Preparing Graphics for Design & Prototyping

OA Fakinlede

oafak@unilag.edu.ng lms.s2pafrica.com

Purpose of Lecture

- This is a response to the questions asked two weeks ago:
  - Many wanted to extend the power of fusion 360 to create things that are not directly supported. The spiral spline was a simple example. Today, the code for this is presented and extended to complete whole components the API using Python
  - Implications of this is enormous in terms of future projects, Capstone projects and issues of Design, Automation, Simulation and Virtualization
  - You will learn how to start and wade through the world of API, Python Scripting, the creation of Add-Ins and projects that prepare models for Simulation, testing and Analysis

Install Python and Visual Studio Code

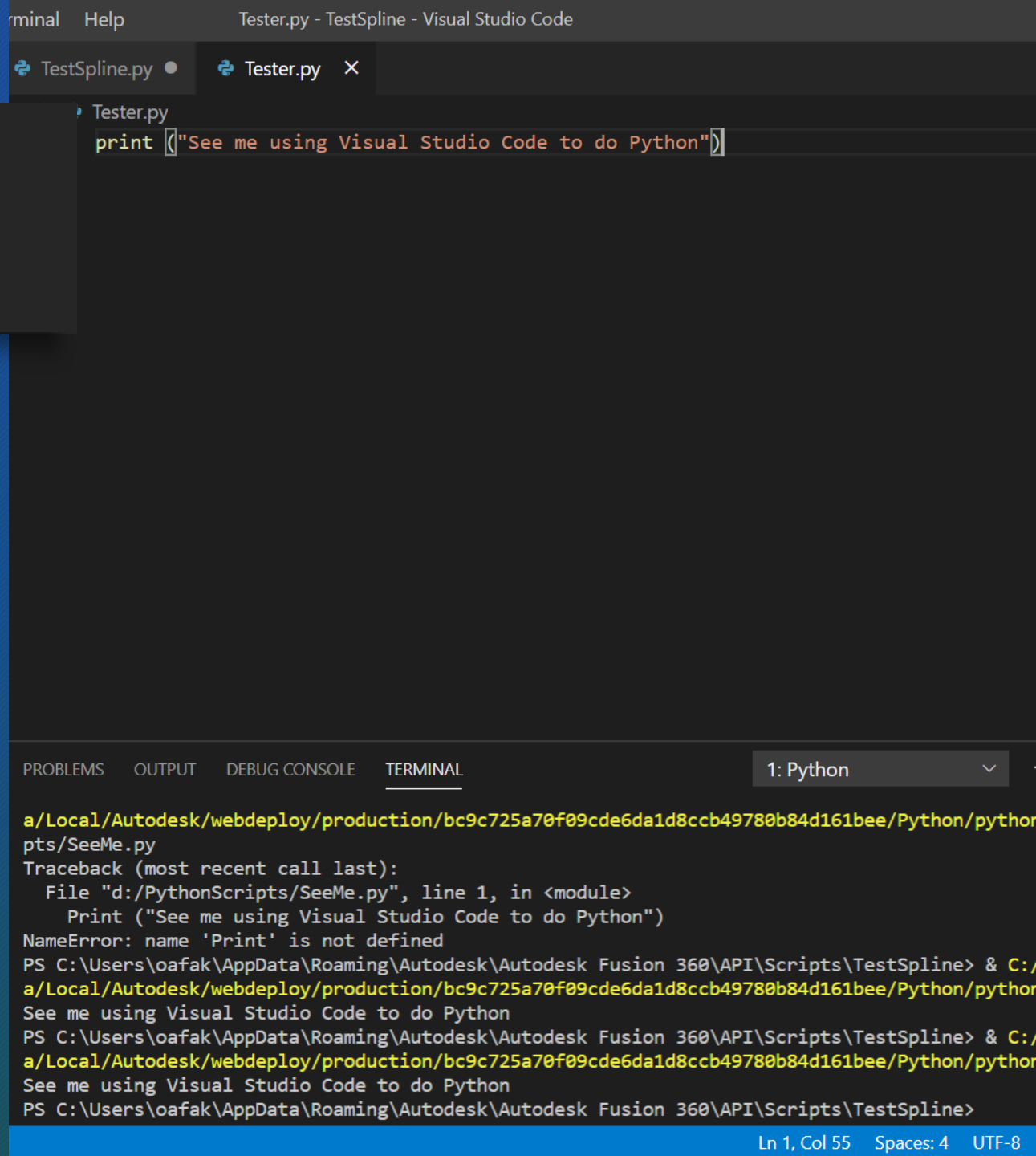
- We describe here the simplest way to go about things here. There are other methods, but time does not permit exhaustive treatment.
  1. Install Python by visiting python.org look for Downloads there and find version 3.8.4 – the latest release. It is free.
  2. Install Visual Studio Code by visiting https://code.visualstudiocode.com/Downloads
- It should take a maximum of 30 minutes to carry out the two activities on a normal computer. A system capable of graphics should be OK.

Install Python and Visual Studio Code, Cont'd

- Do this installation by yourself.
  - There are people making YouTube videos for these. Use them if you care to.
  - Took only like 10 minutes for both installed and they are simple to follow.
- Make sure you read the instructions
  - Accept that either code places itself on the search path of your computer logic. After installation, restart your computer to activate the "Path". This way, they will be able to find each other. Fusion 360 will find VSC on its own
  - Fusion 360 automatically defaults to using visual stusio code. More faciful environments such as PyCharm, Anaconda Studios can be used for this. But, if you want the easy way to do things, Use Visual studio code.

First Program

- Launch Visual Studio Code and type the following line into it:
- print(“see me doing blah, blah blah”) as shown in the screen shot.
- The print keyword, and the string literal will both will change color as shown. If that does not happen, safe your file and do one of two things:
    1. Ensure its extension is .py
    2. In the file type selection that Visual Studio Code gives you, select Python from the list.
- Green button on the top right of the screen launches your program and, Voila! You are a Python programmer!



Programming in Fusion 360

The Fusion 360 API is an enormous behemoth. It is easy to start, it is a bit more difficult to follow. However, patience and perseverance will place you ahead of the pack.

Spend the time, take all the help you need to become familiar with it. Here are the things that will greatly assist your knowledge:

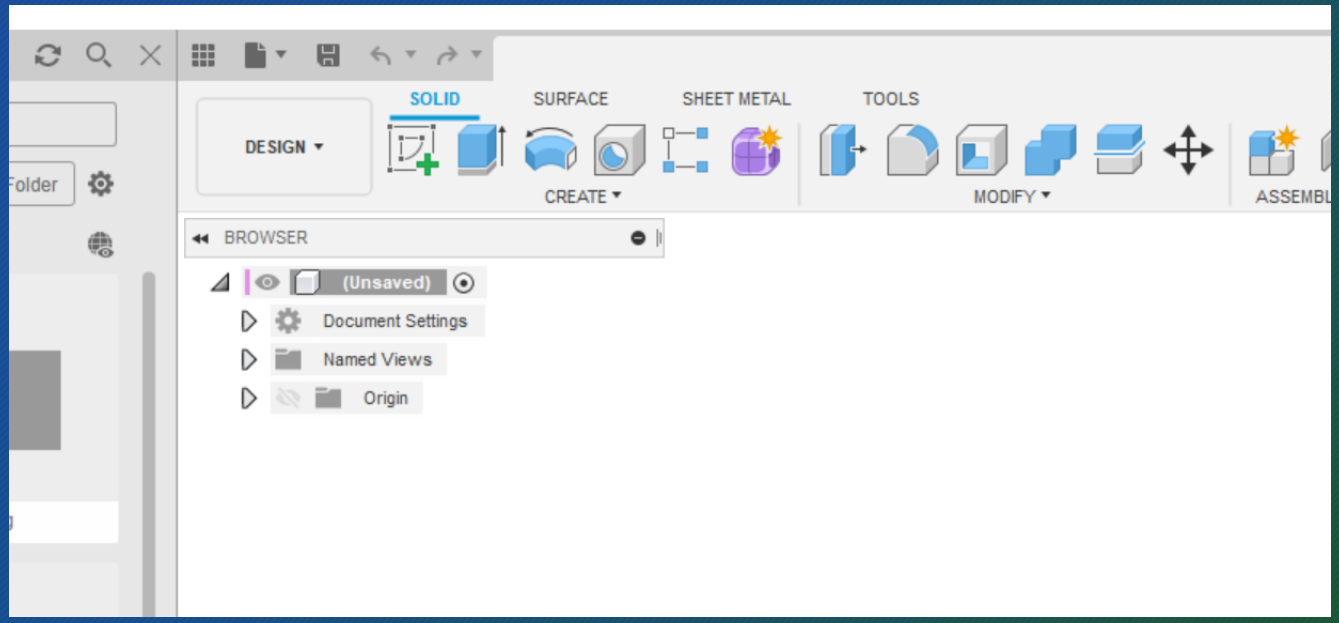
Understanding the Object Model (OOP).

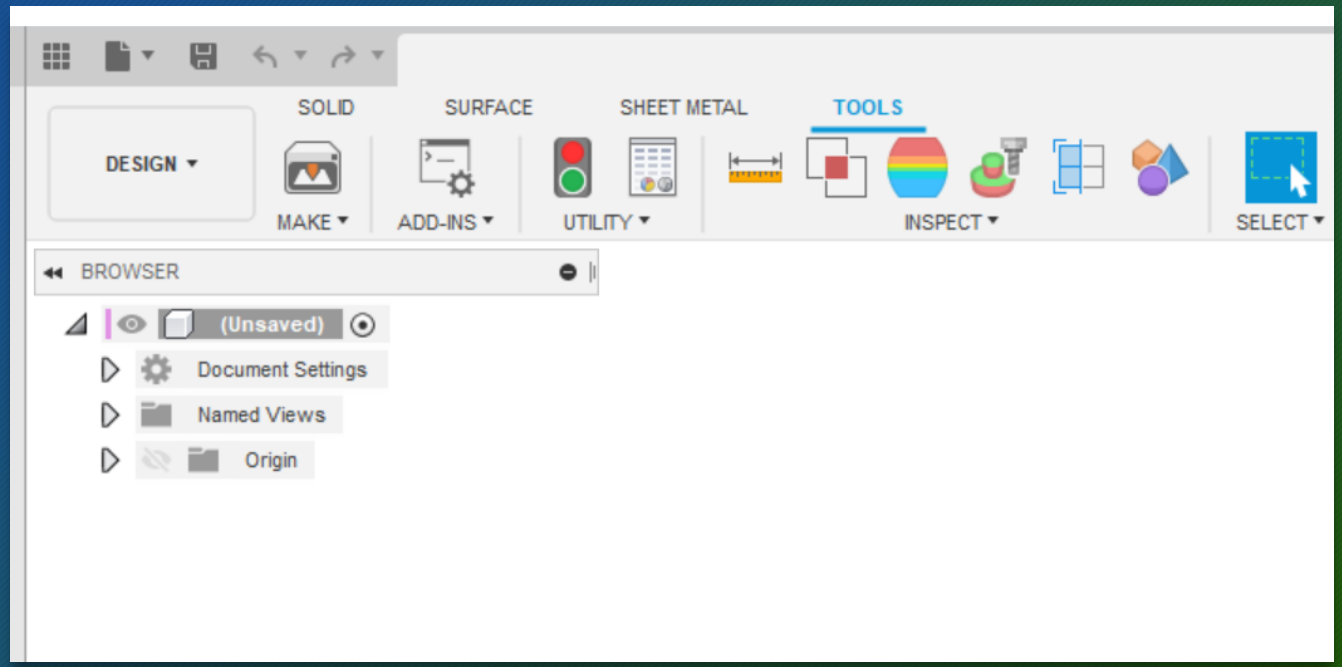
Knowing Python

Ensuring you test everything you see: Make no assumptions that you understand. You DON NOT UNDERSTAND ANYTHING YOU HAVE NOT TRIED!

Diving In

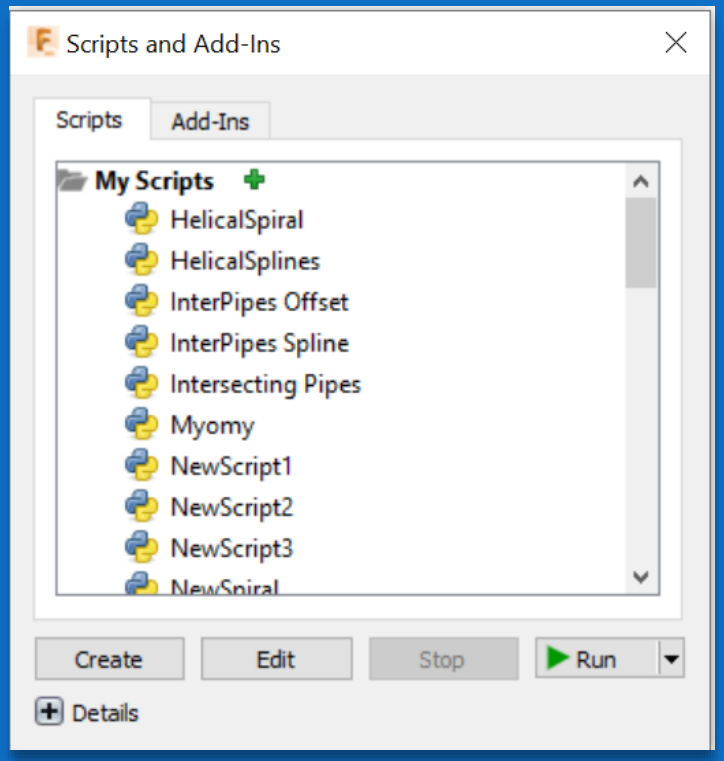
- From your Design Environment, the first screen shows the headings: SOLID, SURFACE, SHEET METAL and TOOLS
- Select TOOLS and you will see the second Screen. The second button after DESIGN and MAKE is the one you are looking for: ADD-INS
- Clicking it allows you to choose between Scripts and ADD-INs and going to the Fusion 360 store.
- Once you select the first option, you are presented with the next screen.





Creating, Editing & Launching Scripts

- Don’t be too fast with this Window.
  - Look around and understand what you are presented with.
  1. See the list of predefined scripts you can learn from. Be inquisitive. Look at some of these files and see the things you will need to do to succeed ion scripting.
  2. You can create a new Script here. Fusion 360 already knows that you have installed VSC and will find it! If you installed Anaconda or PyCharm, you are on your own!
  3. It will store your scripts for you when you ask it to save them. And those Python scripts meant to be executed for graphics in Fusion 360 are special, as you will see.
  4. You want to create a new script, select create. The other button options are straightforward.



The Bottle

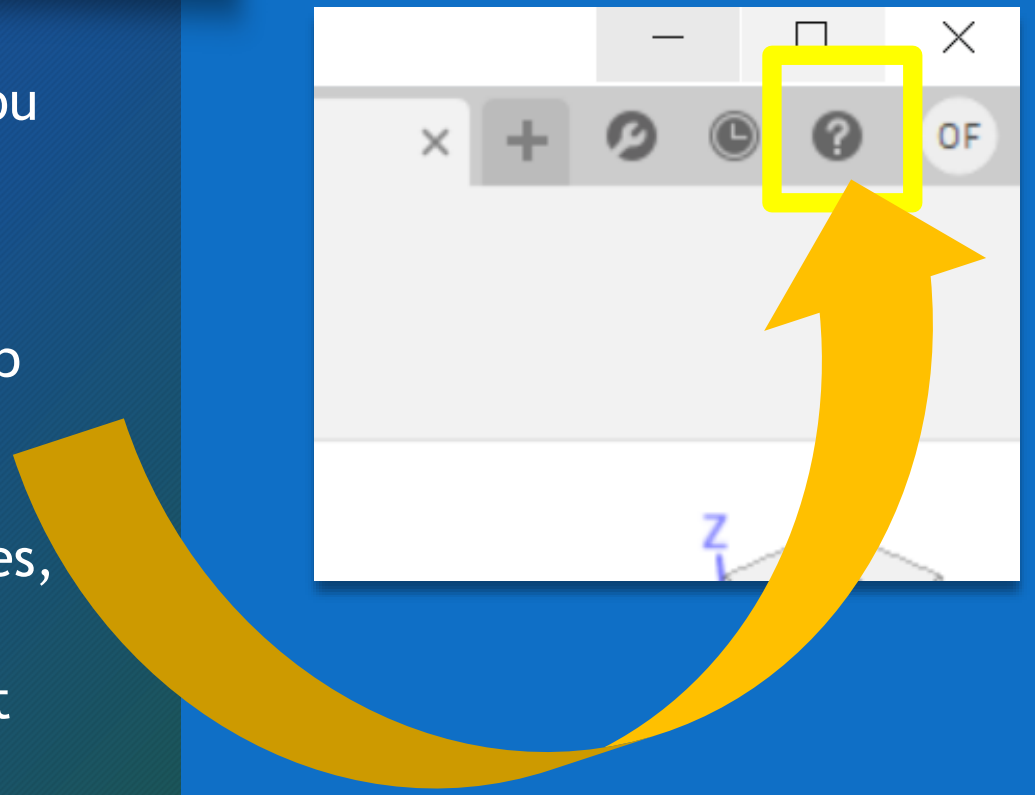
- Why not simply scroll down here and select the bottle script. Double clicking on it immediately executes the script and you will see this bottle.
- Go back again to the script menu. Now select it a little more gently with a single click. Select “Edit” button. What do you see?
  - Immediately, Visual Studio Code is launched.
  - You can see the code that created the bottle!
  - Two choices are before you:
    1. Become intimidated and assume that angels from heaven wrote that code
    2. Become excited, knowing that in a month, you will be creating your own scripts even more complicated than this! Make your choice!

Things to note

- There are other ways to code into the Fusion 360 Programming Interface, API
  1. Object-Oriented C++
  2. Python. I chose to work with Python and strongly recommend that you do so too. I say this, even if you are good in C++. We will eventually no longer need Mathematica because there are Python libraries that achieve the same goals as we did in Mathematica in a more unified language. It is a great idea to be able to do so many things in a single consistent environment.
  3. The python code here are essentially the same steps you will take to manually create this bottle in Fusion 360. “Essentially” is the word – not “exactly”!

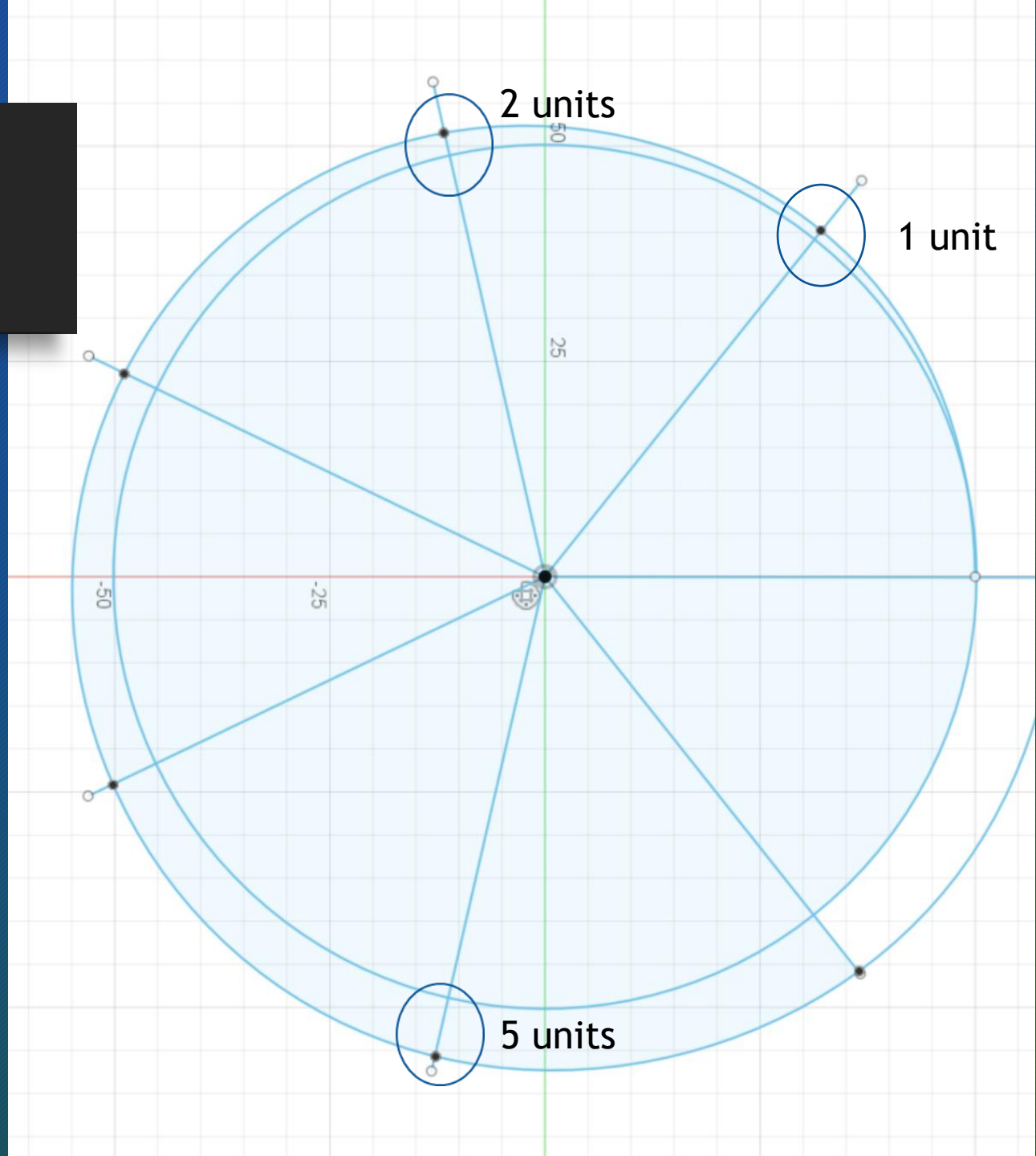
Explore, play around

- You lose nothing, and gain a lot of insight when you explore this environment
- You can get direct help from with Fusion 360 by going to the top right corner of your screen: Just before your initials, you see a question mark (Help Menu) as shown in the screen shot here.
- Clicking on it you can select “Learning and Documentation” under which, among three choices, you will see Fusion 360 API
- You will get a lot of assistance from following that link.



First Python Script: Create a Spiral

- We will begin by creating a spiral in Fusion 360
- What is a spiral? Programming begins, not with writing code, but understanding what you are trying to do. Note that a spiral is a curve which bends like a circle but with increasing (or decreasing) radius as shown.
- We can choose to make this simple spiral where the same amount of increase occurs at each angular shift. Here we are using seven steps. Choose the number of steps you like.
- We are trying to code the logic of adding these units at each of the seven steps. Once we do that, we get the spiral.



The Logic

```
while i <= splinePoints:
    t = startRange + ((endRange - startRange)/splinePoints) * i
    xCoord = ((rad + t /(2*math.pi)) * math.cos(t))
    yCoord = ((rad + t /(2*math.pi)) * math.sin(t))
    points[j].add(adsk.core.Point3D.create(xCoord, yCoord, 0))
    i = i + 1
```

- The central logic of the spiral is inside this while loop of Python.
- You can see that instead of the x, y, and z coordinates simply being $r\cos t$, $r\sin t$, 0, we shall have $(r+n\alpha)\cos t$, $(r+n\alpha)\sin t$, 0 which are now the parametric equations of the spiral.
- Once the central logic is implemented, you need to look at the protocol and housekeeping of the way you will need to present your code so that Fusion 360 knows exactly what you mean. Much of what it already knows is given to you as we shall see
- It should not surprise that we select the spline points to be 7 here.

The Housekeeping I

- The code to make this spiral is 47 lines. That may, initially look like a lot. Out of the 47, apart from the lines of logic described above, I was responsible for only about five more lines! Fusion 360 helps you along!
- Once you select “Create” in the menu described earlier, much of the first 16 lines are generated as a result of your input. If you are observant, you will be asked to name the code, Tell who the author is, etc. These are used to help you arrange your code.
    1. Line 4: add math because you are using the math library
    2. Line 9: Your own message
    3. Line 14: the programmatic equivalent of “Create Sketch” with the selection of the XY plane as construction plane.

SpiralsOne.py > ...

```
1   #Author-OA Fakinlede
2   #Description-Linearly increasing spiral
3   import adsk.core, adsk.fusion, adsk.cam, traceback, math
4
5   try:
6       app = adsk.core.Application.get()
7       ui = app.userInterface
8       ui.messageBox('Linear Spiral Creation')
9       doc = app.documents.add(adsk.core.DocumentTypes.FusionDesignDocumentTyp
10      design = app.activeProduct
11      # Get the root component of the active design.
12      rootComp = design.rootComponent
13      # Create a new sketch on the xy plane.
14      sketch = rootComp.sketches.add(rootComp.xYConstructionPlane)
15      # Create an object collection for the points in each spline.
16      points = adsk.core.ObjectCollection.create()
```

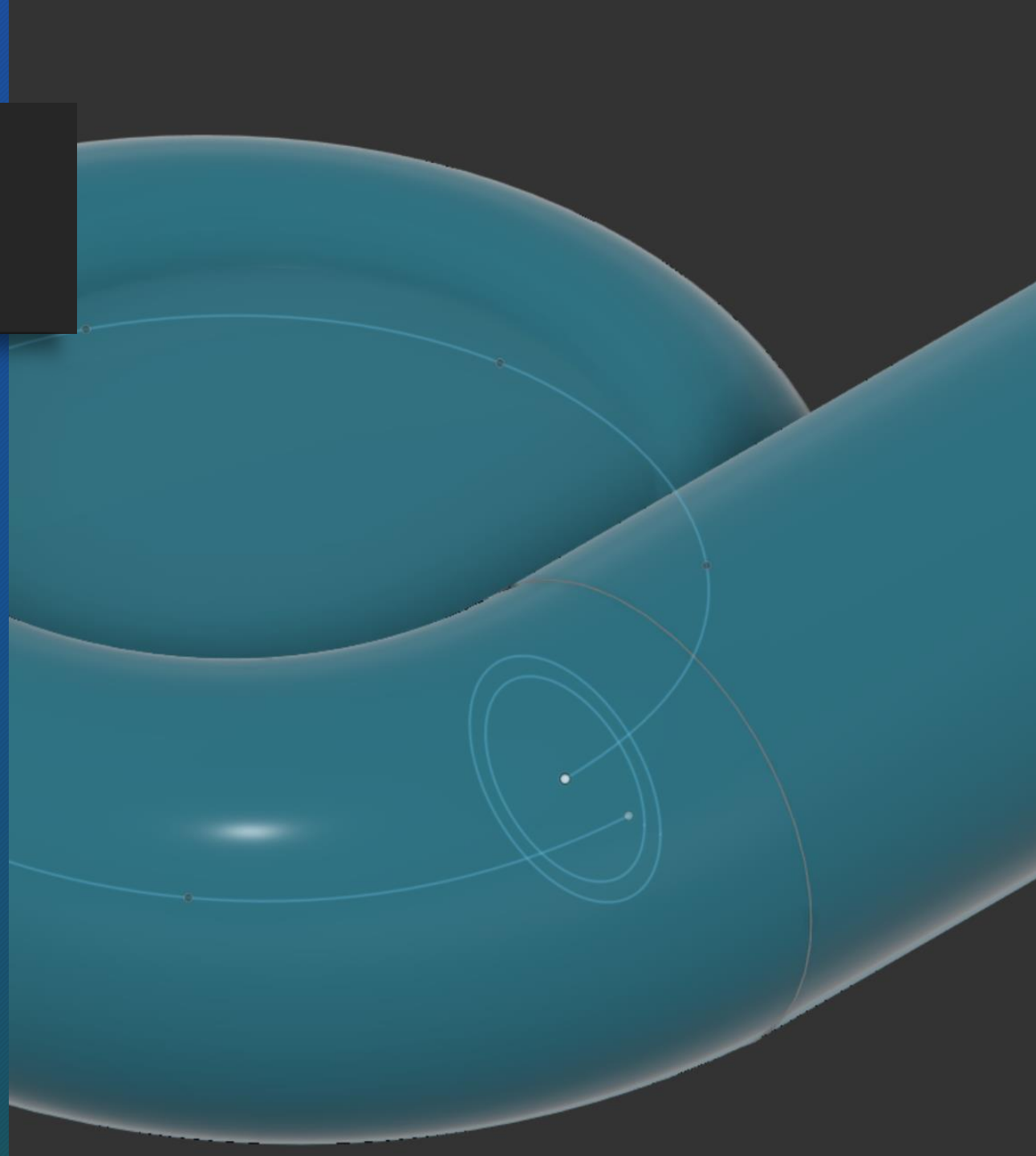
The Housekeeping II

- In the remaining lines, the main addition, as we have seen is the simple program logic. Much of what remains can be identified as the same steps you would take working in the skatech mode of Fusion 360
- In particular, the closing three statements should be left for now.
- We need to create a spline, unlike when working in the manual mode, you will need to define the points and package them to submit to the spline maker on line 32.
- Point generation is seen in the loop in line 29.

```
17      startRange = 0
18      # Start of range to be evaluated.
19      endRange = 2 * math.pi
20      # End of range to be evaluated.
21      splinePoints = 7
22      # Number of points that splines are generated.
23      rad = 5
24      i = 0
25      while i <= splinePoints:
26          t = startRange + ((endRange - startRange)/splinePoints) * i
27          xCoord = ((rad + t /(2*math.pi)) * math.cos(t))
28          yCoord = ((rad + t /(2*math.pi)) * math.sin(t))
29          points.add(adsk.core.Point3D.create(xCoord, yCoord, 0))
30          i = i + 1
31      # Generates the spline curve
32      spl = sketch.sketchCurves.sketchFittedSplines.add(points)
33  # Error handeling
34  except:
35      if ui:
36          ui.messageBox('Failed:\n{}'.format(traceback.format_exc()))
```

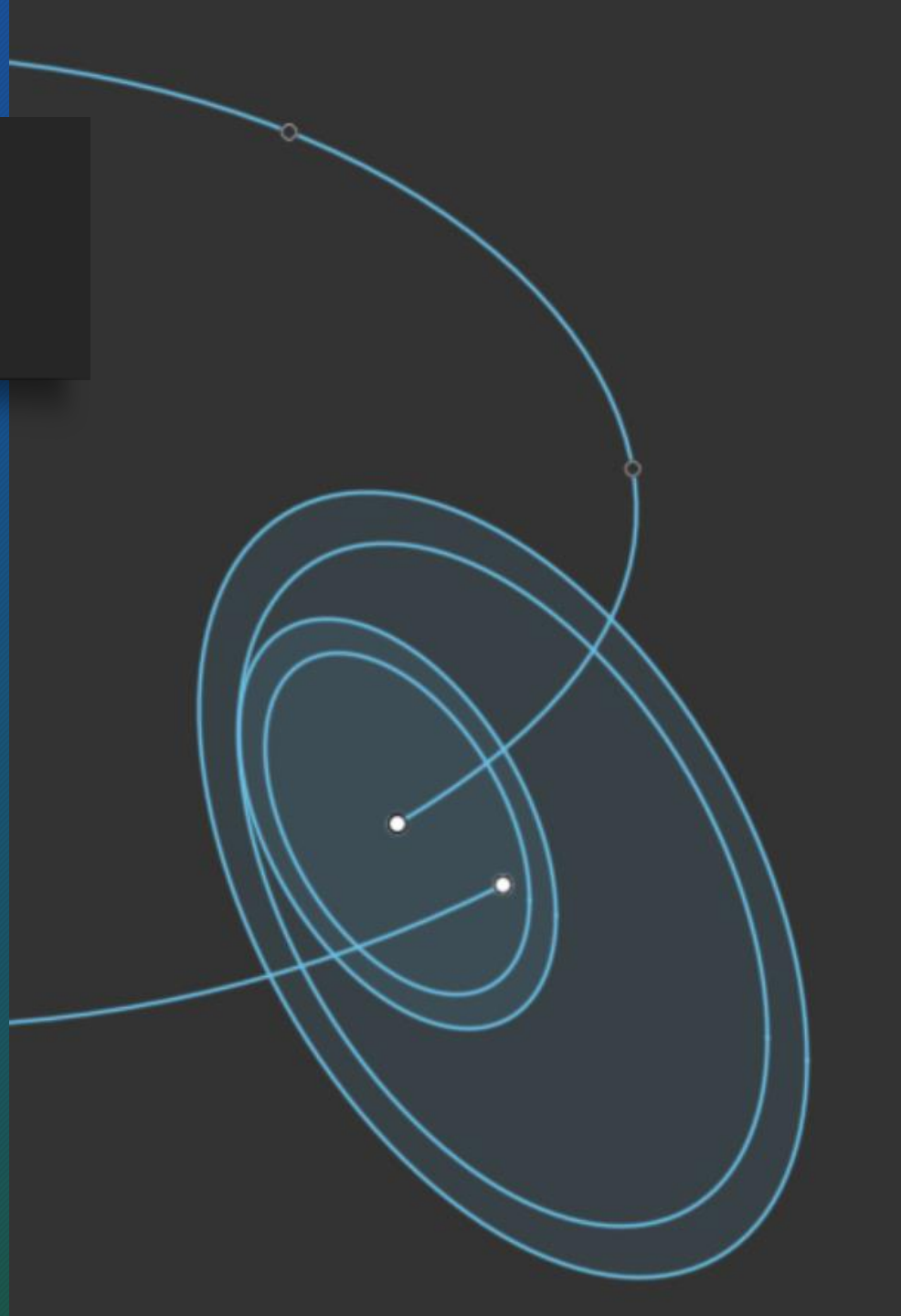
Creating a Volute

- We will use the spiral we have just created in making a volute programmatically.
- In order to understand the difference between working manually and using Python, let us observe the issue in the next step: Creating the lofting circles.
  - This is a simple step working manually. It is not difficult to do it in a program. However, note the subtle workflow difference.
  - It may be more efficient to go into two different planes rather than one to enable you select the loft profiles.



Creating Lofting Circles

- In the code fragment here, I created two planes to loft.
- Ordinarily, this could have been achieved in a single XZ plane, why am I using two?
- To answer the question, let us do it manually and see the problems that arise. You will want, whenever possible to avoid this situation in your program: More may sometimes be better than less.



Making Circles Programtically

- Here is the code that made the circles. They may look long, but remember, they are almost identical.
- It is better to have a clearer longer program than be smart and have a code laden with errors.
- Of course you can improve on this but play around and gain some confidence!

```
# Generates the spline curve
spl = symPlane.sketchCurves.sketchFittedSplines.add(splPoints)
# Create sketch, on XZ plane,
# And draw two center circles at the of the spline
begPlane = rootComp.sketches.add(rootComp.xZConstructionPlane)
sketchCirclesBeg = begPlane.sketchCurves.sketchCircles
centerPoint = adsk.core.Point3D.create(rad, 0, 0)
circle1 = sketchCirclesBeg.addByCenterRadius(centerPoint, rad/4)
circle2 = sketchCirclesBeg.addByCenterRadius(centerPoint, (rad/4)*(12/1(
# Get the profile defined by the two circles. That is the first item
#   in the sketches container
profBeg1A = begPlane.profiles.item(0)
profBeg1B = begPlane.profiles.item(1)
# Create another sketch, on the same XZ plane,
# And draw two center circles at the of the spline
# Find it easier to locate profiles this way
endPlane = rootComp.sketches.add(rootComp.xZConstructionPlane)
sketchCirclesEnd = endPlane.sketchCurves.sketchCircles
centerPoint = adsk.core.Point3D.create(rad*(12/10), 0, 0)
circle1 = sketchCirclesEnd.addByCenterRadius(centerPoint, rad/2)
circle2 = sketchCirclesEnd.addByCenterRadius(centerPoint, (rad/2)*(11.5,
# Get the profile defined by the two circles. That is the first item
#   in the sketches container
profEnd1A = endPlane.profiles.item(0)
profEnd1B = endPlane.profiles.item(1)
```

The Lofts