Exercise: Use a feature class list for geoprocessing

In this course, you will use Python to create treatment areas for invasive plant species within the San Juan National Forest in the state of Colorado. Part of this process will be to define areas in which chemical and non-chemical treatments may be used within the forest.

In this exercise, you will begin to build your treatment areas by creating polygons around lakes and streams. These polygons will define the non-chemical treatment areas.

## Step 1: Explore San Juan data

In this step, you will become familiar with the San Juan National Forest data that you will work with in this course.

* In the Catalog window, browse to your ..\Student\PythonGP10\_0 folder and open SanJuan.mxd.



Your map shows a portion of the San Juan National Forest and some of the surrounding communities. Notice the location of the invasive plants and their proximity to roads and water features. You will use Python to create buffer zones around water bodies to determine the extent where treatment is needed in the invasive plant areas.

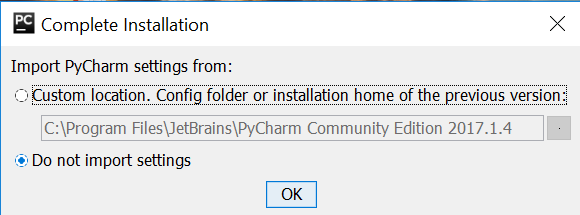
You will come back to ArcMap at the end of this exercise to view the results of your Python script. For now, minimize your ArcMap window.

## Step 4: Set up Python Editor (PyCharm)

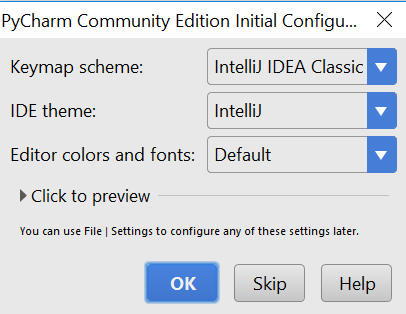
In this step, you will begin your Python script. For the exercises in this course, you will be using PyCharm for your IDE. You will begin your script by setting two geoprocessing environments.

Click the Start menu, find JetBrains PyCharm Community Edition. Double Click to open it.

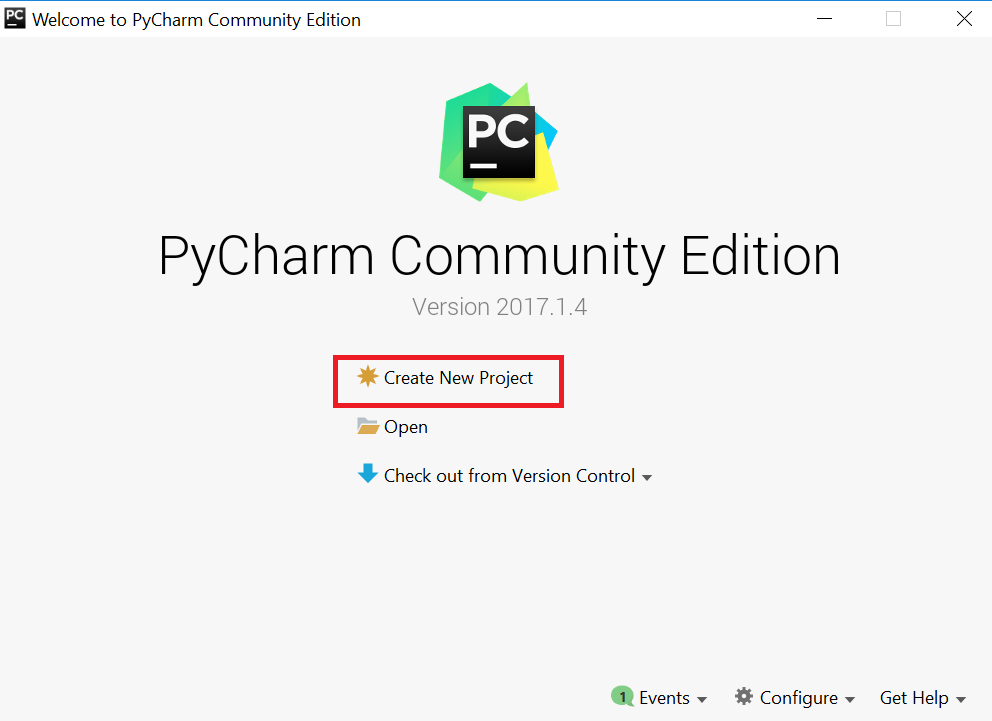
Check “Do not import settings” and OK.



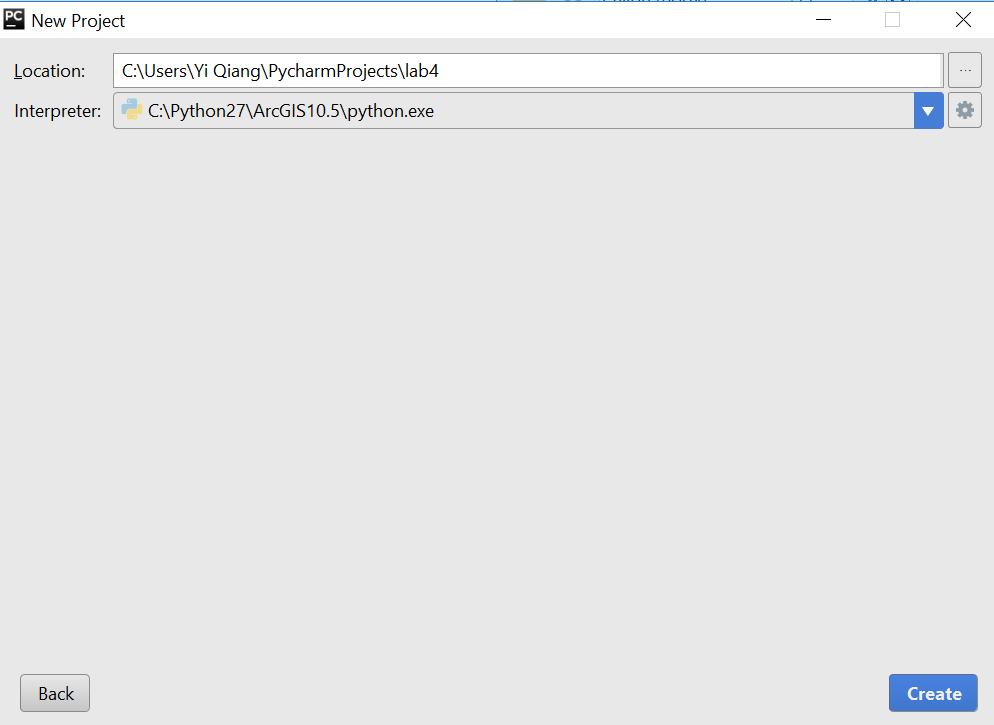
Click OK for the following dialog.



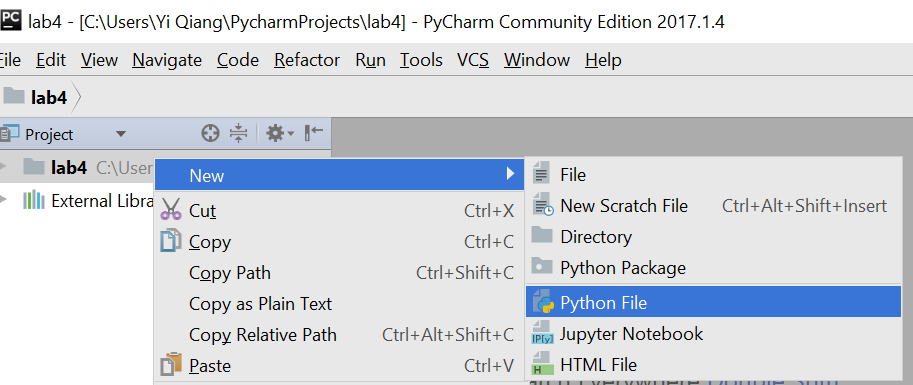
Create a new project



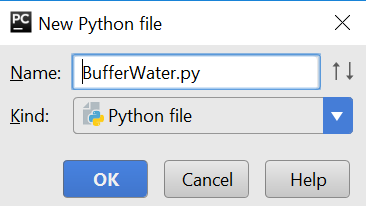
Save the project in your data folder and name the project **lab4**, click Create.



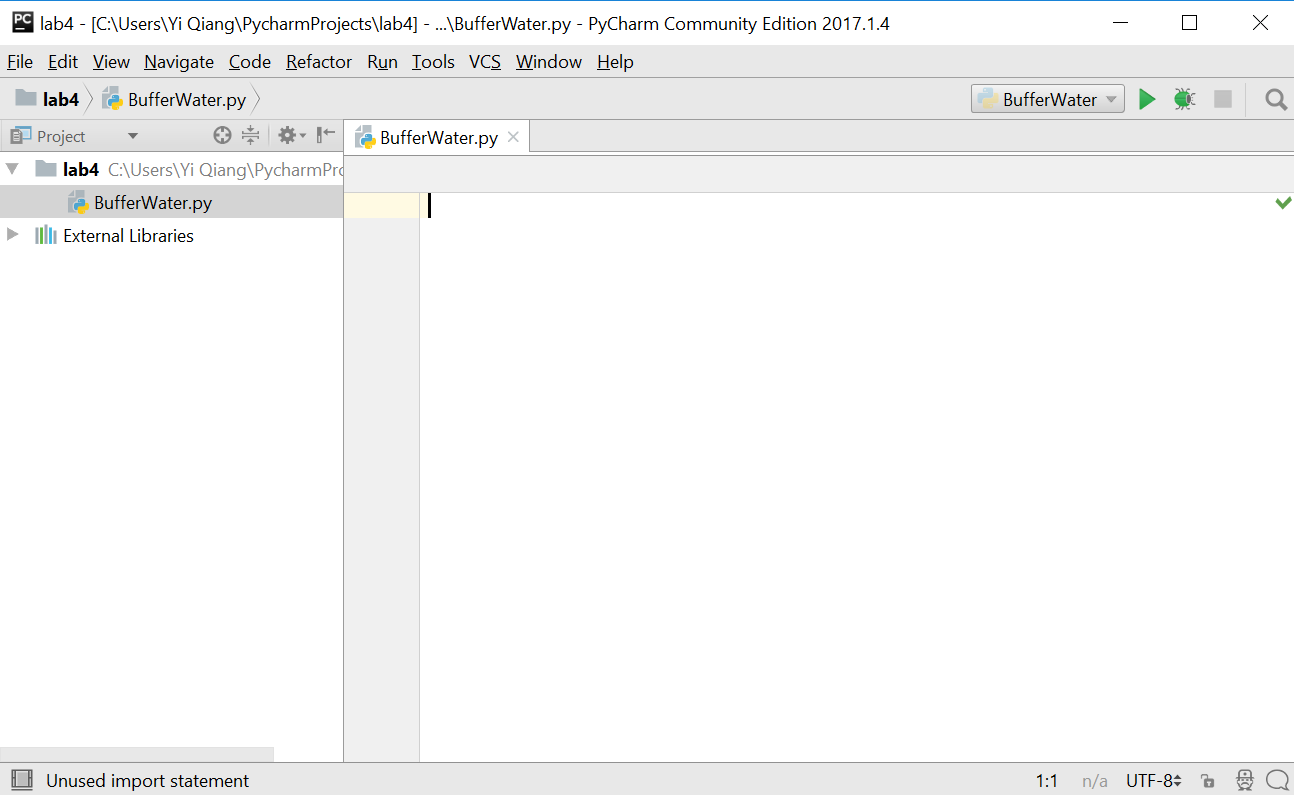
Close Tips dialog if it appears. Right click on the **lab4** project -> New -> Python File



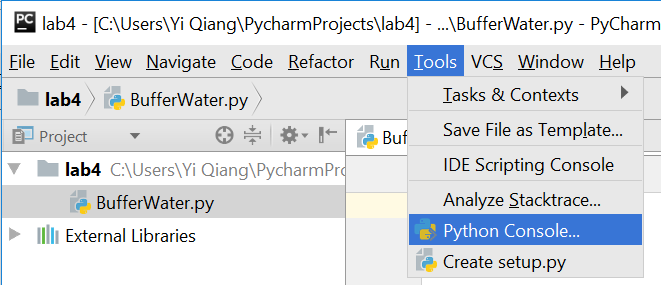
Name the new file as BufferWater.py



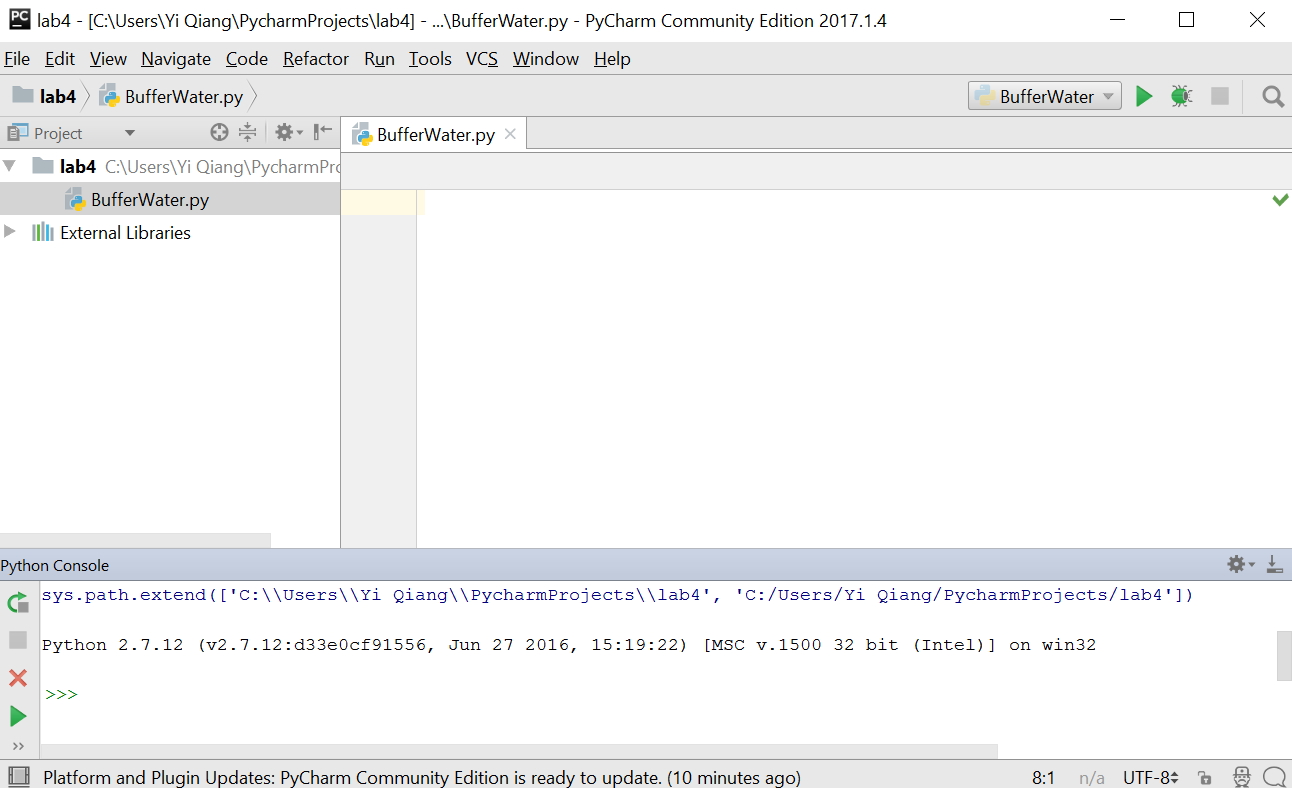
Now you will see an empty editor, where you can input script



Click the **Tool** menu -> Python Console.. to open the Python console under the editor.



Now, your PyCharm interface should look like:



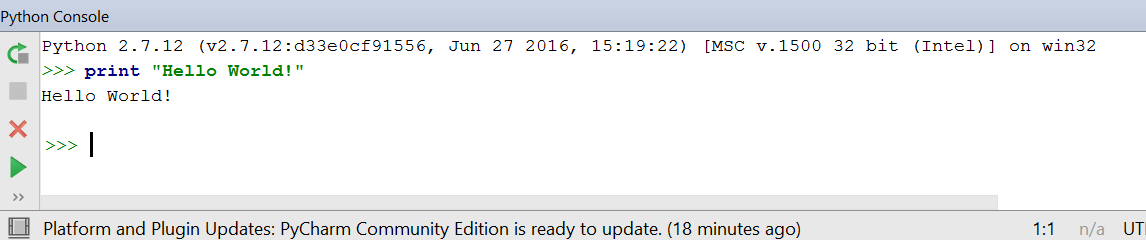
## Step 5: Set geoprocessing environments

In this step, you will set two geoprocessing environments.

Prior to entering any Python code, you will import ArcPy in the Interactive Window in PythonWin. This window is displayed below your BufferWater.py script window. By importing ArcPy first, you will make PythonWin aware of the ArcPy libraries. This will allow you to use the drop-down lists of ArcPy functions as you create your script.

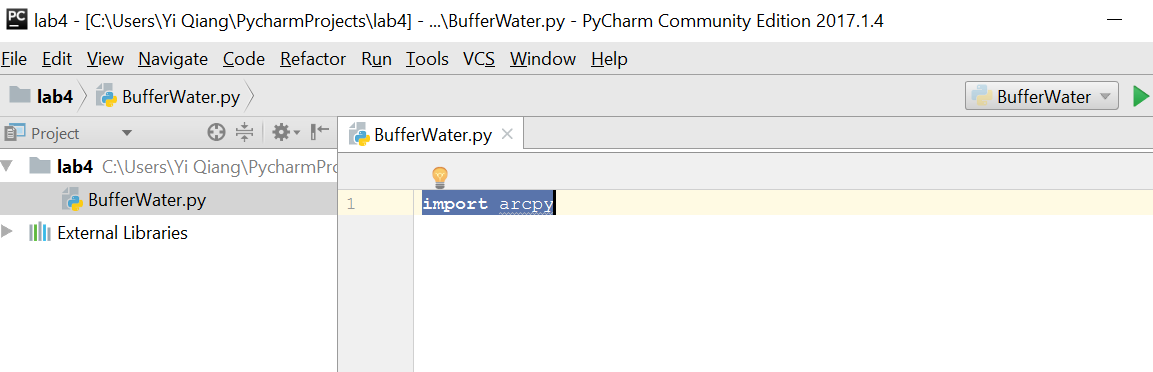
In the Python Console, type **print “Hello World”** and press Enter. The string will be printed immediately. You can try to type other Python script (2 + 3, or 10//3) to see the reaction.

Note: The Python console is like a command terminal, where you can type a script and press Enter to execute it. The output of the script will be shown immediately. The Python console is good for testing single line of script, but not efficient for writing multiple lines of scripts.



Now, we go back to the Editor, and type **import arcpy**. Because you are writing and running your script outside of ArcGIS, you need to include this statement to access ArcGIS Python functionality.

Note Python is case-sensitive. Be careful for the capitalization of scripts.

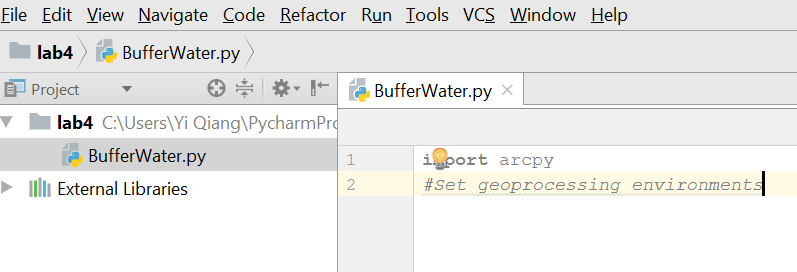


Next, you will enter your geoprocessing workspace.

Documentation is very important when writing scripts, especially if you wish to share your scripts with others who may not be familiar with your code. All text behind the **#** sign is documentation, which will not run as Python scripts.

* On line 2, enter the following comment

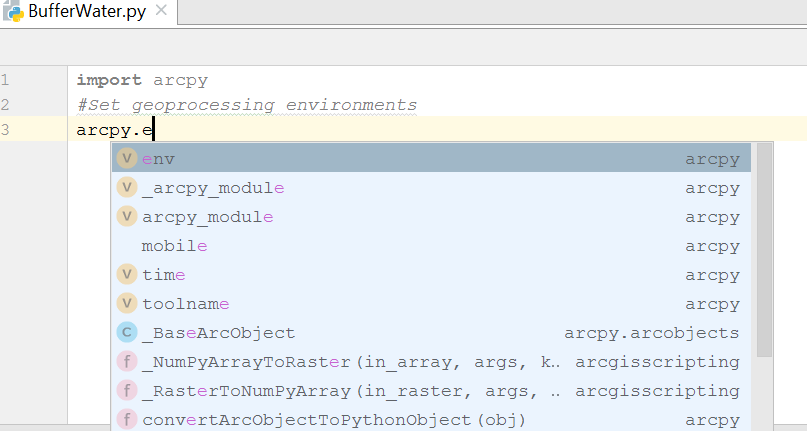
#Set geoprocessing environments



* On line 3, type arcpy.e.

Notice the drop-down list of arcpy functions.

* With your mouse or arrow keys, select env from the list.



* Continue your statement by typing a dot (.) then the workspace function.
* Set your workspace equal to the geodatabase path, C: /PythonGP10\_0/Data/SanJuan.gdb.

Note: If you saved the course data to a location other than the default, set your workspace to

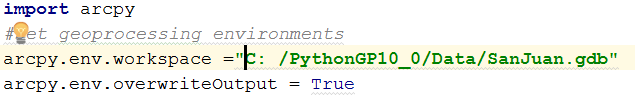
match the location of your data.

* Paths in Python are string values, so make sure to enclose your path in quotation marks.

Next, you will enter one more environment setting.

* On line 4 of your script, type the following code:

arcpy.env.overwriteOutput = True



The overwriteOutput parameter controls whether tools will automatically overwrite any existing output when your script is run. When set to True, tools will execute and overwrite the output dataset. When set to False, existing outputs will not be overwritten, and the tool will return an error.

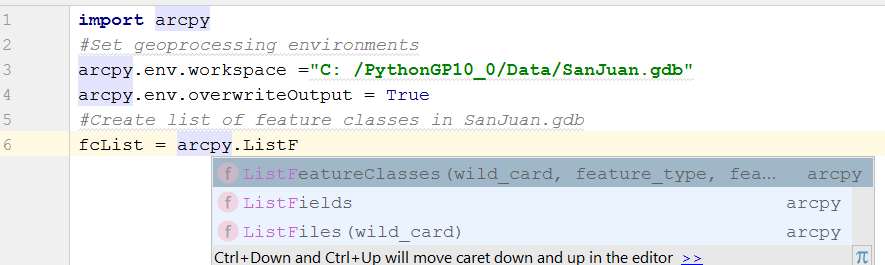
Note: False and True are Boolean data type, not string

* Add the following comment:

#Create list of feature classes in SanJuan.gdb

* Online 6 type:

fcList = arcpy.List



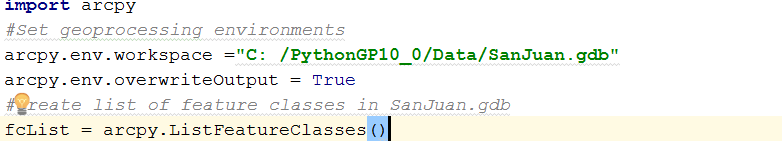
* From the list, select and add ListFeatureClasses to your line of code.

Note: the syntax of the function is shown in the parameters: ListFeatureClasses include variables such as wild\_card, feature\_type, and more.

The wildcard variable allows you to filter feature classes based on their name. For example, to return just the feature classes ending with the string Anno in their name, you would provide "\*Anno" as the wildcard value. You can also return just the feature classes of a specific geometry type by providing a keyword for the feature\_type parameter. The feature\_dataset parameter allows you to constrain your list to a specific feature data set in your geodatabase.

You will list all of the feature classes in your SanJuan geodatabase.

* Complete your line of code by returning all the feature classes to your fcList variable.



# Step 7: Loop through your list of feature classes

Now that you have created a list of feature classes, you will create a loop to iterate through your list.

Your loop will do the following:

* Assign each feature class in fcList to a variable named fc.
* Check the name of each feature class.
* If the name is Lakes or Streams, then the Buffer geoprocessing tool will be run.
  + Write a comment documenting the next part of your script.

#Create a loop to buffer Lakes and Streams

Before you begin the loop, you will initialize a variable named bufferList. Each time a feature class is buffered, the name of the feature class will be added to the Python list. This list will be used at the end of your script when you union your buffered feature classes together.

Enter the following code:

* bufferList = []

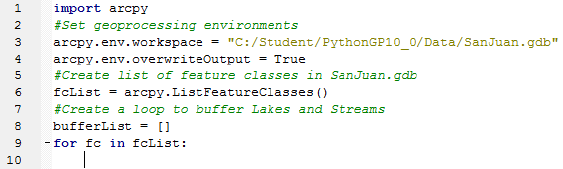
This will create a new, empty Python list.

Now you will create a for-loop to iterate through each feature class stored in fcList

* Write the Python for-loop to assign each feature class in fcList to a variable named fc.

for fc in fcList:

* Make sure to end your loop with a colon “:”.
  + Press Enter and notice that the next line is automatically indented by PyCharm.



Note: Indentation is very important in Python. Standard indentation is four spaces. However, you can use more or less than this, as long as your indentation is consistent.

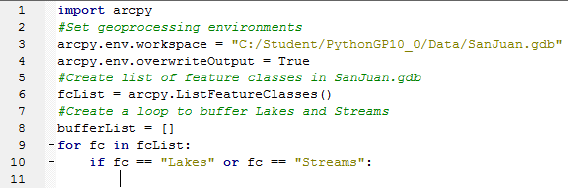
Next, you will write an if statement to determine if the name of your feature class is Lakes or Streams.

* At your indented cursor location, write the following if statement.

if fc == "Lakes" or fc == "Streams":

* Once again, make sure to end your statement with a colon.

Note the or between the two conditions. It is a logical operator that returns True if either condition is True.



## Step 8: Buffer lakes and streams

Now you are ready to write the code to create the buffer polygons around the Lakes and Streams.

Before writing the Buffer functions, you may need to know its syntax (what are the input and output). You can use arcpy.Usage(“FUNCTION NAME”) to display the syntax of the function.

* You can type the following code in the Python Console, and press Enter

Import arcpy

arcpy.Usage("Buffer\_analysis")

Note: Geoprocessing tools in ArcToolbox are functions in ArcPy. In Python terms, you are adding the Buffer function to your script.

* Enter a left parenthesis to display the usage for the Buffer function.
* To display a more permanent usage for the Buffer function, in the lower Interactive Window, enter the following code:
* arcpy.Usage("Buffer\_analysis")