ArcGIS automation through scripting with Python

2_Interacting with Python

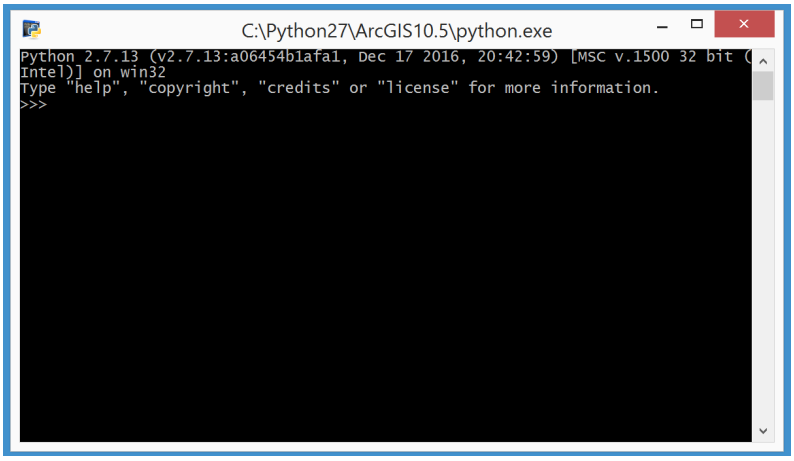
Step 1: Where and what is Python and how does it interact with ArcGIS?

If you have followed the default installation of Python, and not changed any paths during the installation of ArcGIS on your computer, then you will proceed through this with no issues at all. If changes have been made, then you will likely be able to locate it without any issue.

Locate Windows Explorer and navigate to the root directory of the local system disk (Usually drive C). You are looking for "Python27", and then click through into the directory, until you end up in a folder that looks like Figure 1. On some occasions you may see several folders each named "ArcGIS10.x", this usually is due to left over files when the system is upgraded. You should ensure that you locate the **correct** Python version that matches your ArcGIS version.

Once located, double click:  python.exe

This will open the Python Window:



Within here you can run any Python command you wish, try entering the commands (press return after each line):

```
print "Hellooooooo!"
print str(1+1)
1+3
```

Congratulations, you have run some Python code! But what about arcpy?

Enter the package:

```
import arcpy
```

This tells the Python command prompt that we want to make the entirety of the arcpy package available (usually takes a couple of seconds to run). Once the prompt returns, enter:

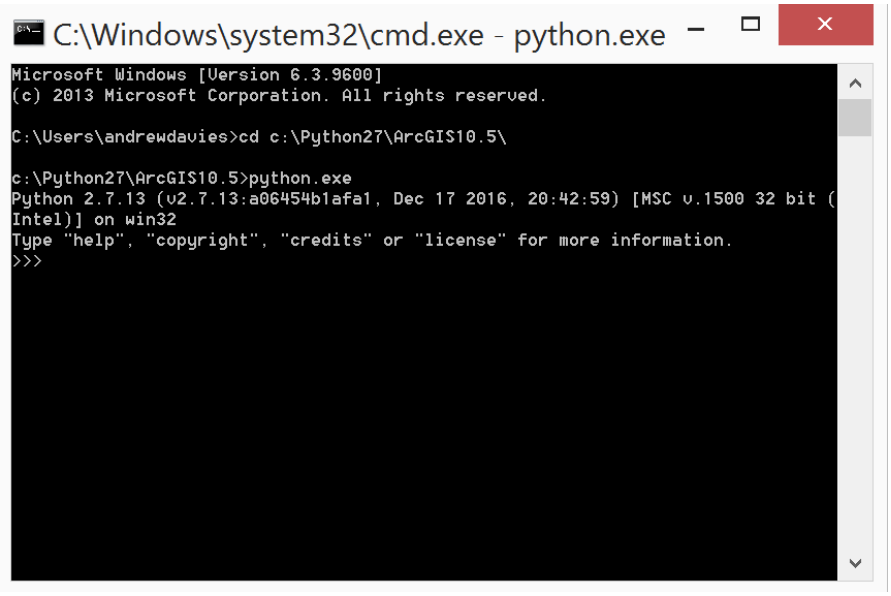
```
arcpy.AddMessage("Hello!")
```

AddMessage is an arcpy command that essentially works as “print”, but interfaces with ArcGIS in a more detailed way.

Step 2: Running from the Windows Command Line

Another method to achieve the same result as the above is to use the Windows Command Prompt, to access this. First, load the Command Prompt, you can usually find it in the Windows Application Menu, or by Windows Key + R, typing “cmd” (no quotes).

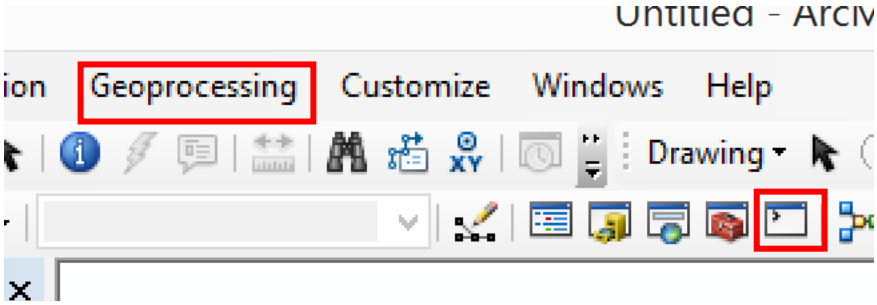
Duplicate steps shown in the screenshot below to execute the Python.exe file and bring up the Python command prompt (>>>):



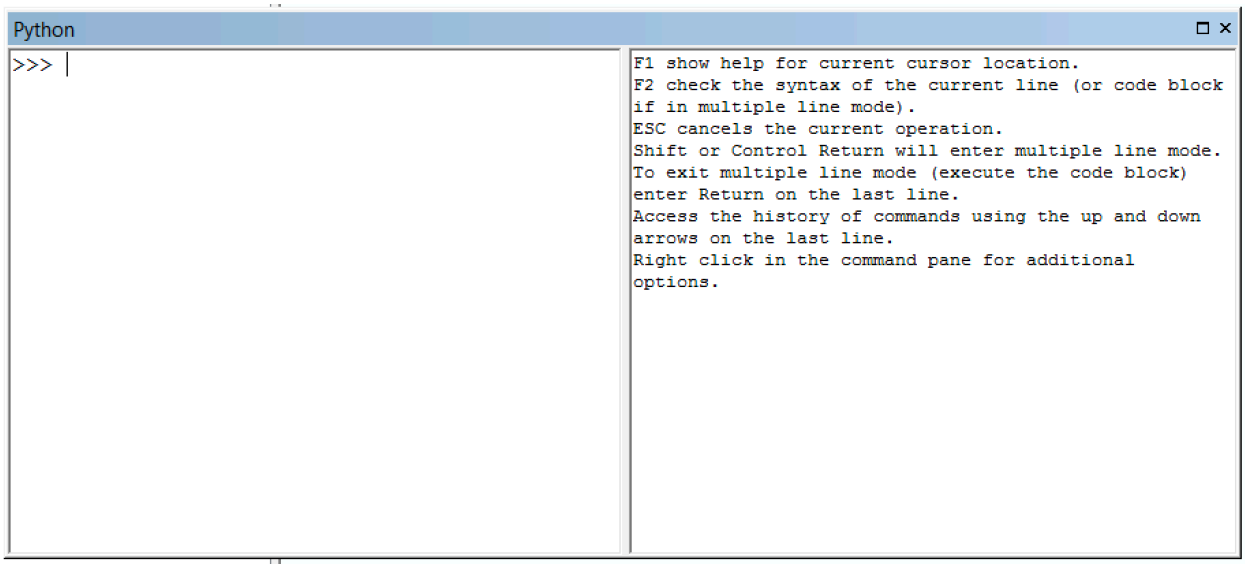
Whilst useful to know, you will probably never interact with Python/arcpy in this manner. It is unwieldy, provides no assistance in terms of syntax highlighting and so on.

Step 3: Running from the Geoprocessing Window

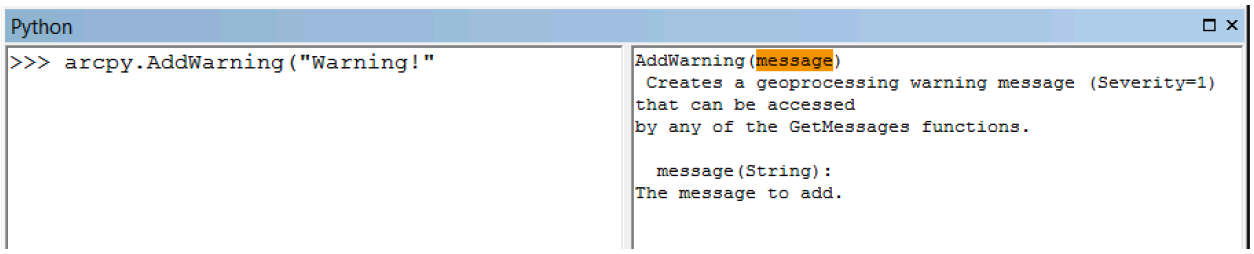
You are far more likely to use Python by interacting with it through ArcGIS Desktop, particularly ArcMap. In this step, bring up an ArcMap window. You can access the "Python" window by either clicking the small icon (see below image), or heading to the drop down "Geoprocessing" and then "Python".

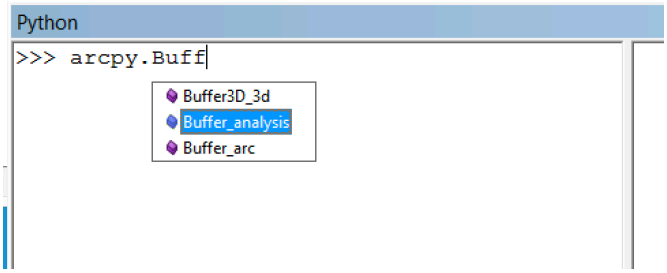


This window is far more intuitive than using the previous two options:



Try it out with a few commands, note how the Python window gives you a help pane (on the right), and auto-completion of the tool name:





Try making an empty shapefile:

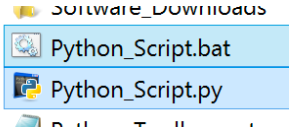
```
arcpy.CreateFeatureclass_management("D:\Python-Class",
"shapefile.shp","POINT")
```

Next, why not try using some other tools that you may have used in the past?

Step 4: Clicking a *.py file

This is one of the more common methods of executing a Python script that is linked to an ArcGIS operation. It allows you to create a fully coded file, that you can run in one go, without ever opening ArcMap.

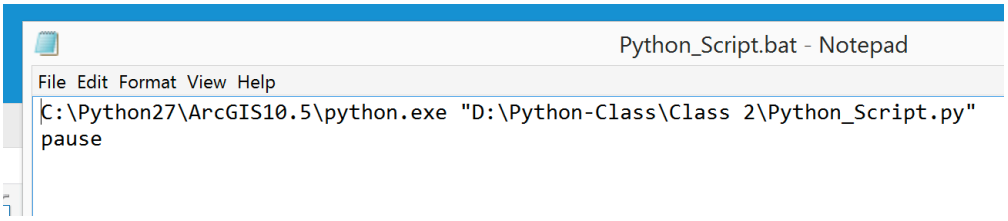
On Github, I prepared a few files within the *Classes > 2_Interacting* folder, download these and store them into a known place:



Try executing Python_Script.py by double clicking on the file.

The result will be a black window that automatically closes, the script has either completed successfully or has run into a problem. But you won't ever know, as it closes so fast you can't see it. This is also an issue if you are relying on any error messages or print statements to debug your code.

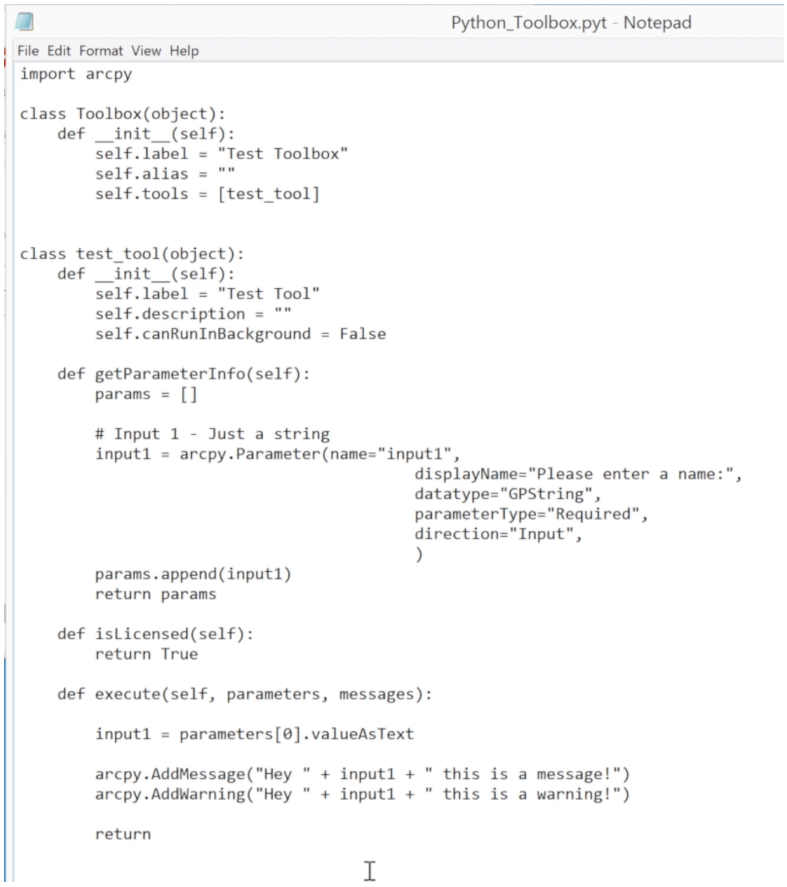
The way around this is to execute the Python_Script.py file via a Bat file, which will call the Python.exe executable and tell it to run the Python_Script.py file. By adding a pause argument on a second line in the Bat file, you will tell Command Prompt not to close the window:



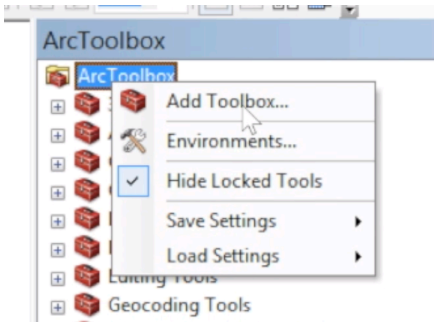
If you try to run the Bat file provided it may not complete if you are storing your files in a different location to me, so edit the file (Right click > Edit) and adjust the location of the script or the Python.exe executable.

Step 5: Toolboxes (*.pyt)

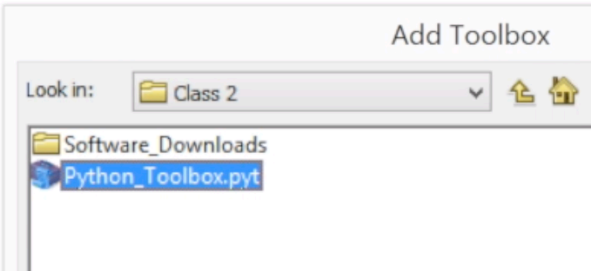
If you eventually want to disseminate your code to non-specialist GIS users, you may want to consider using a Python Toolbox. This can be either be a single file (or several files, e.g. individual tools as *.py files) that use a *.pyt file extension. See below for an example structure of a Python toolbox (don't worry though, we will go through this in detail during a later session):



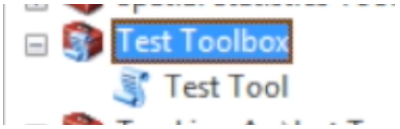
On Github, I prepared an example toolbox within the *Classes > 2_Interacting* folder, download the file named Python_Toolbox.pyt and store it into a known place. To add the toolbox to ArcMap, open ArcToolbox and right click at the very top, selecting "Add Toolbox":



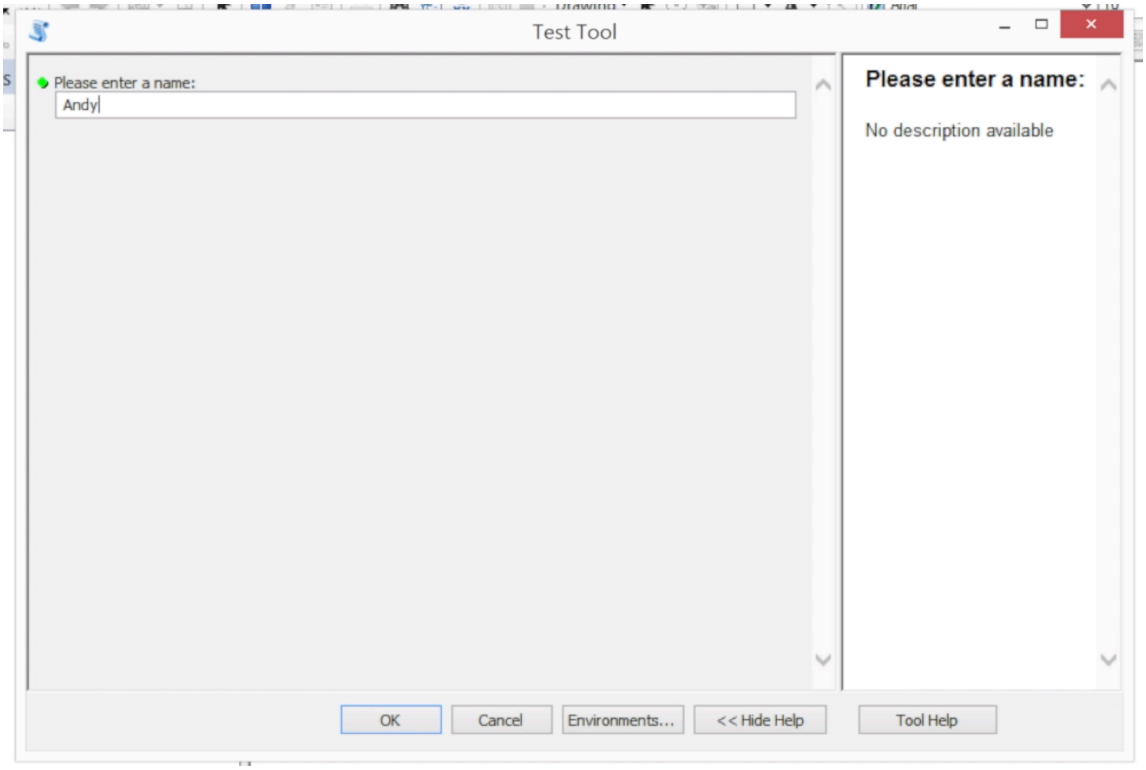
Navigate to the saved toolbox location, noting the icon style (red toolbox with a scroll), if you see a red x on there, it means you have a code error with your toolbox:



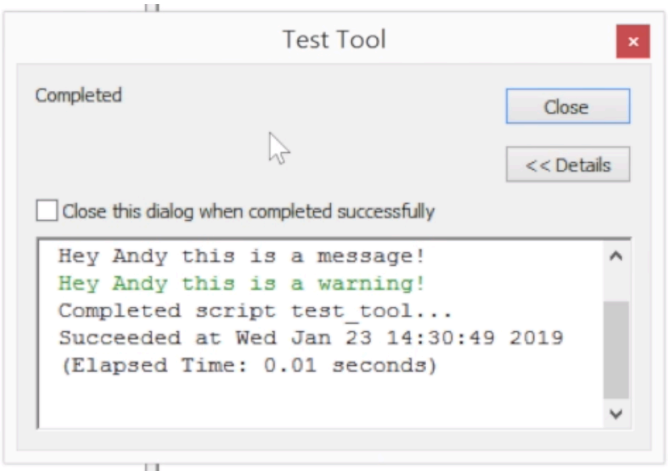
Upon successfully adding your toolbox, you should see the below within ArcToolbox. Refer back to the code, and note how the toolbox is constructed using classes and functions:



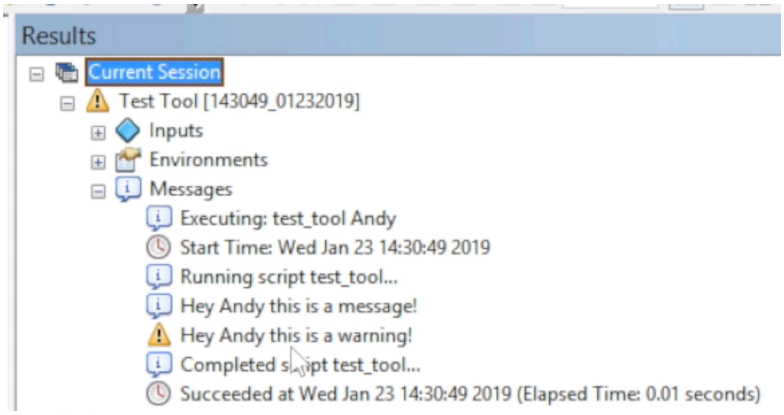
Now execute the "Test Tool", and you will see a simplistic one field entry, all this does is take a string and print it in two different outputs. Very simple, but far more complexity can be built into a toolbox than this, and there is a myriad of different functions, datatypes and coding practices that can be used.



I explicitly told “Test Tool” to run in the foreground, so you will see this dialog:



You can also review the results of the tool in “Results”, viewable by selecting that option on the Geoprocessing menu dropdown:



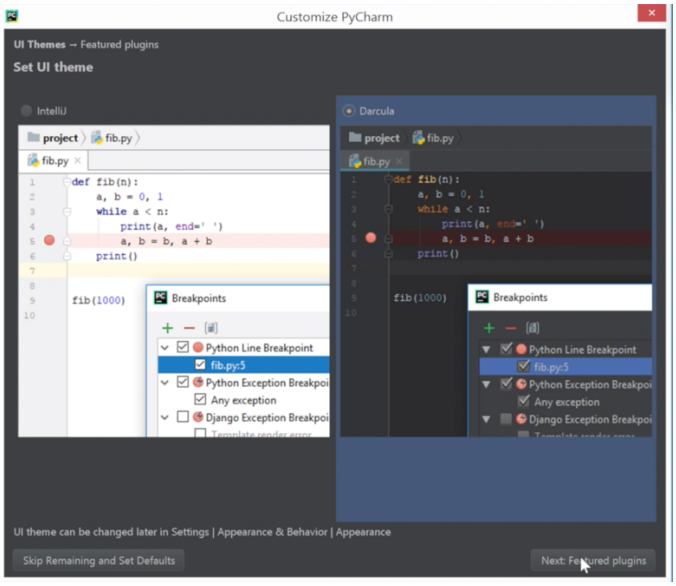
Step 6: PyCharm

My favoured method of interacting with Python and arcpy for development, revising and executing code is to use the IDE (integrated development environment) ***PyCharm***. In this section, I take you through the "portable" version of PyCharm (this is a version that you can run from a USB stick, or on a computer where you don't have installation rights). However, if you are using your own computer, you can install as normal and I would suggest downloading it from: https://www.jetbrains.com/pycharm/download/, you can use the "Community Edition", but if you want additional features, you can download the "Professional Edition" and apply for a free license (https://www.jetbrains.com/student/).

First download the Portable versions of Atom (not covered here, but is a nice text editor to replace Notepad), PyCharm_Portable.exe and Git_Portable.exe. These are "portableapps", so you need to execute them and install them on your USB drive, or in a location that you have write access to.

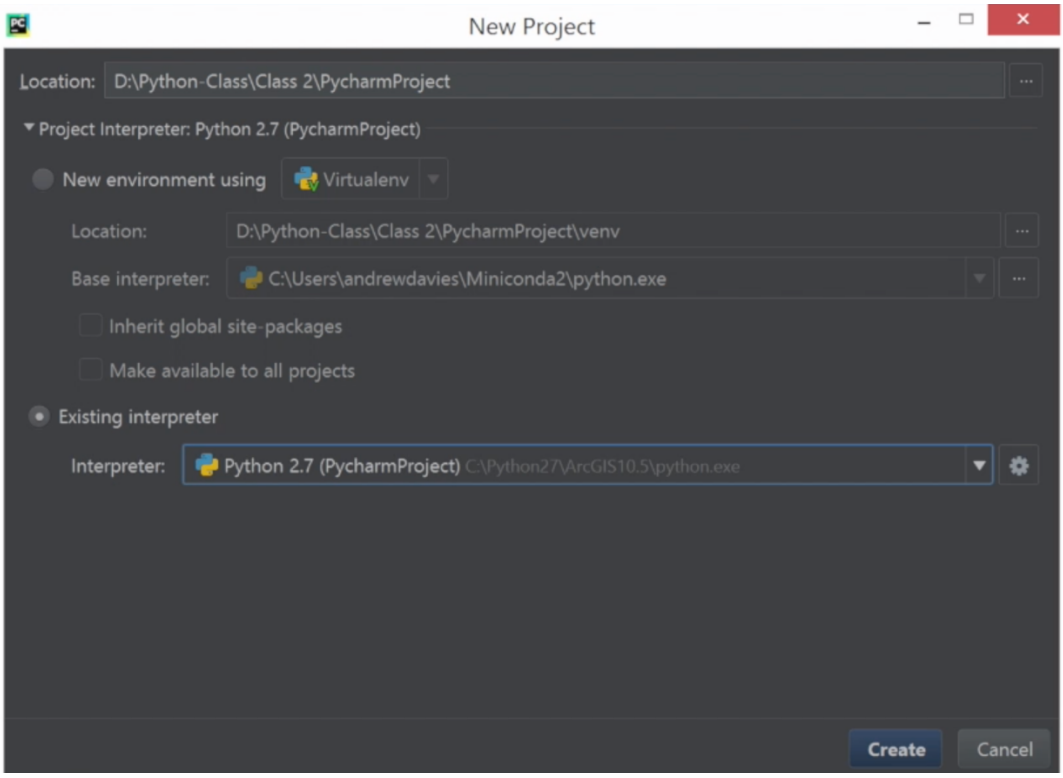
Locate and execute the PyCharm executable:

You will be faced with a short setup wizard (I usually just select Dark theme and skip the rest).



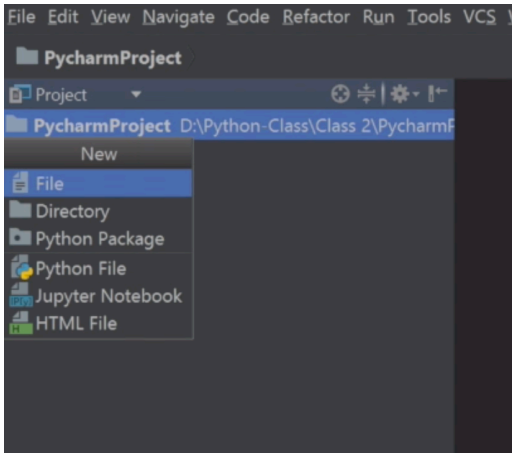
Next is to set up a “New Project”, and from there we will link PyCharm to the ArcGIS10.x python installation:

You need to set up the “Project Interpreter” to use an “Existing Interpreter” and we will point that to the ‘*/Python27/ArcGIS10.x/python.exe*’ location:



Once you click create, we are in for a bit of a wait, as we wait for the system to “Index” the Python installation to build an understanding of the packages that are held within, this really helps with our development, as it gives some autocomplete and context driven help options (similar to the Geoprocessing/Python window in ArcMap):

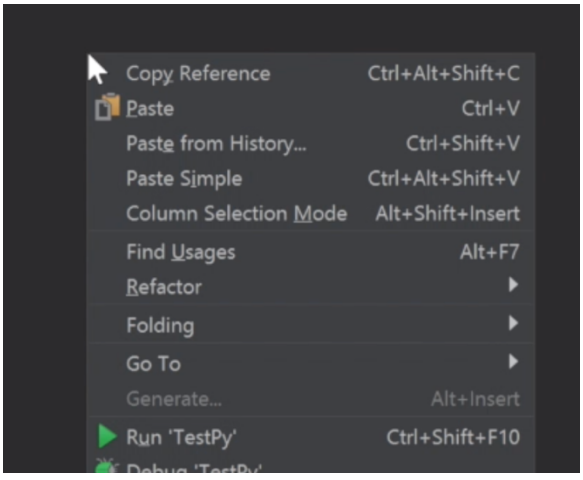
Once the indexing is complete, create a new empty Python file within your project:



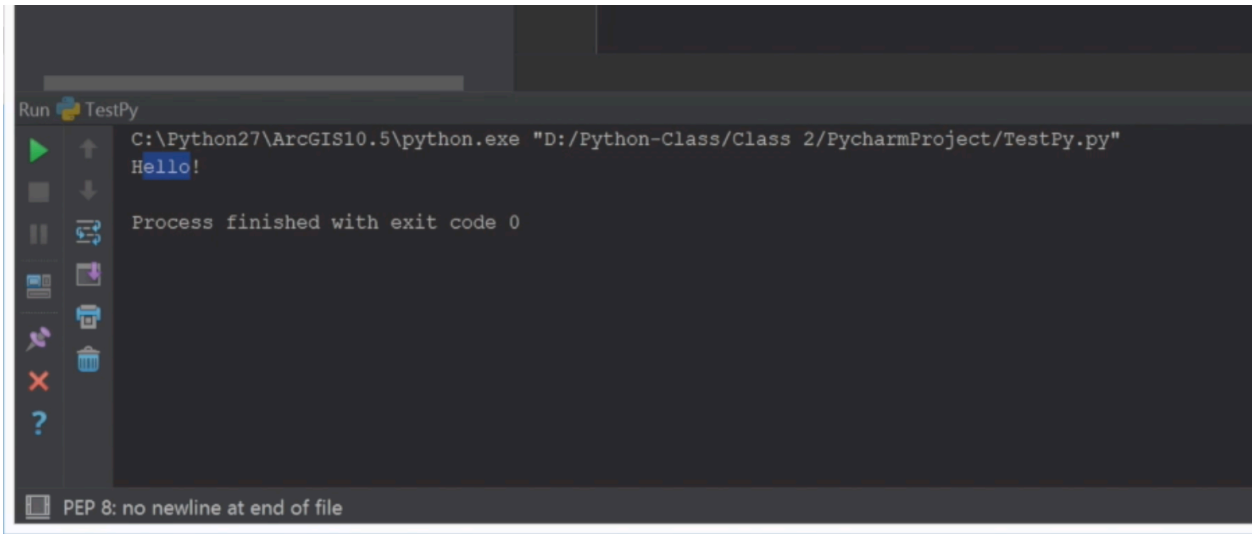
Try adding some example code:

```
print "Hellooooooo!"
print str(1+1)
1+3
```

Then right click within the code editor window, select Run, or use the shortcut Ctrl + Shift + F10:



The “Run” window will appear at the bottom of the IDE, showing you some responses from your code:

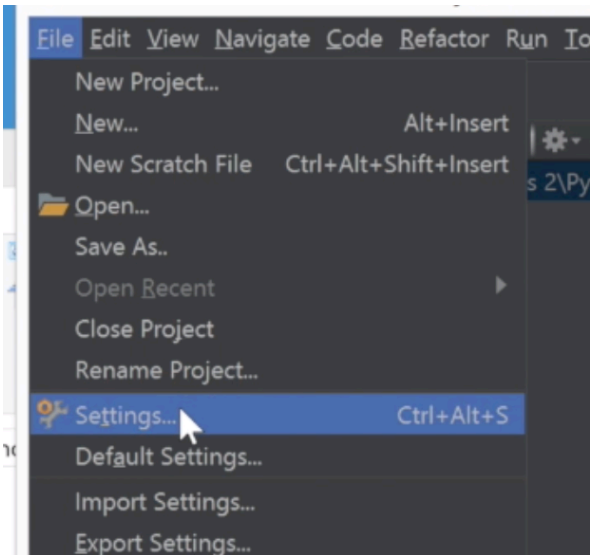


Next try some other commands!

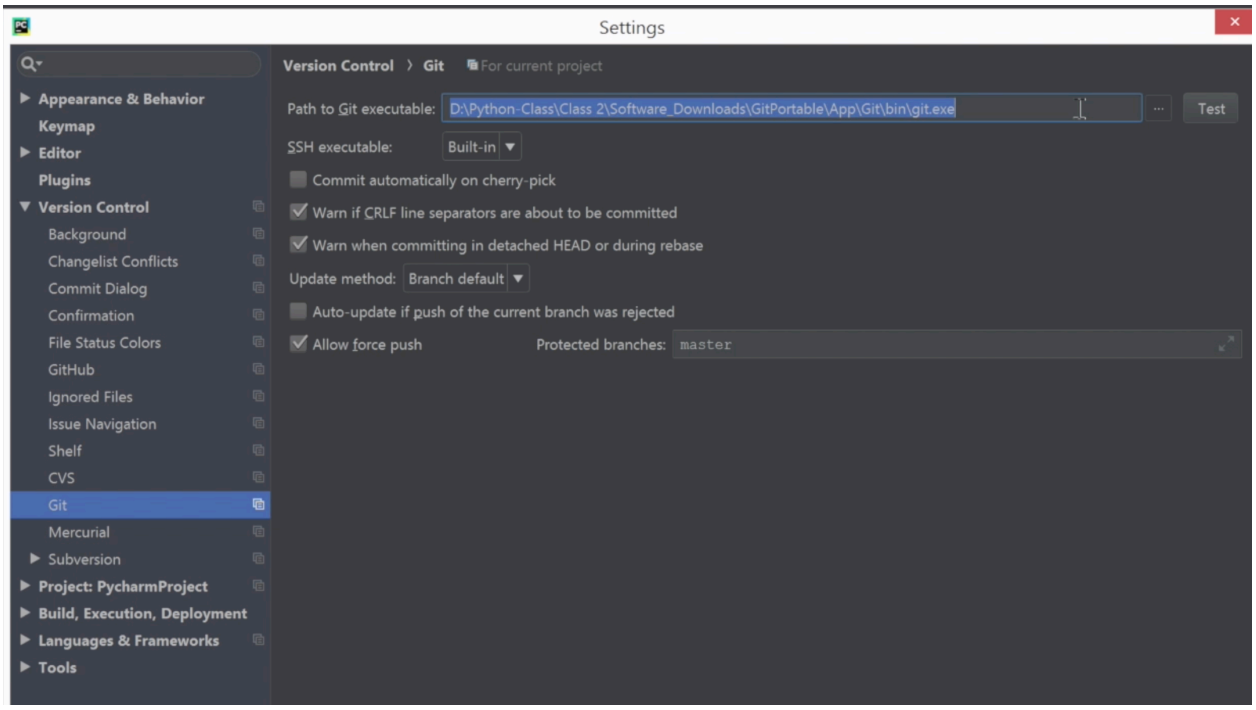
```
import arcpy
arcpy.CreateFeatureclass_management("D:\Python-Class",
"shapefile2.shp","POINT")
```

That's basically it, but there is one final step that we would benefit from doing, and that is adding Git and Github functionality into PyCharm, this means you can download repo's and commit directly from within PyCharm! Be great for our coding challenges.

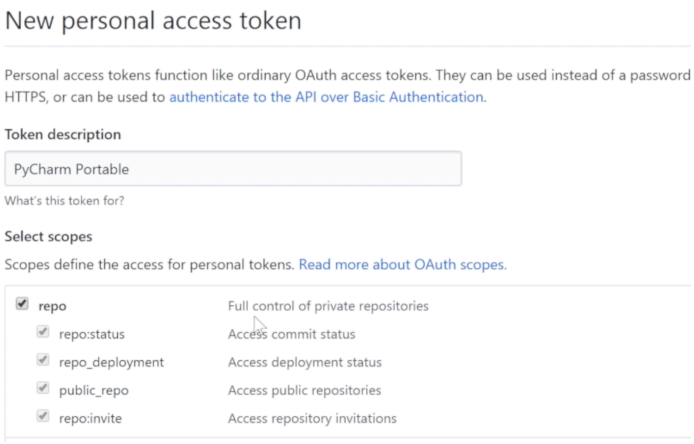
Go to Settings from File > Settings:



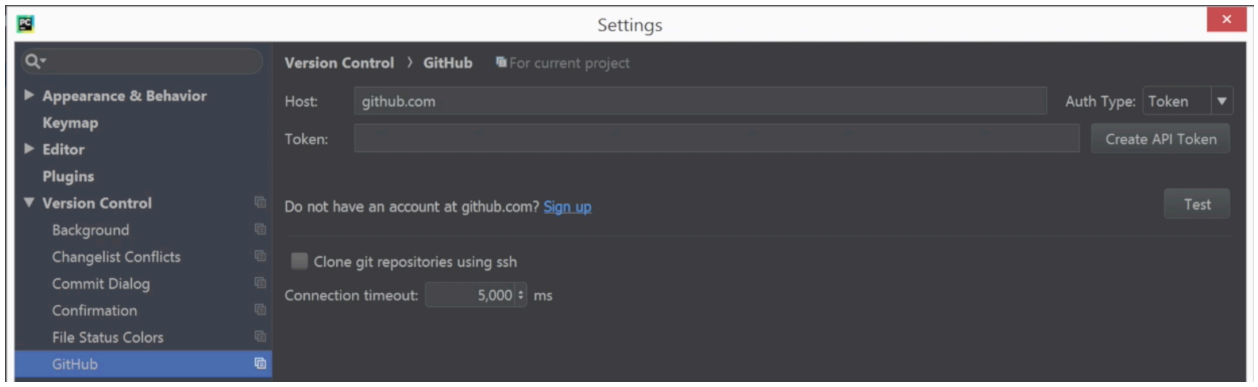
Locate “Version Control” > “Git” and enter the location of the Git Portable file that, note that you want to locate the version within the subdirectory */bin/git.exe, not git_bash.exe etc, click test, if you get a success message you are good!



Next add Github credentials, you need to use a “Personal Access Token” to link it, and you will of course need a Github.com account (remember to verify email first). You can generate an access token within Github (first log in, then “Settings” (for your account, not your repo), and then Developer Tools, and Personal Access Tokens, select repo and give it a description:



Copy the resulting access token to your PyCharm installation within the Token field, then click “Test”, if you are successful then awesome!



WARNING! Never share your personal access token with anyone, they can cause damage to your account by using it.

Also take normal care with your Github account, enable Two Factor Authentication to protect yourself and your code!