# Part 1: 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 get familiar with the San Juan National Forest data that you will work with in this course.

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



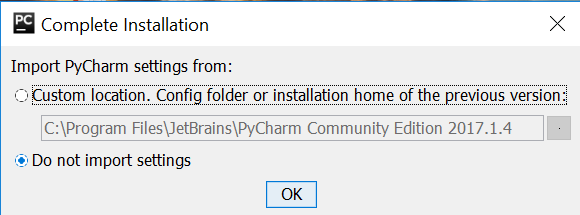
Your map shows a portion of the San Juan National Forest and some of the surrounding communities. Notice the location of the invasive plants and their proximity to roads and water features. You will use Python to create buffer zones around water bodies to determine the extent where treatment is needed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

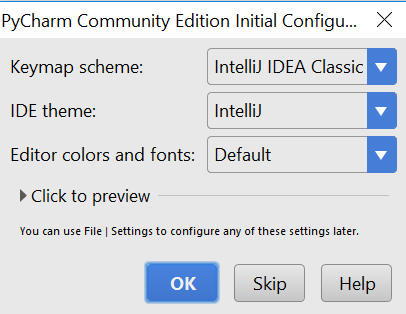
## Step 4: Set up Python Editor (PyCharm)

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

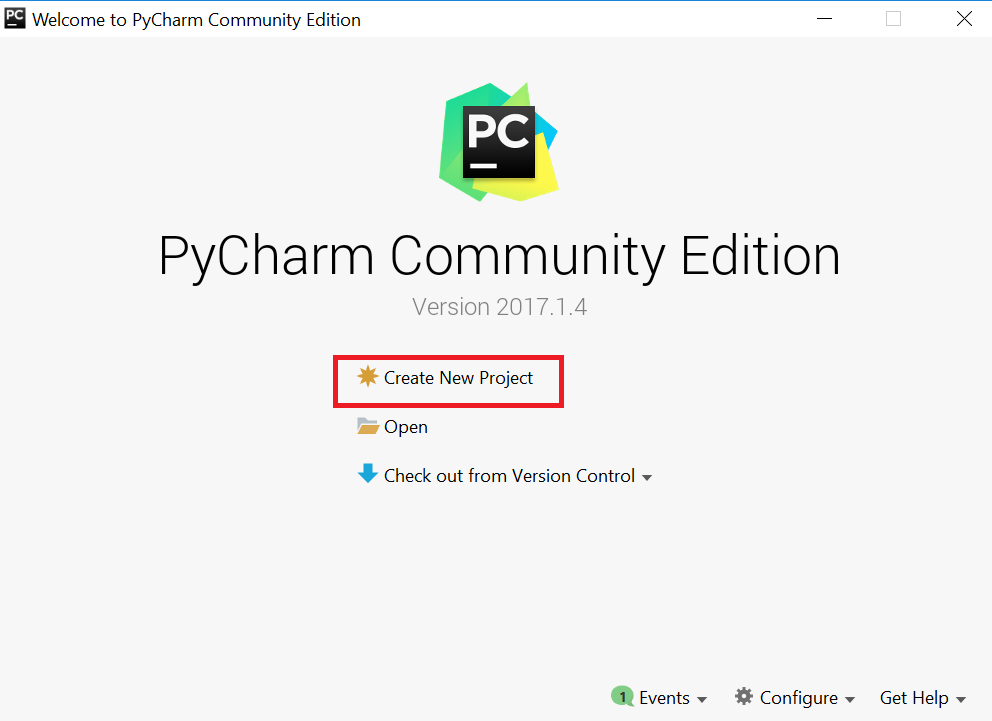
* Click the Start menu, find JetBrains PyCharm Community Edition. Double Click to open it.
* Check “Do not import settings” and OK.



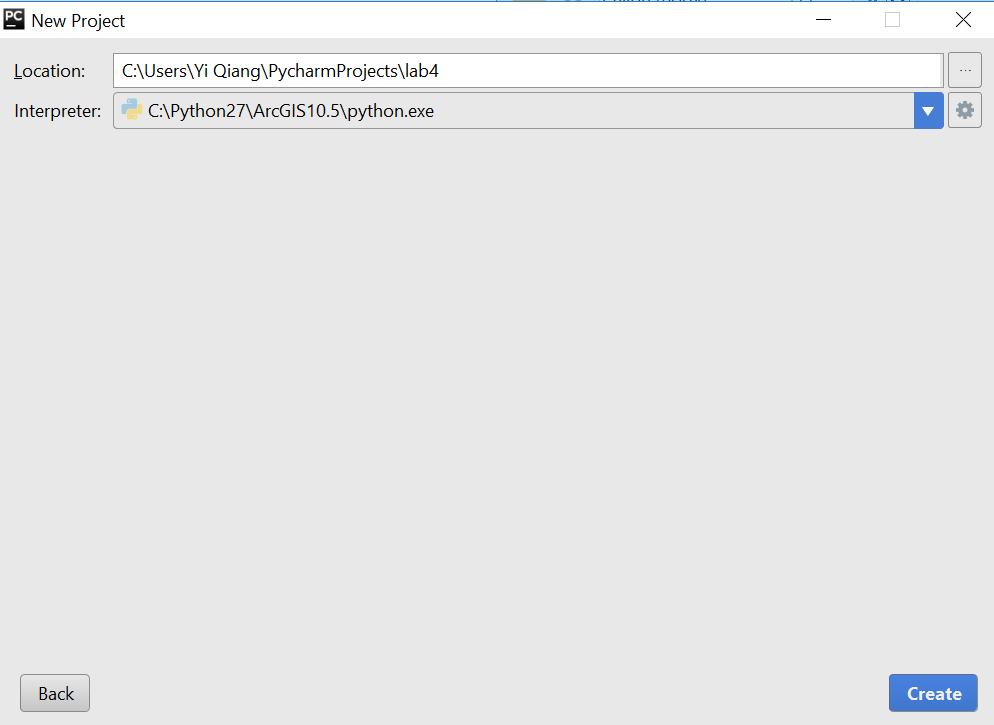
* Click OK for the following dialog.



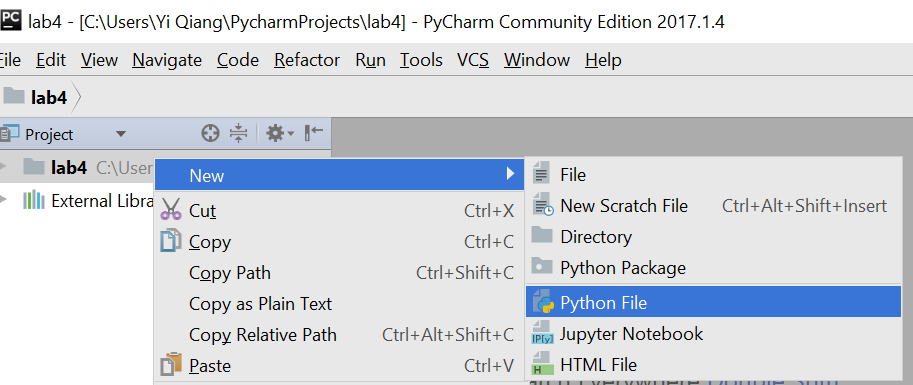
* Create a new project



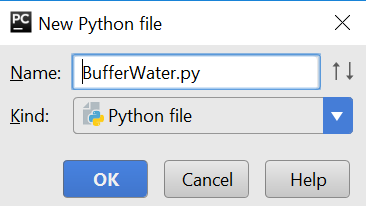
* Save the project in your data folder and name the project **lab4**, click Create.



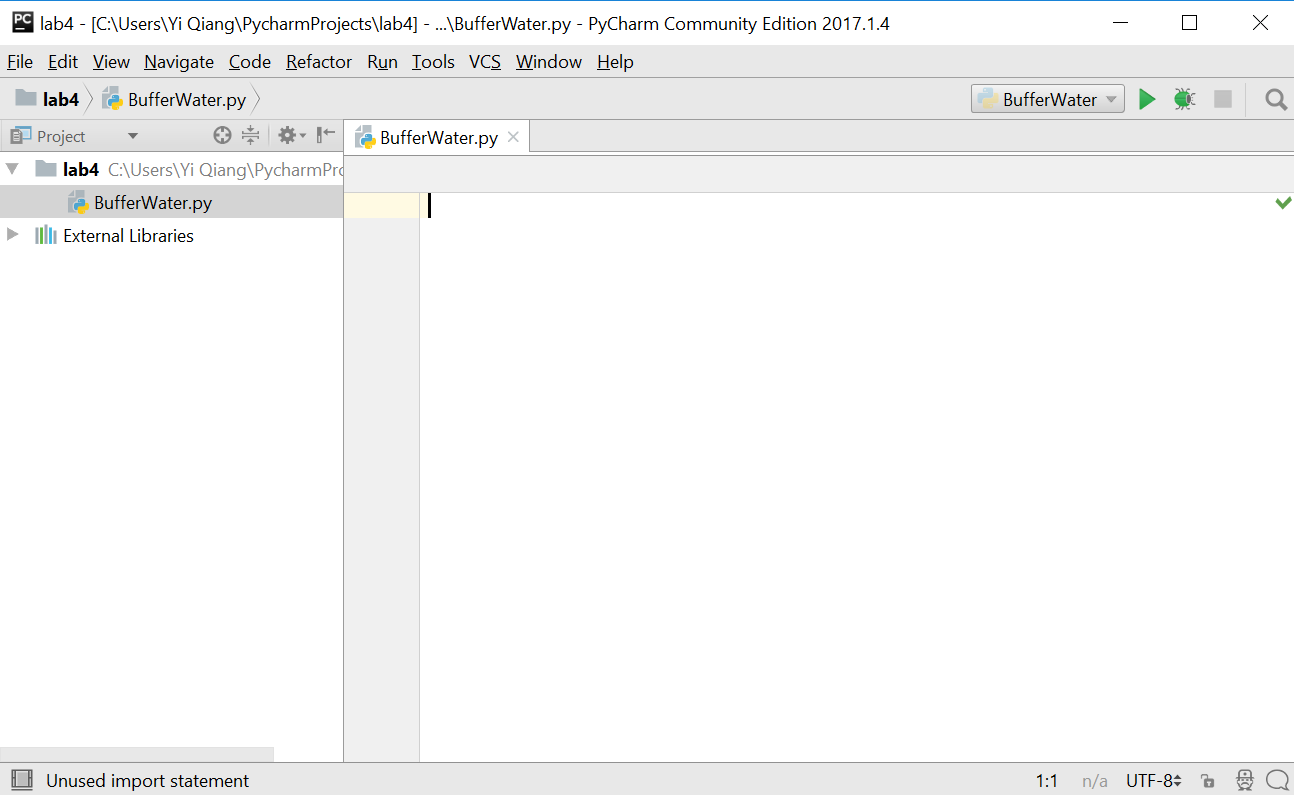
* Close Tips dialog if it appears. Right click on the **lab4** project -> New -> Python File



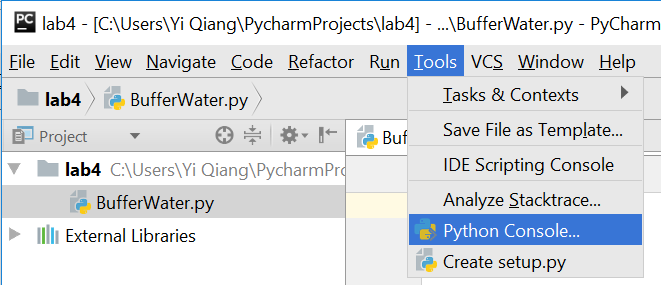
* Name the new file as BufferWater.py



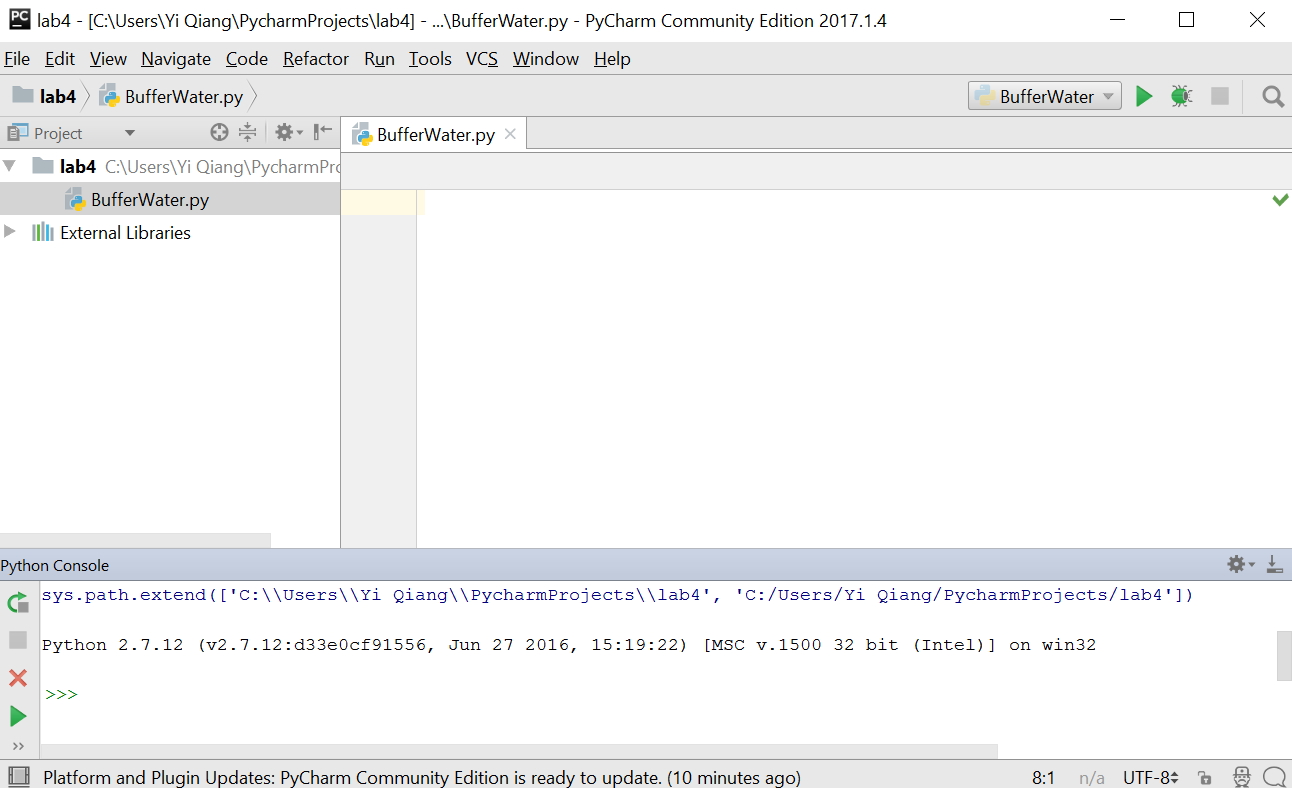
* Now you will see an empty editor, where you can input script



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

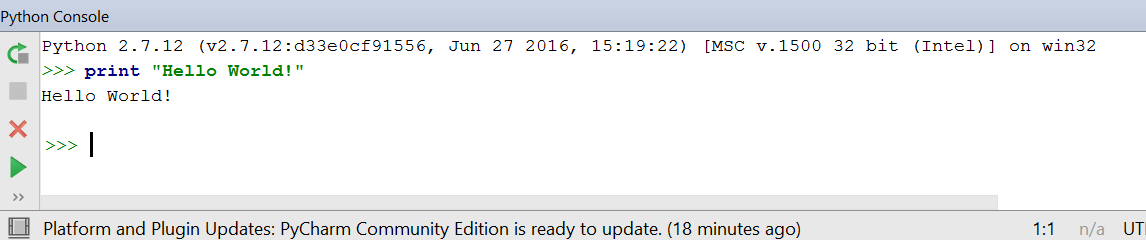


* Now, your PyCharm interface should look like:



* 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.

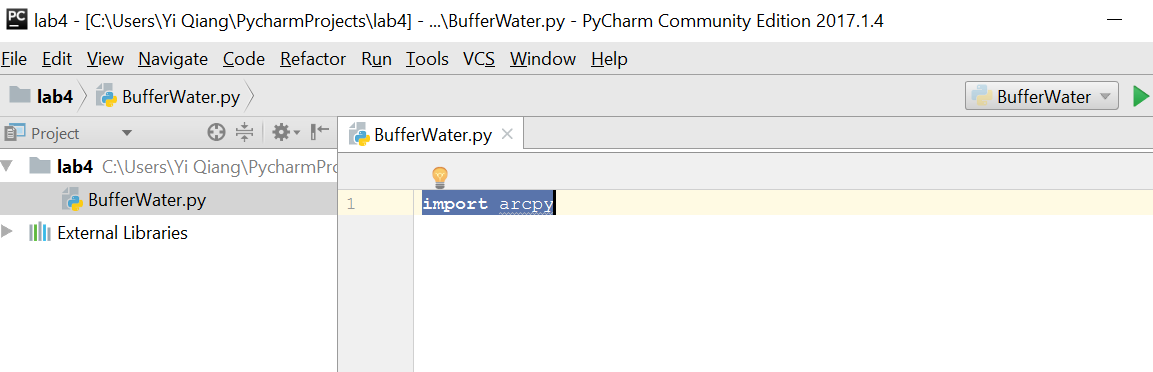


## Step 5: Set geoprocessing environments

In this step, you will set two geoprocessing environments.

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.

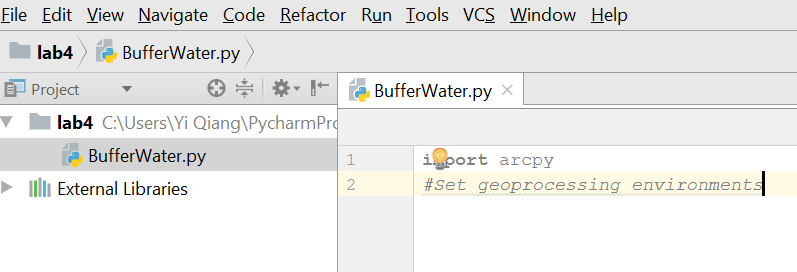


Next, you will enter your geoprocessing workspace.

Documentation (e.g. comments) 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 are just description of the script and will not run.

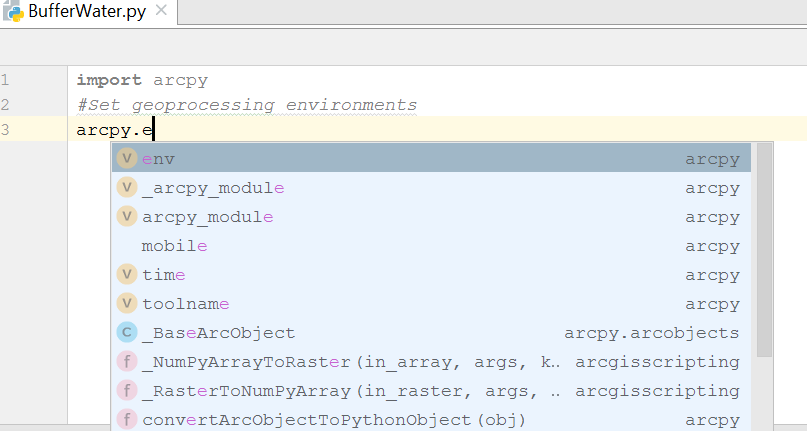
* On line 2, enter the following comment

#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 select the workspace function.
* Set your workspace equal to the geodatabase path, C:/PythonGP10\_0/Data/SanJuan.gdb.

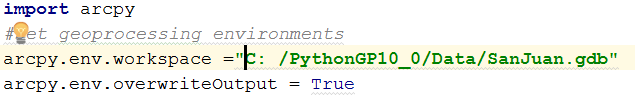
Note: If you saved the course data to a location other than the default, set your workspace to match the location of your data.

* 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.

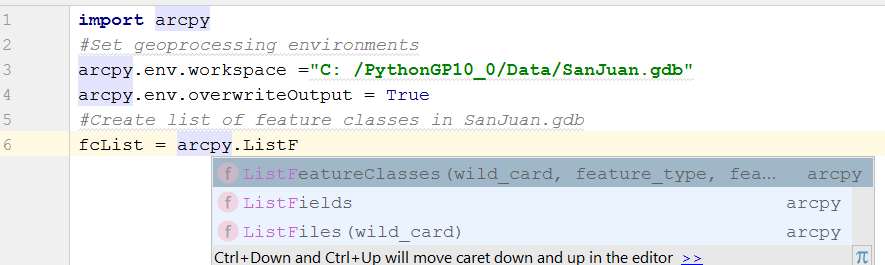
Note: False and True are Boolean data type, not string

* Add the following comment:

#Create list of feature classes in SanJuan.gdb

* In line 6 type:

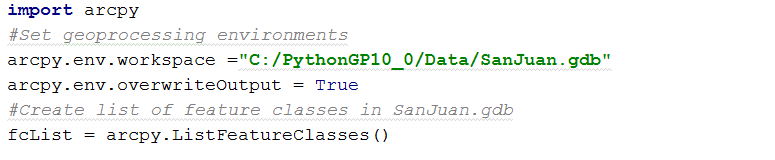
fcList = arcpy.List



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

Using ListFeatureClasses,you will load all of the feature classes in your SanJuan geodatabase into a list.

* 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 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 = []

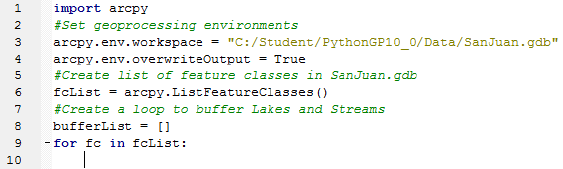
This will create a new, empty Python list.

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.

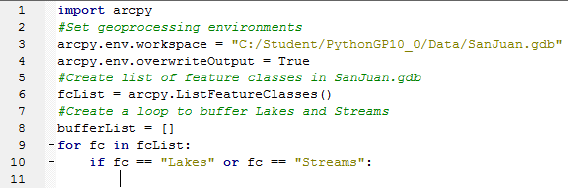
Next, you will write an if statement to determine if the name of your feature class is Lakes or Streams.

* 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

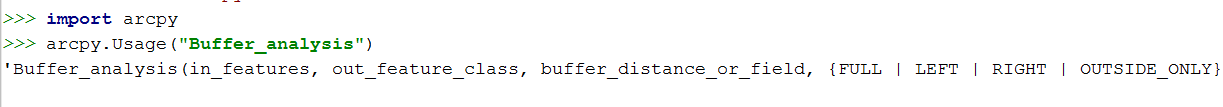
Now you are ready to write the code to create the buffer polygons around the Lakes and Streams.

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. Then, you can see the

Import arcpy

arcpy.Usage("Buffer\_analysis")



Note: Variables without curly parenthesis (e.g. in\_feature, out\_feature\_class, buffer\_distance\_or\_field) are required to run the function. Variables in the curly parenthesis are optional parameters. You can still run the function if you do not specify the optional parameters.

You will buffer your water features a distance of 1000 meters and store the buffer polygons in a new feature class named WaterBuffer.

* Complete the code for your buffer of the Lakes and Streams feature classes:

arcpy.Buffer\_analysis(fc, fc + "Buffer", "1000 meters")

## Answer Question 1

Enter the following code to finish your if statement.

bufferList.append(fc + "Buffer")

## Answer Question 2

## Step 9: Union buffer polygons

Now that you have created buffer polygons for the Lakes and Streams feature classes, you will union these polygons into a new WaterBuffer feature class.

Your for-loop is now complete and you have a list of feature classes stored in bufferList. You will union these feature classes, using the list as input.

* Make sure your cursor is not indented. You want the Union function to run after your loop is complete.
* Using your knowledge of adding geoprocessing functions to your script, add the Union\_analysis function.

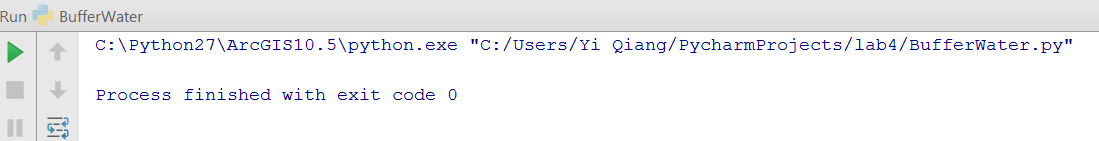
Tip: if you don’t know what should be the input for the Union tool, you can always type arcpy.Usage(“Union\_Analysis”) to learn its syntax (e.g. inputs).

* Now, write a line of script to union the buffers of Streams and Lakes into a new shapefile “WaterBuffer”. **There is no instruction given. You need to write this line by yourself.**
* Press Ctrl + S in your keyboard to save the WaterBuffer.py.
* Click the Run button to run the script.



If the Stop button is red, the program is running. When the program stops, it will be grey.

If your program has finished without error, you will see



Otherwise, the console will return an error message and tell you in which line the error is.

* Now restore ArcMap and add WaterBuffer into it to see if it has created correctly. The water buffer should be the union of stream and lake buffers.

## Answer Question 3

## Answer Question 4

# Part 2: Use Python to create buffers around forest roads

In this exercise, you will create treatment areas for invasive plant species by creating polygons around the forest roads. These polygons will define the non-chemical treatment areas.

## Step 1: Examine BufferDistance table

In this step, you will examine the data that will be used by your script.

* Start ArcMap and open the ..\PythonGP10\_0\SanJuan.mxd, if necessary.
* Open the Catalog window and expand the ..\PythonGP10\_0\Data\SanJuan geodatabase.



* Drag the BufferDistance table into your map.
* In the table of contents, right-click the BufferDistance table and choose Open.



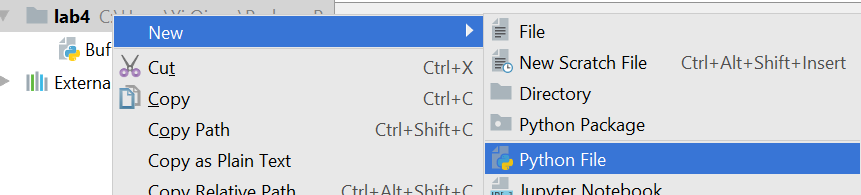
This is the table of buffer distances that will be used to create buffer polygons around your road features.

* Close the BufferDistance table.
* Save your map document and close ArcMap.

# Step 3: Begin your script and set the workspace environment

Now, you will create a new Python file.

* Right click on the **lab4** project -> New -> Python File. Name it **BufferRoads.py**.



* First, please write three lines of scripts to
  + 1) link your program to arcpy;
  + 2) set workspace to SanJuan.gdb;
  + 3) allow your script to overwrite the output if it already exists.

The next section of your script will join your BufferDistance table to the Roads feature class. Prior to executing the join, you will create four variables that will store the parameters used by the arcpy.Join function.

* Add a comment indicating that you are setting the parameters for the Join function.

#Set parameters used to join the BufferDistance table to the Roads feature class

* In the Python console, enter the following to check the syntax of the JoinField\_management tool.

arcpy.Usage("JoinField\_management")

## Answer Question 5

Now, write a line (or multiple lines) script to join the **Roads** layer with the **BufferDistance** Table using the JoinField\_management tool. Please refer to the following table for the input of the tool.

|  |  |
| --- | --- |
| Input variable | Value |
| in\_data | “Roads” |
| in\_field | “ROUTE\_TYPE” |
| join\_table | “BufferDistance” |
| join\_field | “ROUTE\_TYPE |

You need to write Python script to fill the empty part below by yourself.



## Step 5: Buffer the Roads

Now you are ready to buffer the roads using the joined DISTANCE attribute. Before you execute the Buffer function, you will store the parameters as variables.

* Create a comment for this section of your script

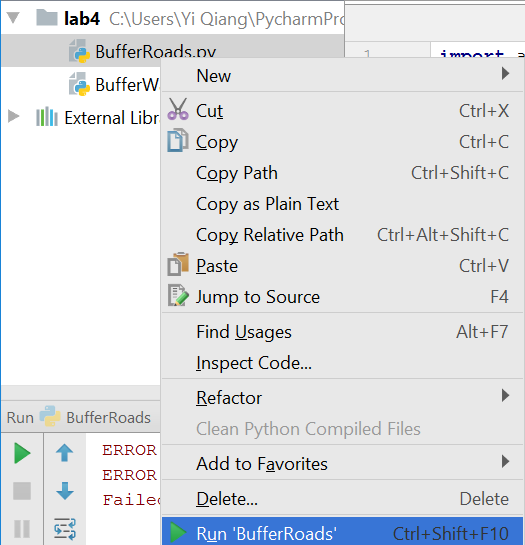
#Set parameters used to buffer Roads feature class

* Add a line (or multiple lines) of script to create buffers around roads and use the Distance field as the buffer distance. Name the output “RoadBuffers”

Finally, your scripts should look like below. The red parts are scripts you write.



* Run BufferRoads.py. You can write click on BufferRoads.py under the lab4 project.



If you program has run without error, add RoadBuffers into ArcMap.

## Answer Question 6

## Answer Question 7