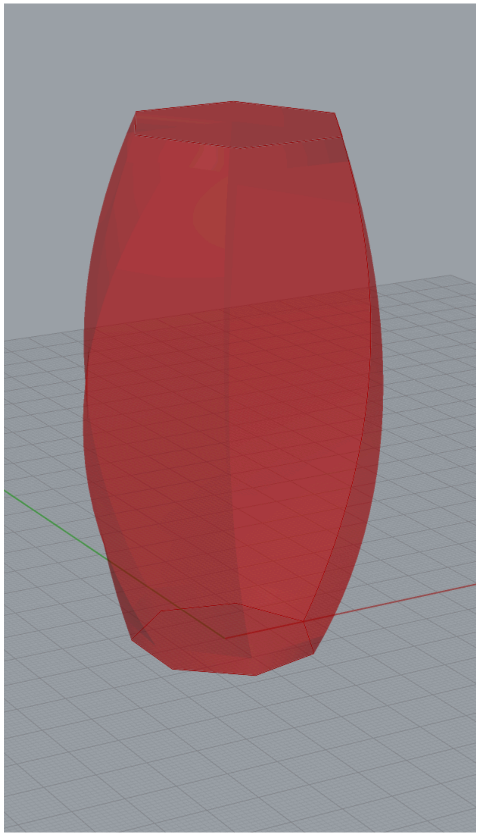




Create a solid that defines the outside of vessel

- Use AddLoftSrf to create a surface through the input curves.
- Use CapPlanarHoles to create a solid. Note: operates on input geometry. Doesn't generate a new object.

```
import rhinoscriptsyntax as rs
import math

vase_outer = rs.AddLoftSrf(curves)
rs.CapPlanarHoles(vase_outer)
a = vase_outer
```



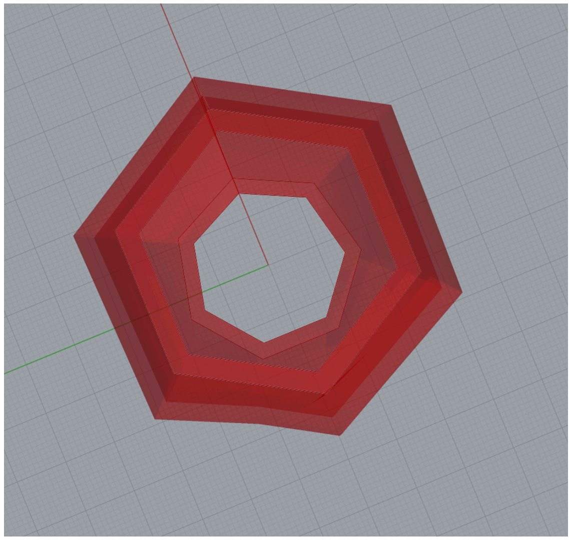
Create a solid that defines the inside of vessel

- Use OffsetCurve to create a set of curves that are offset a distance **thickness** from the original curves. Note: this function returns a list.
- Add a thickness input to your Python block. Good range: -5.0 to 5.0.
- Generate a solid using these offset curves.

```
offset_curves = []
point = rs.CreatePoint(0,0,0)
for i in range (0,len(curves)):
    offset_curve = rs.OffsetCurve(curves[i],point,thickness)
    offset_curves.append(offset_curve[0])

vase_inner = rs.AddLoftSrf(offset_curves)
rs.CapPlanarHoles(vase_inner)
```

Create the vessel walls

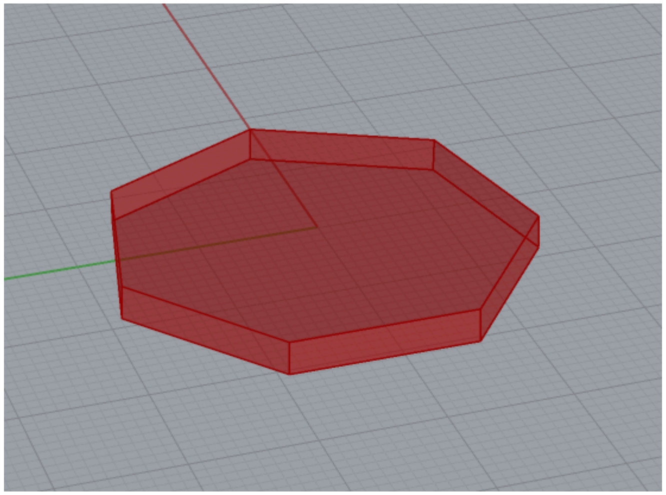
- Use BooleanDifference to create a vessel shell. Note: generates a new object and deletes the two input objects.



```
vase_shell = rs.BooleanDifference(vase_outer, vase_inner)
```

Create the bottom

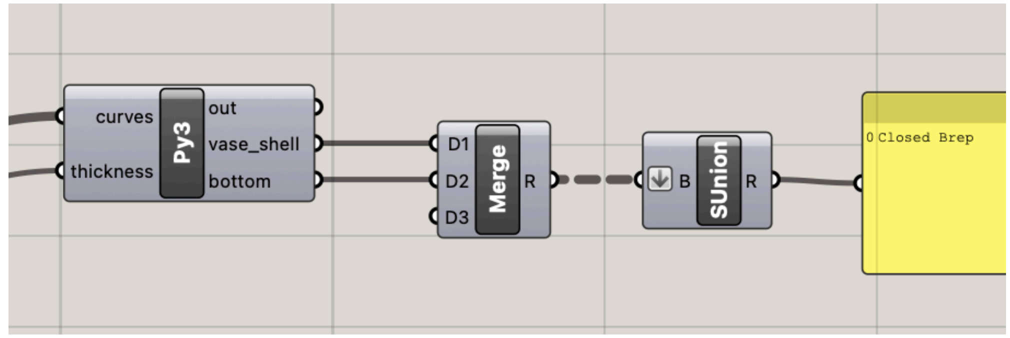
- Use AddPlanarSrf to create a surface defined by the bottom most curve in the list. Note: takes a list as input.
- Use ExtrudeSurface to create a bottom.

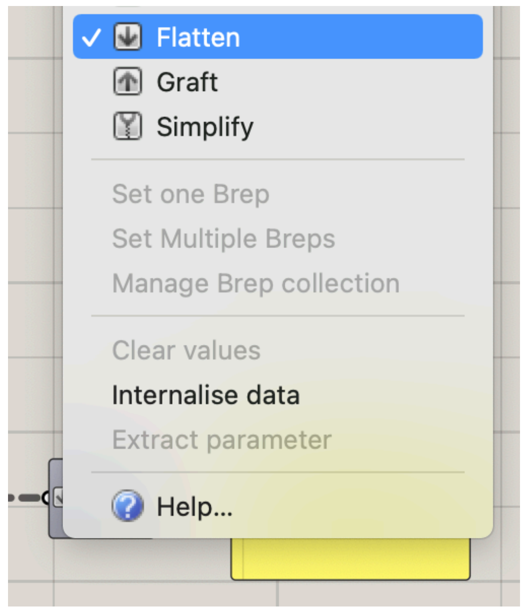


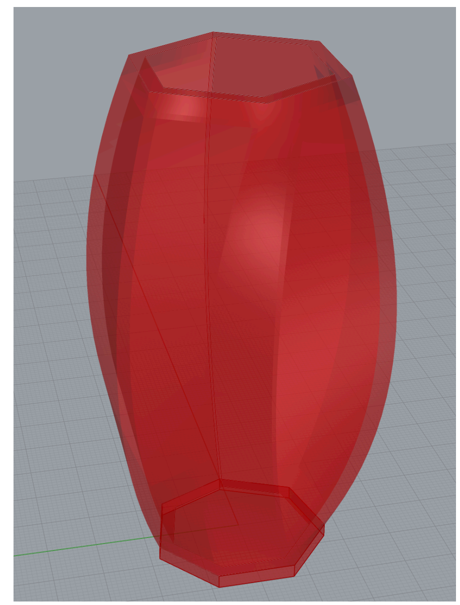
```
bottom_curve = curves[len(curves)-1]
bottom_surface = rs.AddPlanarSrf([bottom_curve])
curve=rs.AddLine(rs.CreatePoint(0,0,0),rs.CreatePoint(0,0,-thickness))
bottom = rs.ExtrudeSurface(bottom_surface,curve)
a = bottom
```

Join the sides and bottom together in GH

- Rename the **a** output from your Python block to your vessel wall variable.
- Add a second output for your bottom.
- Add a merge block to create a list of the two outputs.
- Use a Solid Union block to join the two solids together. The final result should be a Closed Prep. Note: Flatten the input to this block. Right click & select Flatten.

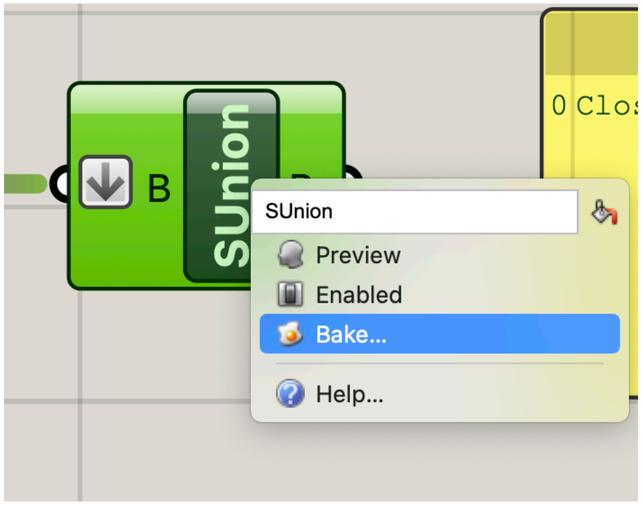


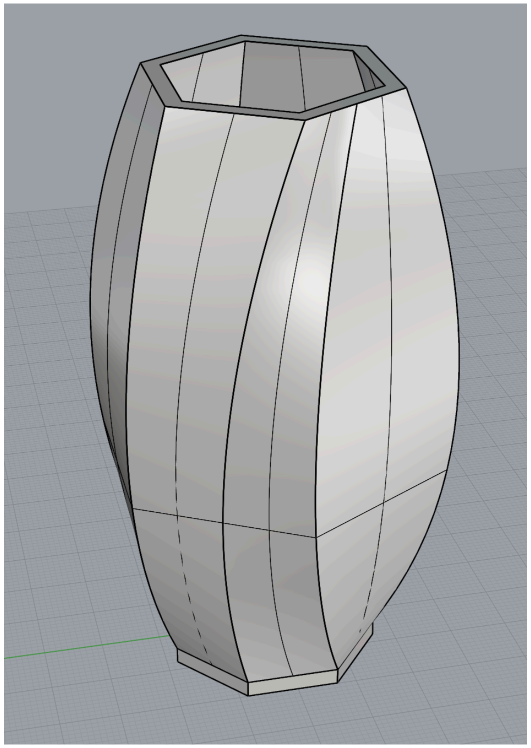




Bake your shape to create a Rhino object

- Double check the size of your vessel.
- Units are mm.
- Right click on the Solid Union block and click Bake...



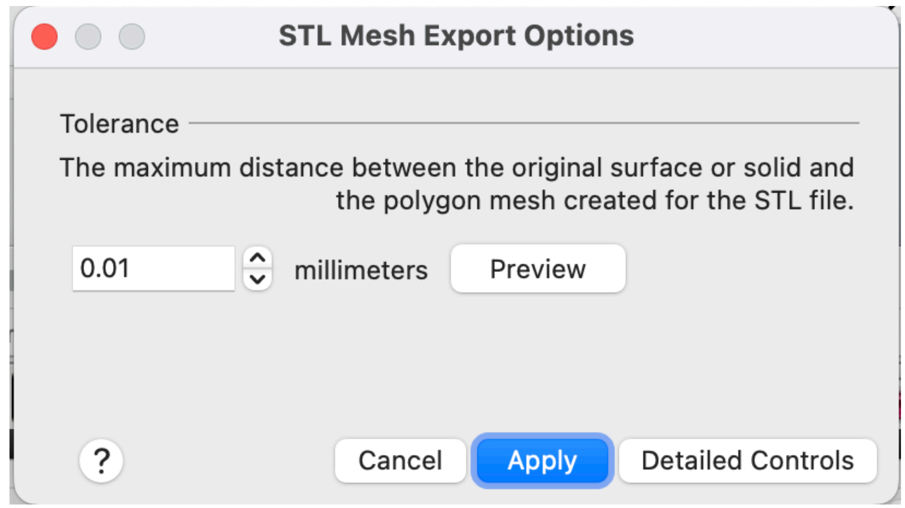


Rendering in Rhino

- To generate a nice image of your part, select Rendered from the View menu in Rhino.

Export as .stl

- If you haven't already, doublecheck the size of the object being generated in Python & GH. Units: mm
- Select your object.
  Under the File menu, select Export Selected.
- Select .stl as the file type
- In the Mesh Export Options box, select a tolerance of .01mm or lower.



Transformations

Now we’ll experiment with transforming curves before lofting them.

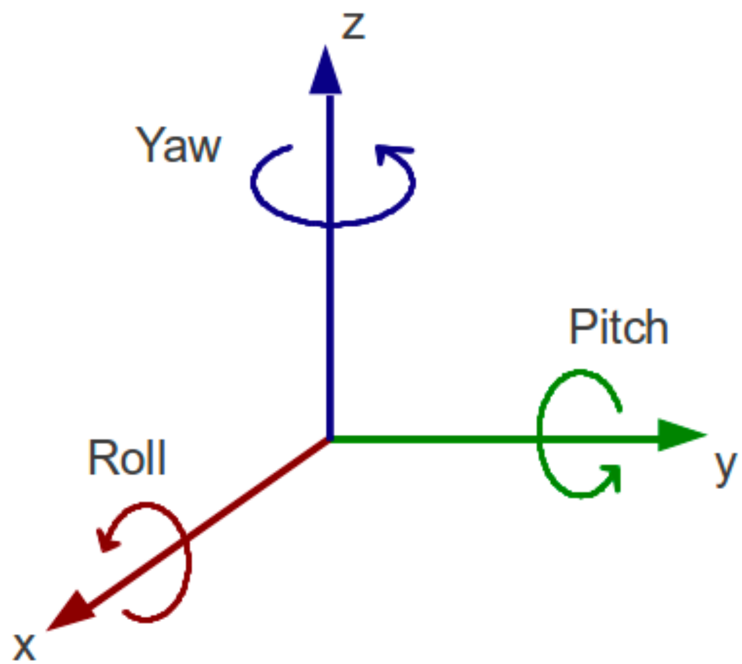
We'll use transformation tools from the Rhino Geometry library:

https://developer.rhino3d.com/api/rhinocommon/rhino.geometry

Rhino Geometry library
is separate and different from
Rhinoscript library

We’ll twist/rotate our curves
around the Z-axis

Rotation in 3D



RotationZYX method

Class: Rhino.Geometry.Transform

Description:

Create rotation transformation From Tait-Byran angles (also loosely known as Euler angles).

Syntax:

```
static Transform RotationZYX(
   Double yaw,
   Double pitch,
   Double roll
)
```

Parameters:

yaw
Type: System.Double
Angle, in radians, to rotate about the Z axis.

pitch
Type: System.Double
Angle, in radians, to rotate about the Y axis.

roll
Type: System.Double
Angle, in radians, to rotate about the X axis.

Returns:

Type: Transform

A transform matrix from Tait-Byran angles.

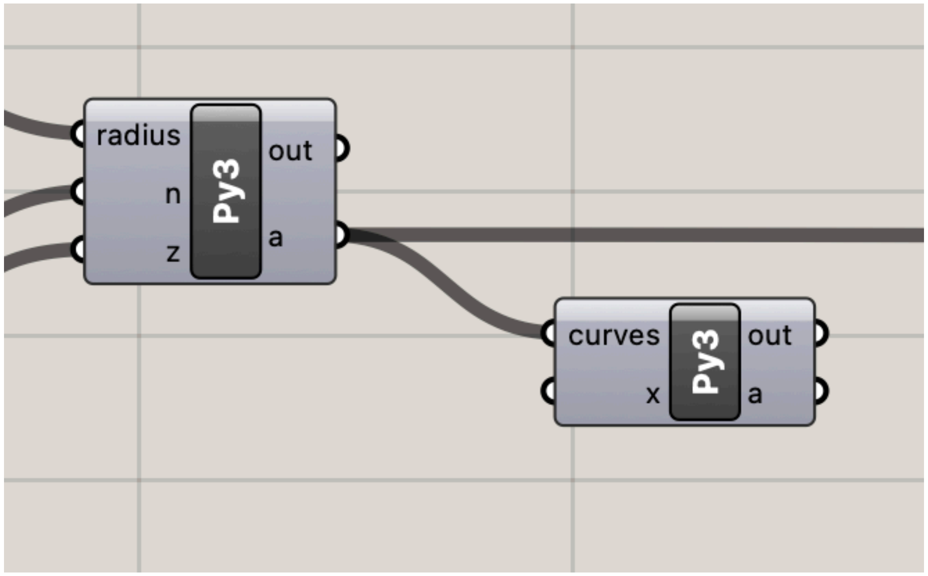
Remarks:

RotationZYX(yaw, pitch, roll) = R_z(yaw) * R_y(pitch) * R_x(roll) where R_*(angle) is rotation of angle radians about the corresponding world coordinate axis.

https://developer.rhino3d.com/api/rhinocommon/rhino.geometry.transform/rotationzyx

Add a new Python block

- This will also take the curves output from the first block as input.
- Like you did for the last Python block, rename the input variable, select List access and choose Curve for the type.

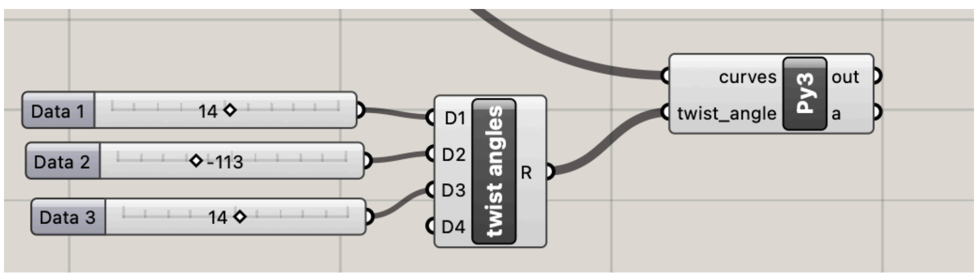


Write a twist function

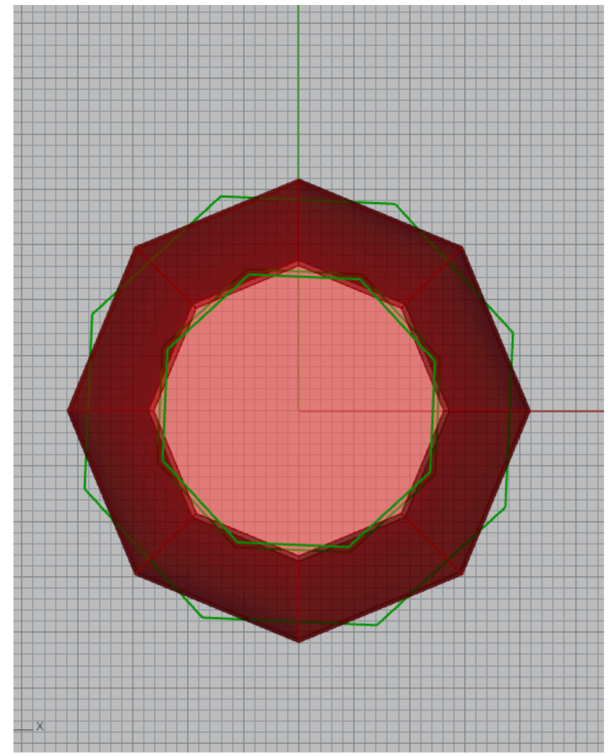
```
import rhinoscriptsyntax as rs
import Rhino.Geometry as geom
import math

def twist(curve, angle):
    twist = geom.Transform.RotationZYX(math.radians(angle),0,0)
    curve.Transform(twist)
```

Apply the twist function to our polygons

- Add a new variable for twist angle.
- Create and merge 3 input sliders for twist. Good range: -360 - 360
- Apply the twist function to your input curves. Note: transformations act on the input geometry.




```
for i in range (len(curves)):
    twist(curves[i],twist_angle[i])
```

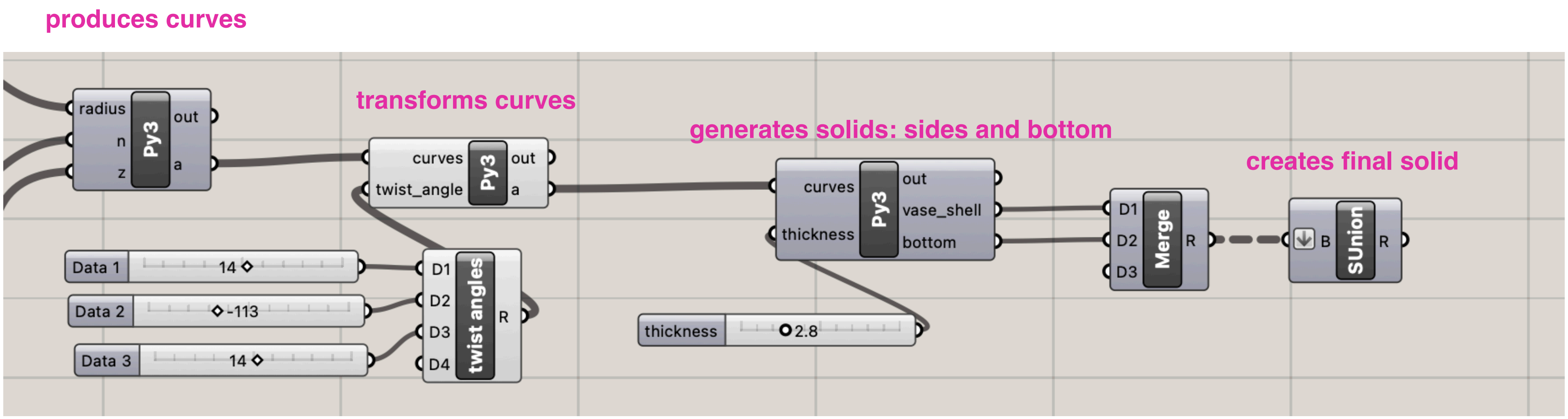


green shows rotated polygons

Transformations, process

- Create a transformation using geom.Transform.RotationZYX() or other method. This returns a transformation matrix.
- Apply the returned matrix to your geometry. ie: curve.Transform(your_transformation)
- You can define your own transformation matrices and use them in the same way. See: https://developer.rhino3d.com/api/rhinocommon/rhino.geometry.matrix
- More info: https://developer.rhino3d.com/api/rhinocommon/rhino.geometry.transform

Use transformed curves as input to vase generator



Play with twists and other transformations

Thank you!

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