

## 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.

**Estimated completion time: 20 minutes**

To complete exercises, you need the following:

ArcGIS Desktop 10.0 or ArcGIS Desktop 10.1 or ArcGIS Desktop 10.2 or ArcGIS Desktop 10.3 or ArcGIS Desktop 10.4 (Basic, Standard, or Advanced)

PythonWin 2.6 Or PythonWin 2.7

**Note:** PythonWin 2.6 and 2.7 are available as a free download. If you do not have PythonWin installed, the first course exercise provides instructions for downloading the appropriate version.

### - Step 1: Download the data


To complete the exercise, you must download the data. If you have already downloaded and installed the data, continue to the next step.

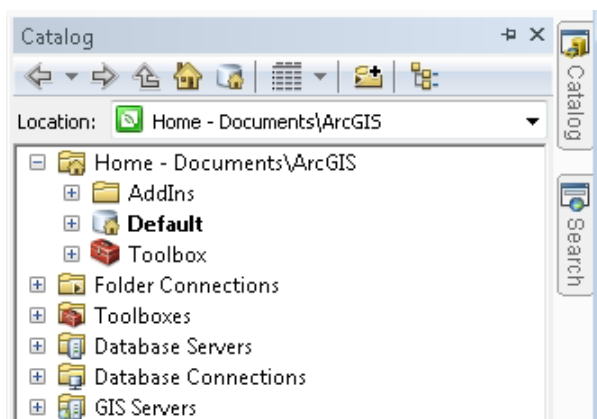
### - Step 2: Connect to your course data

In this step, you will create a connection to your course data.

Open ArcMap with a blank map.

Click the Catalog tab on the right side of the ArcMap window to open the Catalog window.


**Tip:** If you do not see the Catalog tab, click the Catalog window button  on the Standard toolbar.



Step 2a: Connect to your course data.

Make a connection to your **C:\Student\PythonGP10\_0** folder.

### Remind me how

1. In the Catalog window, click the Connect to Folder button .
2. In the Connect to folder dialog box, browse to your **C:\Student\PythonGP10\_0** folder.

**Note:** If you installed your course data in another location, browse to your course data folder.

Your course data folder is now listed as a folder connection in the Catalog window.

## - Step 3: Explore San Juan data

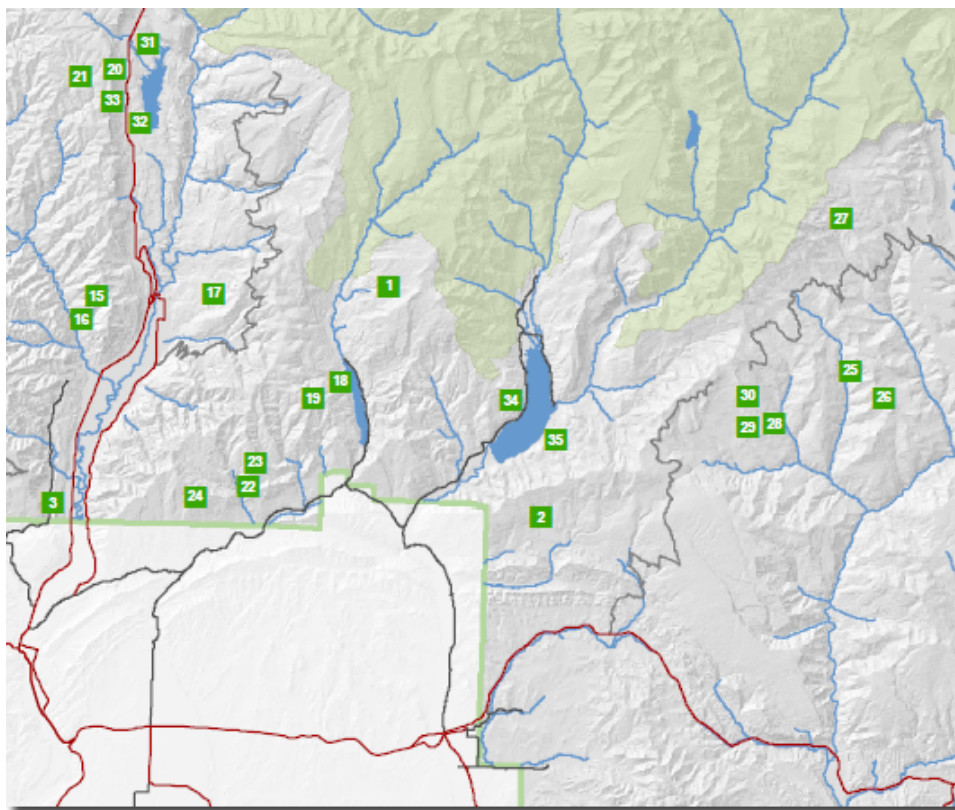
In this step, you will become familiar with the San Juan National Forest data that you will work with in this course.

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

Zoom to the Invasive Plants bookmark.

### Remind me how

From the Bookmarks menu, choose Invasive Plants.



Step 3a: Explore San Juan data.

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 PythonWin

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

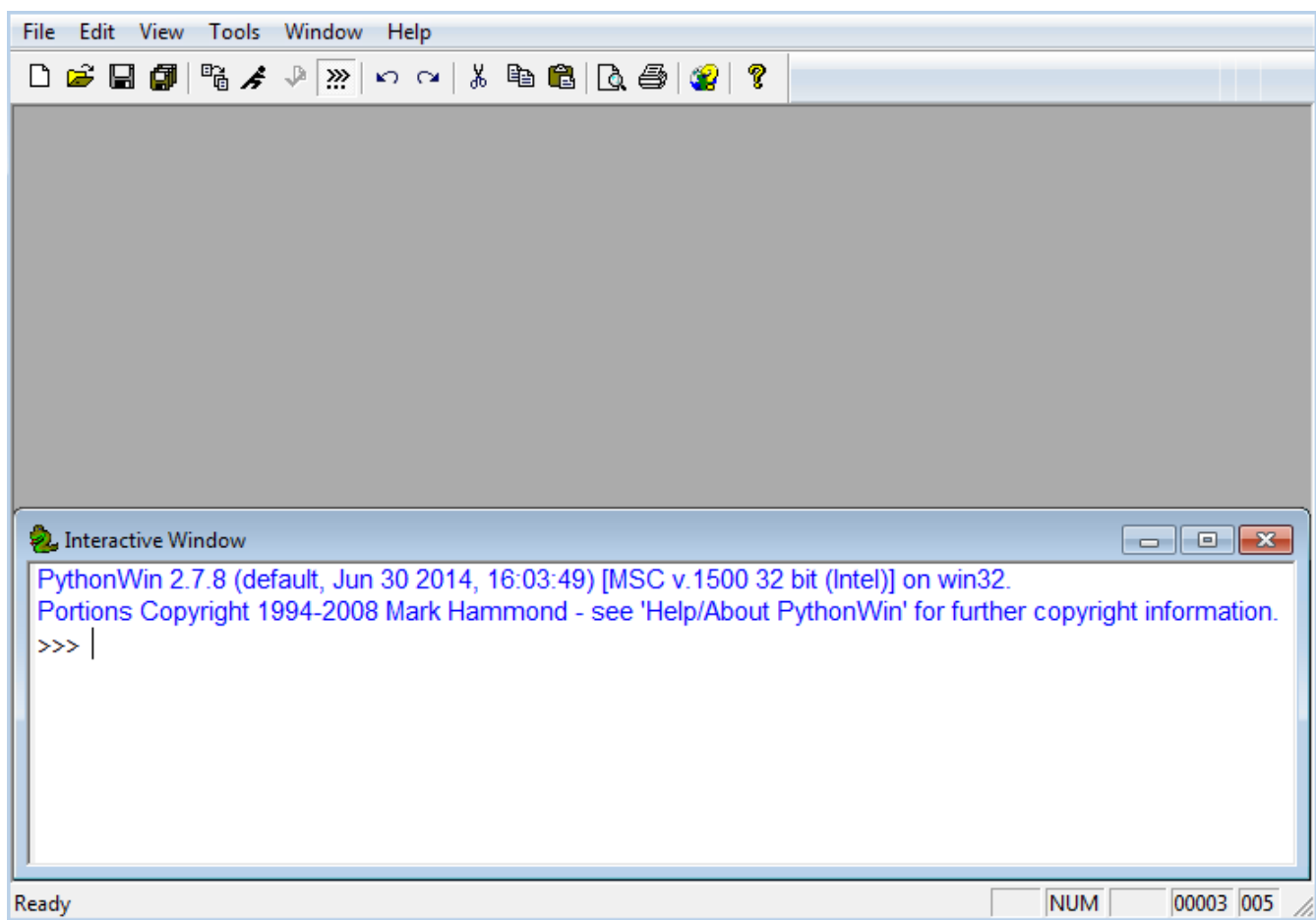
Open Windows Explorer.

Navigate to **C:\Python27\ArcGIS10.x\Lib\site-packages\pythonwin**.

(**ArcGIS 10.0 users:** Navigate to C:\Python26\ArcGIS10.0\Lib\site-packages\pythonwin.)

**i** Can't find PythonWin?

Double-click **Pythonwin.exe**.

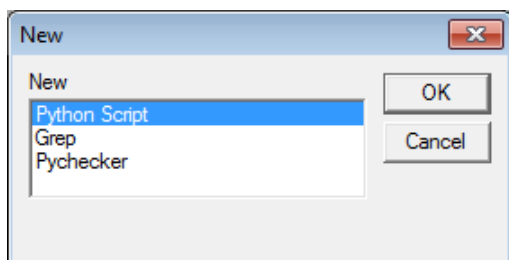


Step 4a: Set up PythonWin.

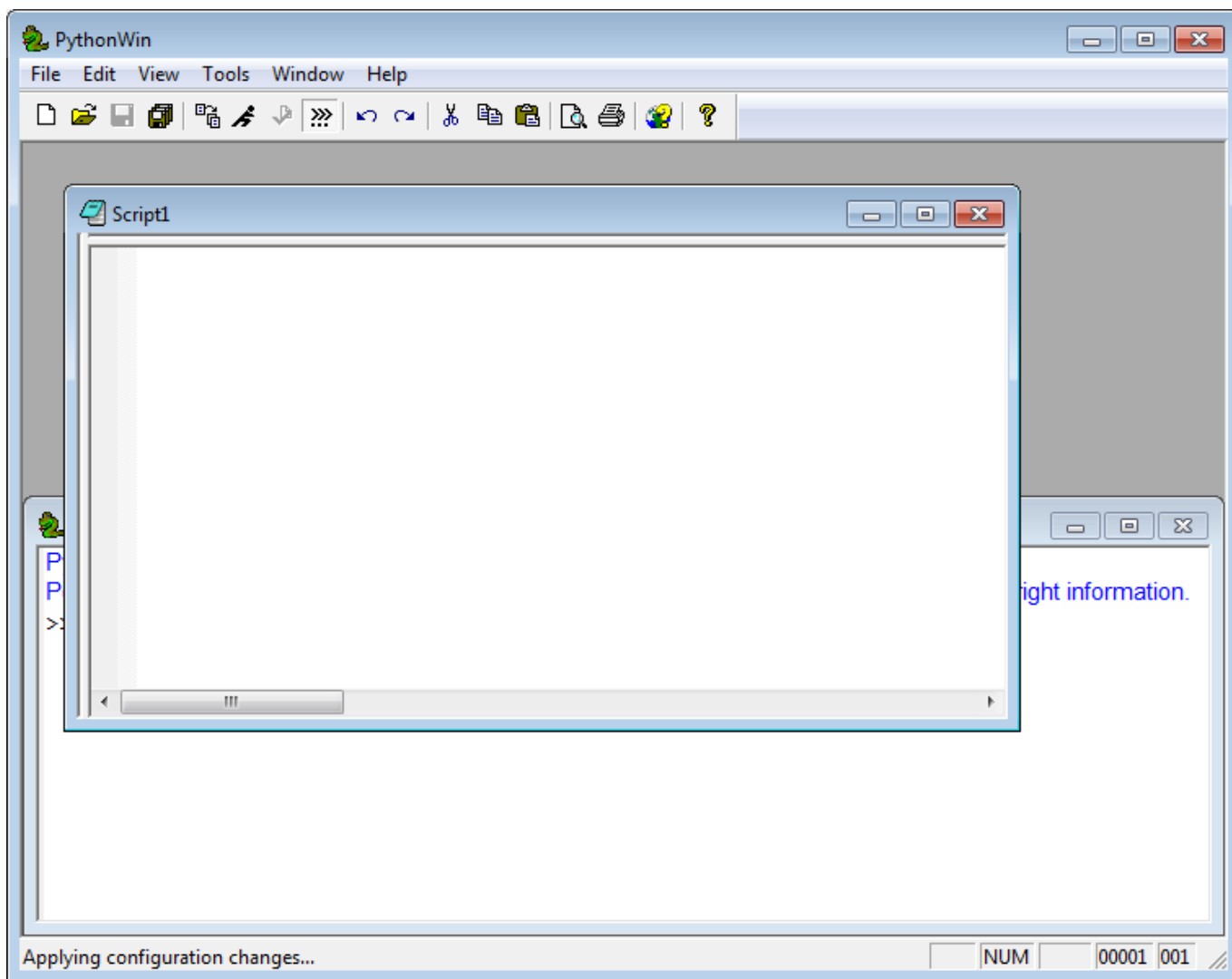
Your PythonWin version may differ.

Click the New button .

On the New dialog box, confirm that Python Script is highlighted.



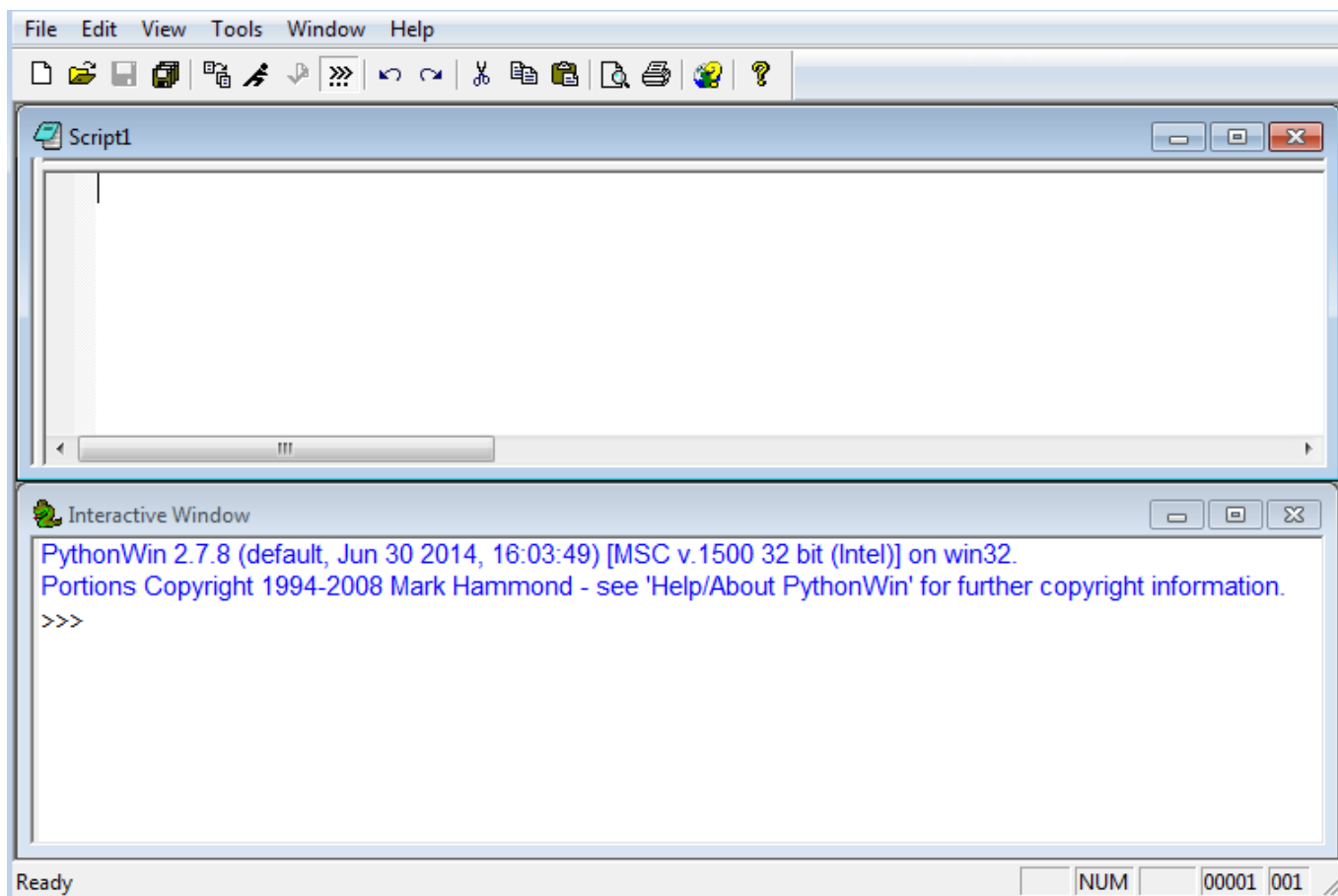
Click OK.



Step 4b: Set up PythonWin.

Your new script is called Script1 and appears in a new window.

From the Window menu, choose Tile.



Step 4c: Set up PythonWin.

Your PythonWin version may differ.

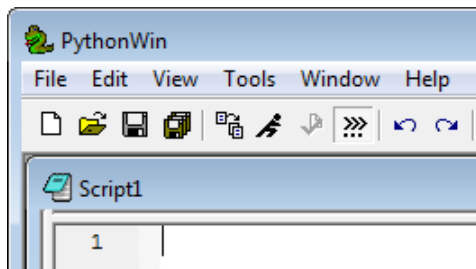
Next, you will display line numbers to more easily edit and debug your scripts.

From the View menu, choose Options.

Click the Editor tab.

Under Margins Widths, for Line Numbers, enter **30** and click OK.

Each line in your script will now be numbered. Your cursor is set to begin your script on the first line.



Save your script in your `..\PythonGP10_0\Scripts` folder as **BufferWater.py**.

#### - Step 5: Set geoprocessing environments

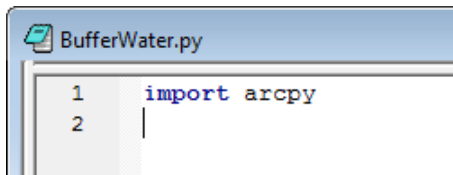
In this step, you will set two geoprocessing environments.

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 Interactive Window, type **import arcpy** and press Enter.

PythonWin will take a few moments to import the ArcPy libraries.

In your BufferWater.py script window, on line 1, type **import arcpy** and press Enter.



Step 5a: Set geoprocessing environments.

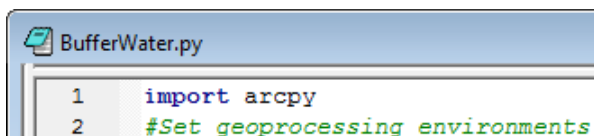
Because you are writing and running your script outside of ArcGIS, you need to include this statement to access ArcGIS Python functionality.

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. You will add comments to begin each major part of your script.

On line 2, enter the following comment:

**#Set geoprocessing environments**

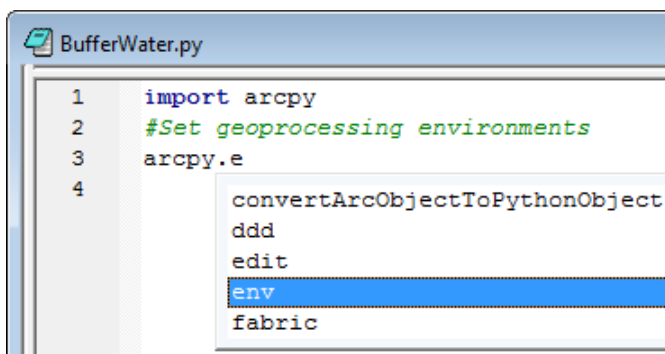


Step 5b: Set geoprocessing environments.

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.



Step 5c: Set geoprocessing environments.

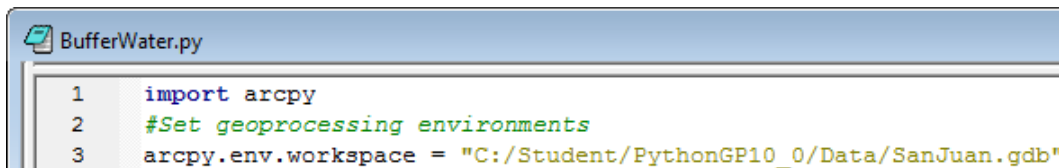
Press the Tab key to add the **env** class to your statement.

**Tip:** You can also double-click an item in the list to add it to your script.

Continue your statement by typing a dot (.) then the **workspace** function.

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

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



```
1 import arcpy
2 #Set geoprocessing environments
3 arcpy.env.workspace = "C:/Student/PythonGP10_0/Data/SanJuan.gdb"
```

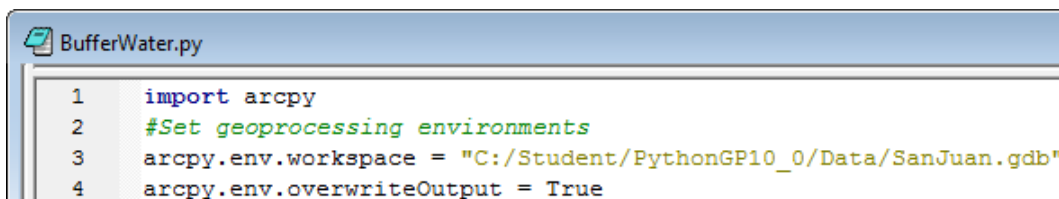
Step 5d: Set geoprocessing environments.

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

Next, you will enter one more environment setting.

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

**arcpy.env.overwriteOutput = True**



```
1 import arcpy
2 #Set geoprocessing environments
3 arcpy.env.workspace = "C:/Student/PythonGP10_0/Data/SanJuan.gdb"
4 arcpy.env.overwriteOutput = True
```

Step 5e: Set geoprocessing environments.

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.

**Tip:** Because `overwriteOutput` is a boolean data type, you could also enter 1 for `True` or 0 for `False`.

#### - Step 6: Create a list of feature classes

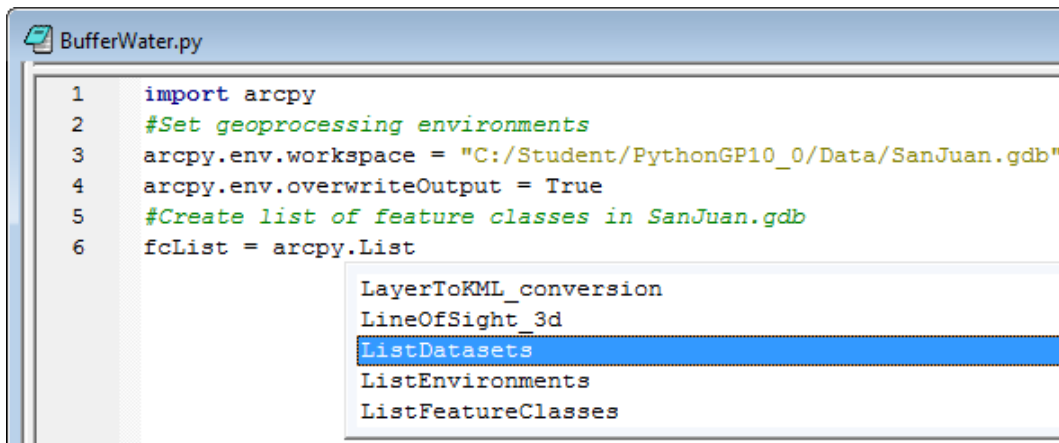
In this step, you will create a list of feature classes in the SanJuan geodatabase. You will create buffer polygons for the streams and lakes feature classes.

First, you will create a comment to document the next section of your script.

Add the following comment:

**#Create list of feature classes in SanJuan.gdb**

On line 6, type **fcList = arcpy.List**



Step 6a: Create a list of feature classes.

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

To begin entering the parameters for ListFeatureClasses, type a left-parenthesis.

```
(wild_card=None, feature_type=None, feature_dataset=None)
ListFeatureClasses({wild card}, {feature type}, {feature dataset})
```

Notice that the usage for the ListFeatureClasses function appears. (**ArcGIS 10.0 users:** Only the parameters for the ListFeatureClasses function appear.)

All the parameters are listed in braces, { }, which indicate that these are optional.

Refer to the ArcGIS Help for ListFeatureClasses (ArcGIS 10.2 users | ArcGIS 10.1 users | ArcGIS 10.0 users).

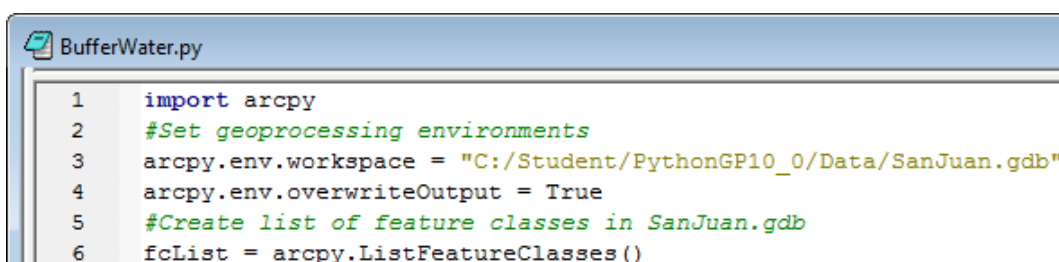


If you do not provide any parameters, what will be returned by the ListFeatureClasses function?

The wildcard 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 6b: Create a list of feature classes.

### - 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.

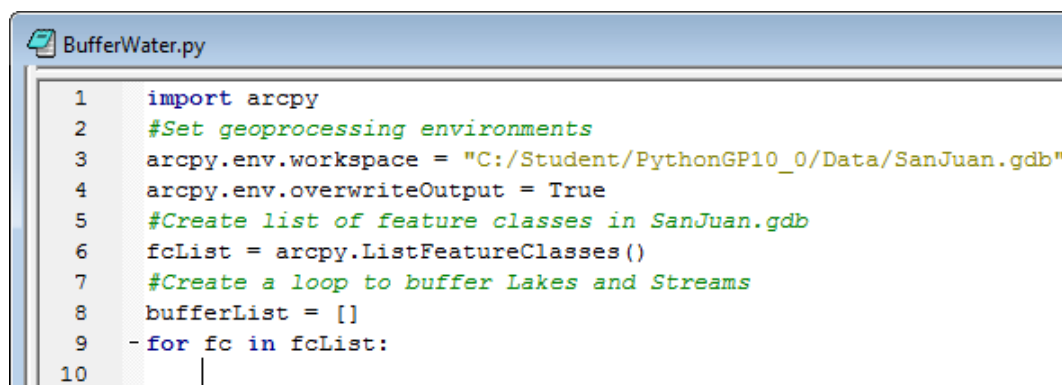
Now you will create a `for`-loop to iterate through each feature class stored in `fcList`

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 PythonWin.



```
BufferWater.py
1  import arcpy
2  #Set geoprocessing environments
3  arcpy.env.workspace = "C:/Student/PythonGP10_0/Data/SanJuan.gdb"
4  arcpy.env.overwriteOutput = True
5  #Create list of feature classes in SanJuan.gdb
6  fcList = arcpy.ListFeatureClasses()
7  #Create a loop to buffer Lakes and Streams
8  bufferList = []
9  - for fc in fcList:
10     |
```

Step 7a: Loop through your list of feature classes.

**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.

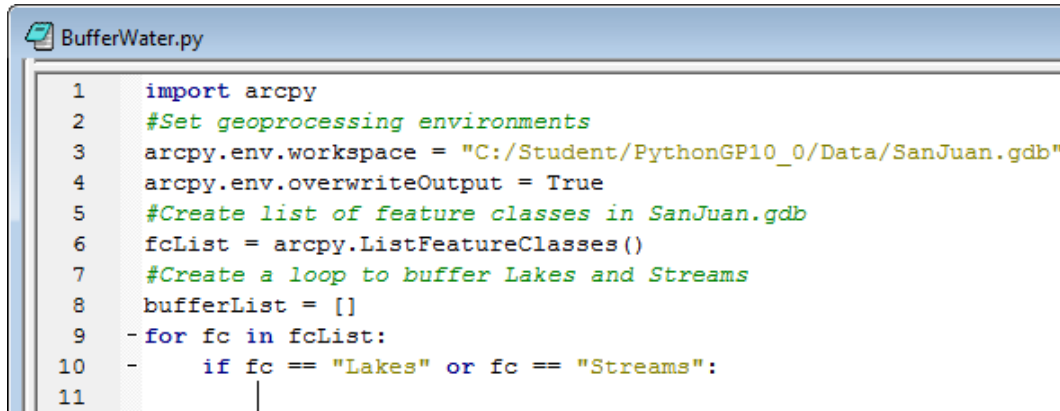
Next, you will write an `if` statement to determine if the name of your feature class is Lakes or Streams.

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.

Press Enter to see your indented cursor location.



```
1 import arcpy
2 #Set geoprocessing environments
3 arcpy.env.workspace = "C:/Student/PythonGP10_0/Data/SanJuan.gdb"
4 arcpy.env.overwriteOutput = True
5 #Create list of feature classes in SanJuan.gdb
6 fcList = arcpy.ListFeatureClasses()
7 #Create a loop to buffer Lakes and Streams
8 bufferList = []
9 - for fc in fcList:
10 -     if fc == "Lakes" or fc == "Streams":
11         |
```

Step 7b: Loop through your list of feature classes.

#### - Step 8: Buffer lakes and streams

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

Enter the following code:

```
arcpy.Buffer
```

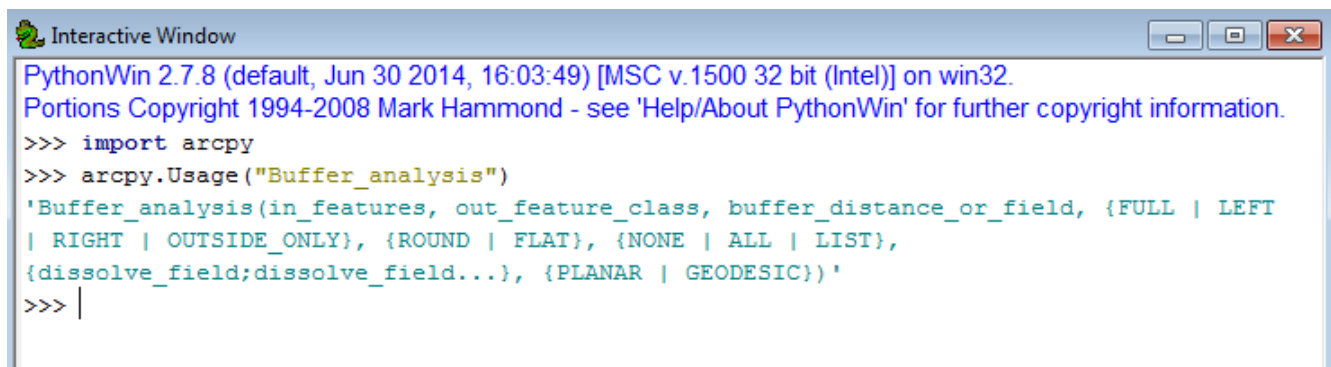
Choose the Buffer\_analysis tool from the list.

**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")
```



```
PythonWin 2.7.8 (default, Jun 30 2014, 16:03:49) [MSC v.1500 32 bit (Intel)] on win32.
Portions Copyright 1994-2008 Mark Hammond - see 'Help/About PythonWin' for further copyright information.
>>> import arcpy
>>> arcpy.Usage("Buffer_analysis")
'Buffer_analysis(in_features, out_feature_class, buffer_distance_or_field, {FULL | LEFT
| RIGHT | OUTSIDE_ONLY}, {ROUND | FLAT}, {NONE | ALL | LIST},
{dissolve_field;dissolve_field...}, {PLANAR | GEODESIC})'
>>> |
```

Step 8a: Buffer lakes and streams.

ArcGIS 10.0, 10.1, or 10.2 users: You do not have the optional method parameter {PLANAR | GEODESIC}.



What are the three required parameters for the Buffer function?

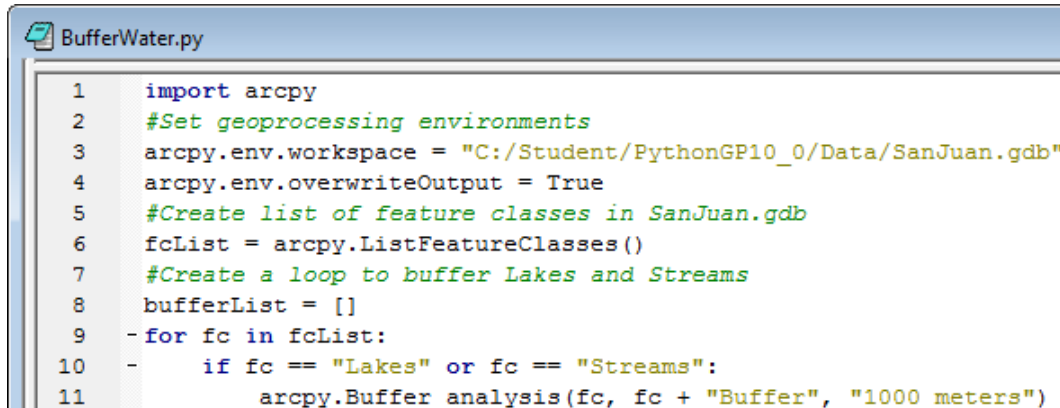


What will be your in\_features parameter?

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")
```



```
1 import arcpy
2 #Set geoprocessing environments
3 arcpy.env.workspace = "C:/Student/PythonGP10_0/Data/SanJuan.gdb"
4 arcpy.env.overwriteOutput = True
5 #Create list of feature classes in SanJuan.gdb
6 fcList = arcpy.ListFeatureClasses()
7 #Create a loop to buffer Lakes and Streams
8 bufferList = []
9 for fc in fcList:
10     if fc == "Lakes" or fc == "Streams":
11         arcpy.Buffer_analysis(fc, fc + "Buffer", "1000 meters")
```

Step 8b: Buffer lakes and streams.



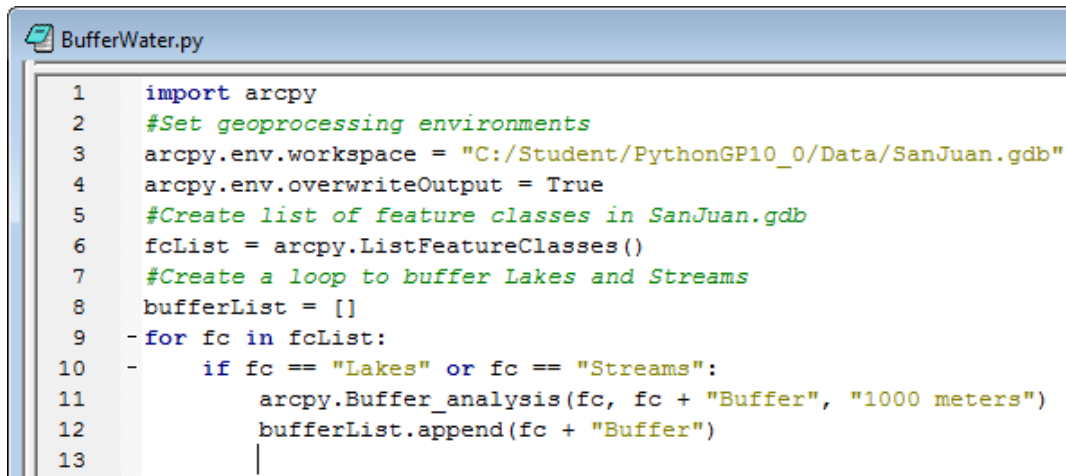
From the above code, what will be the output name of each feature class after it is buffered?

**Tip:** The buffer\_distance parameter could also have been given as "1000" without the units. However, specifying the units allows your code to be more readable. In addition, you do not have to know the units of the spatial reference. If you only specify the distance, the units of the spatial reference will be used. In this case, your feature classes are already stored in meters, so you could have provided the distance without the units.

Now you will add the name of the new buffer feature class to the bufferList variable. Earlier you created an empty Python list. Now you will use the append method to add each buffered feature class to your list.

Enter the following code to finish your if statement.

```
bufferList.append(fc + "Buffer")
```



```

1  import arcpy
2  #Set geoprocessing environments
3  arcpy.env.workspace = "C:/Student/PythonGP10_0/Data/SanJuan.gdb"
4  arcpy.env.overwriteOutput = True
5  #Create list of feature classes in SanJuan.gdb
6  fcList = arcpy.ListFeatureClasses()
7  #Create a loop to buffer Lakes and Streams
8  bufferList = []
9  -for fc in fcList:
10 -    if fc == "Lakes" or fc == "Streams":
11         arcpy.Buffer_analysis(fc, fc + "Buffer", "1000 meters")
12         bufferList.append(fc + "Buffer")
13

```

Step 8c: Buffer lakes and streams.

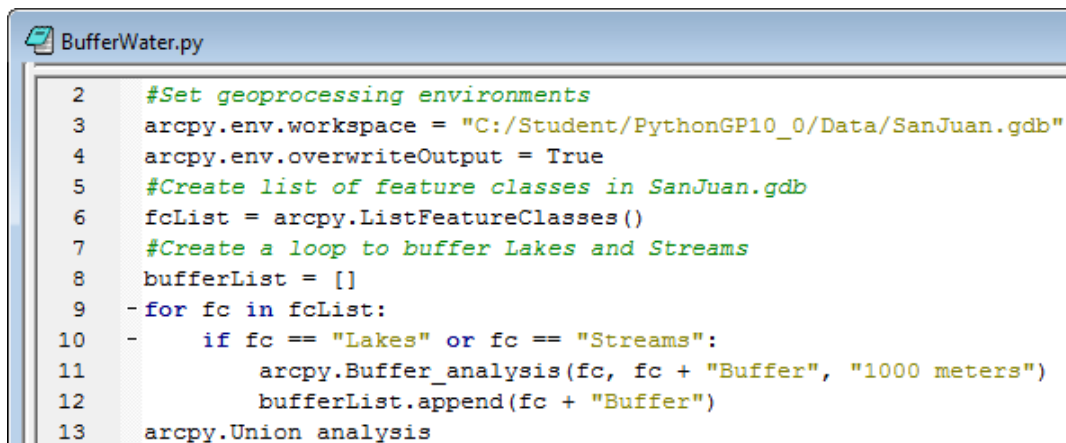
#### - Step 9: Union buffer polygons

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 your 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.



```

2  #Set geoprocessing environments
3  arcpy.env.workspace = "C:/Student/PythonGP10_0/Data/SanJuan.gdb"
4  arcpy.env.overwriteOutput = True
5  #Create list of feature classes in SanJuan.gdb
6  fcList = arcpy.ListFeatureClasses()
7  #Create a loop to buffer Lakes and Streams
8  bufferList = []
9  -for fc in fcList:
10 -    if fc == "Lakes" or fc == "Streams":
11         arcpy.Buffer_analysis(fc, fc + "Buffer", "1000 meters")
12         bufferList.append(fc + "Buffer")
13  arcpy.Union_analysis

```

Step 9a: Union buffer polygons.

Type a left parenthesis and view the usage for the function.

**Tip:** Remember that you can also use the `arcpy.Usage` function in the Interactive window.



What are the required parameters?

Complete your code as follows:

```
arcpy.Union_analysis(bufferList, "WaterBuffers")
```

#### - Step 10: Run your script

You are now ready to run your script.

Click the Save button  to save your script.

Click the Run button .

Click OK in the Run Script dialog box.

Your script will take a few moments to run. When PythonWin is finished, you should see the following message at the bottom of the PythonWin window:

```
Script 'C:\Student\PythonGP10_0\Scripts\BufferWater.py' returned exit code 0
```

 What if I receive an error?

Close PythonWin.

#### - Step 11: View WaterBuffer in ArcMap

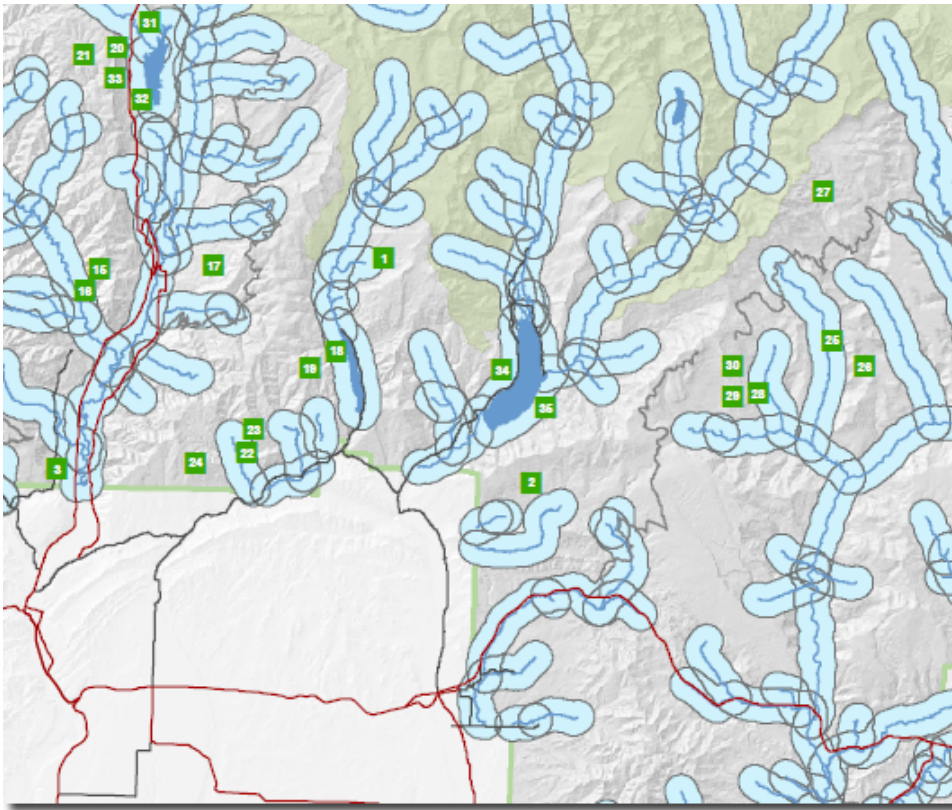
Now that your water buffers are complete, you will look at the result of your Python script.

Restore your ArcMap session.

In the Catalog window, browse to your **..\PythonGP10\_0\Data\SanJuan** geodatabase.

Refresh your database to show the new buffer feature classes.

Drag the WaterBuffers feature class into your table of contents below the Lakes layer.



Step 11a: View WaterBuffer in ArcMap.

You now have the non-chemical treatment areas defined around water features. In the next exercise, you will create buffer polygons around the forest roads using a different technique.

Save your map document.

If you are going on to the Challenge step or the next exercise, keep ArcMap open; otherwise, close ArcMap.

#### + Challenge: Dissolve polygons

Notice that each stream segment in your map has its own buffer polygon. When buffering the polygons, you can choose to reduce the number of overlapping features produced by the tool.

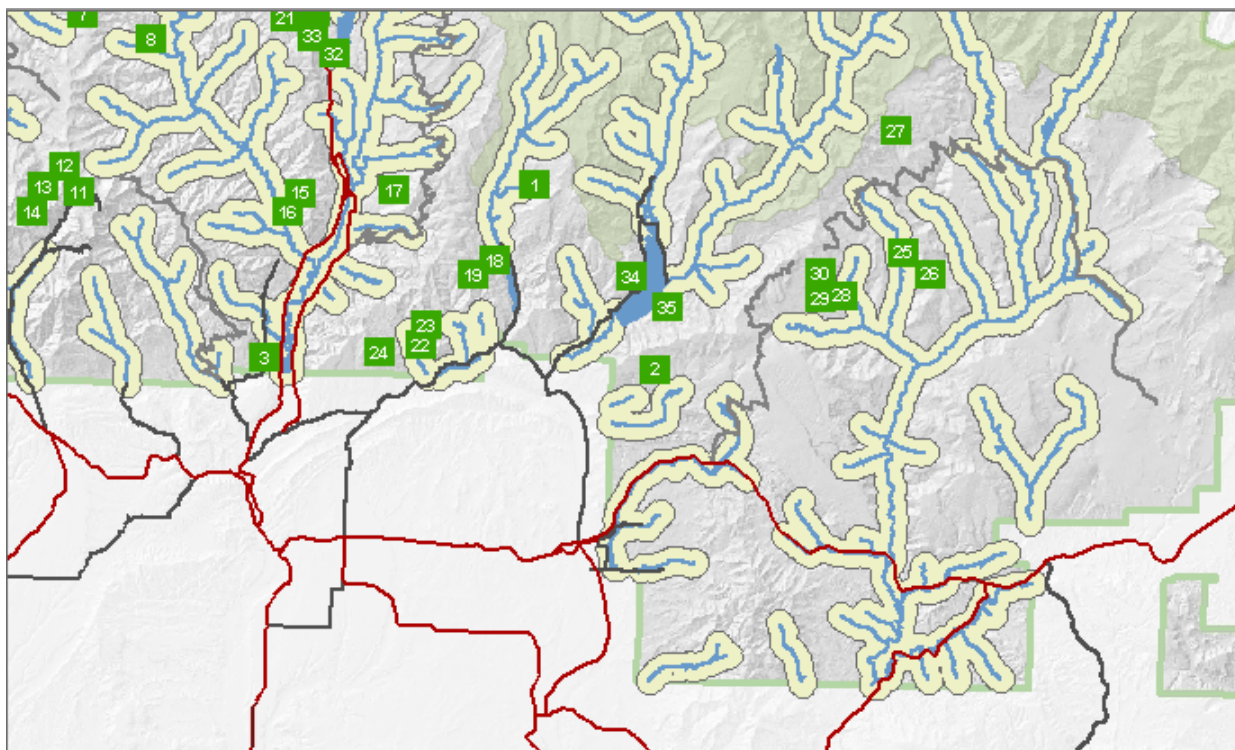
Modify the Buffer function in your script to set the `dissolve_option` parameter to dissolve all the polygon boundaries where they overlap.

**Note:** Alternatively, you could add the `arcpy.Dissolve_management` function to your script.

If you want to test your script:

1. Remove the WaterBuffers layer from ArcMap. (Having the layer active in ArcMap will create a lock on the WaterBuffers feature class, preventing your script from running.)
2. Save and run your script—it may take a few minutes to run.
3. In ArcMap, add the StreamsBuffer feature class to your SanJuan.mxd.





Challenge Result.

If you are going on to the next exercise, keep ArcMap open. Otherwise, close ArcMap without saving changes and close PythonWin.