Practice 10: Query

Objectives:

- Utilize the escape character ("\\"), wildcard character, and common operators in the query expression
- Use the data manage tool to avoid the data locks
- Use the Get Count tool to track and verify the selection set
- Utilize escape character in Python script

Datasets:

volcano	point shapefile of volcano
quakehis	point shapefile of historic earth quake hits
rivers	line shapefile of major rivers
parks	polygon shapefile of state/national parks
States	polygon shapefile of state boundary

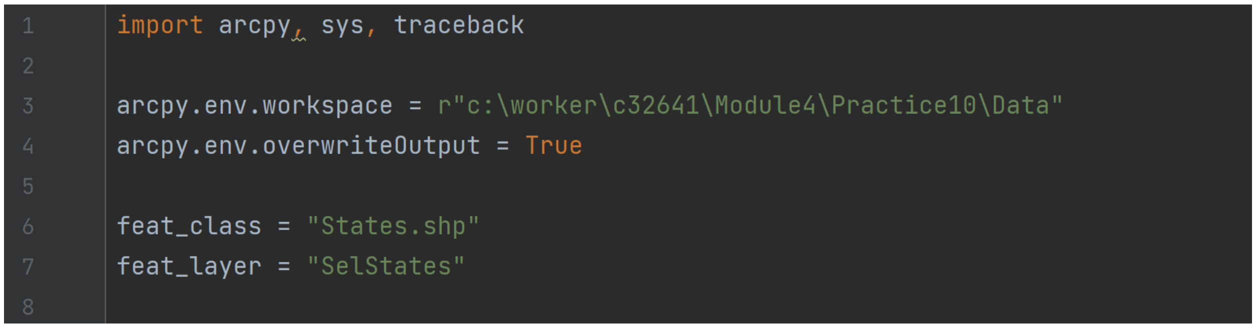
Data Source:
Esri

The codes in this practice are long; make sure that you understand each block of the code. Type one block into the script editor; make sure it run before you start on the next block. Debugging short script is a lot easier than to find the error or typo in the long script.

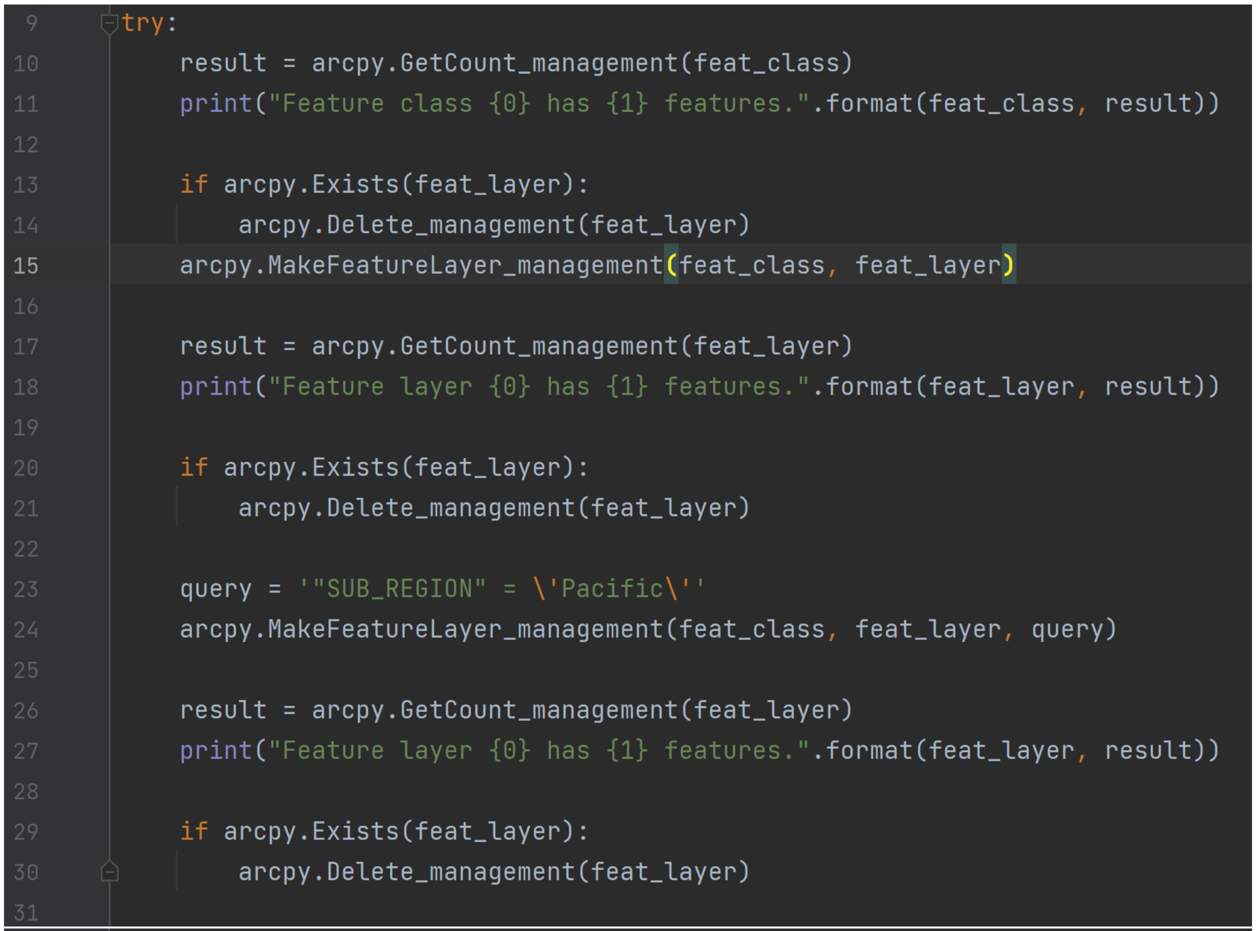
Part I: Use the Make Feature Layer tool to create a feature layer

Create a simple script and run it directly through the Python editor. The script shows how the *Make Feature Layer* tool is used to create a feature layer with and without a query.

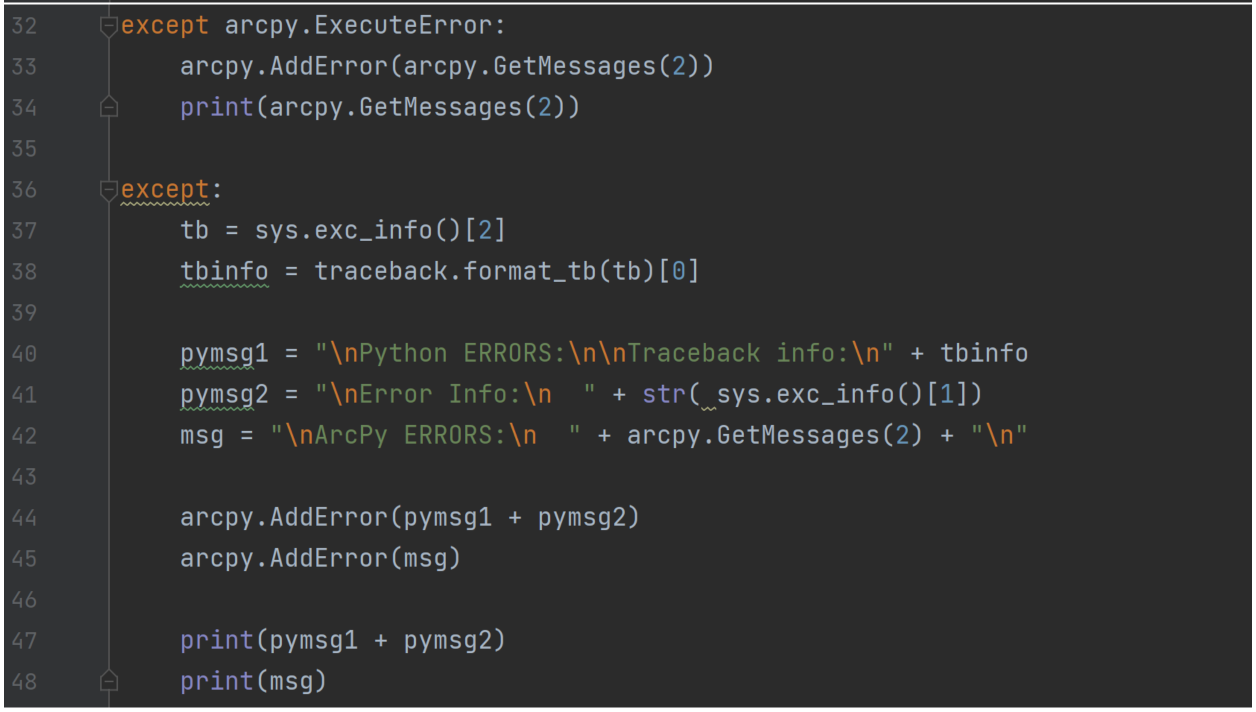
- Practice the `MakeFeatureLayer` tool to create a feature layer
- Use escape character for a single quote (`\'`)
- Utilize the `Exists` and `Delete` method to clean up the in-memory feature layer and avoid the data locks.
- Apply the `GetCount` tool and print statement to keep track and verify the selection set.



```
import arcpy, sys, traceback

arcpy.env.workspace = r"c:\worker\c32641\Module4\Practice10\Data"
arcpy.env.overwriteOutput = True

feat_class = "States.shp"
feat_layer = "SelStates"

```



```python
try:
    result = arcpy.GetCount_management(feat_class)
    print("Feature class {0} has {1} features.".format(feat_class, result))

    if arcpy.Exists(feat_layer):
        arcpy.Delete_management(feat_layer)
    arcpy.MakeFeatureLayer_management(feat_class, feat_layer)

    result = arcpy.GetCount_management(feat_layer)
    print("Feature layer {0} has {1} features.".format(feat_layer, result))

    if arcpy.Exists(feat_layer):
        arcpy.Delete_management(feat_layer)

    query = '"SUB_REGION" = \'Pacific\''
    arcpy.MakeFeatureLayer_management(feat_class, feat_layer, query)

    result = arcpy.GetCount_management(feat_layer)
    print("Feature layer {0} has {1} features.".format(feat_layer, result))

    if arcpy.Exists(feat_layer):
        arcpy.Delete_management(feat_layer)

```



```python
except arcpy.ExecuteError:
    arcpy.AddError(arcpy.GetMessages(2))
    print(arcpy.GetMessages(2))

except:
    tb = sys.exc_info()[2]
    tbinfo = traceback.format_tb(tb)[0]

    pymsg1 = "\nPython ERRORS:\n\nTraceback info:\n" + tbinfo
    pymsg2 = "\nError Info:\n  " + str( sys.exc_info()[1])
    msg = "\nArcPy ERRORS:\n  " + arcpy.GetMessages(2) + "\n"

    arcpy.AddError(pymsg1 + pymsg2)
    arcpy.AddError(msg)

    print(pymsg1 + pymsg2)
    print(msg)
```

Code Explanations:

1. Line 1:
   Import module.

   - In addition to the common module, the `traceback` module is used for the error message.

2. Lines 3 – 4
   Set the path name for the current workspace and allow the output results to be overwritten.

3. Lines 6 – 7
   Assign the feature class into variable.

   - The `feat_class` variable is the shapefile in the **Data** folder, so it needs to be exactly the same as existed data.
   - The `feat_layer` variable is just a string of characters and can be represent any name you choose.

4. Line 9
   Use the `Try/Except` blocks for error handling.

   - See Module 3 lecture note.
   - The end `Except` block is at Lines 32 – 48.

IMPORTANT: The following Lines 10 – 30 are inside the `try` statement, remember to ident the code, so they will be considered as part of `try` statement block.

5. Lines 10 – 11
   Use the `Get Count` tool to retrieve the number of feature (records) in the **States.shp** feature class. Report the message with the `print` function.

   ```
   Python Syntax

   arcpy.management.GetCount(in_rows)
   ```

   ArcGIS Pro Help: https://pro.arcgis.com/en/pro-app/latest/tool-reference/data-management/get-count.htm

6. Lines 13 – 14
   Use the `Exists` function to check whether there is an existing feature layer with the same name. If yes, then utilize the `Delete` function to remove it from the computer memory. If not, then just skip to the next line.

```
Python Syntax

Exists(dataset)
```

ArcGIS Pro Help: https://pro.arcgis.com/en/pro-app/latest/arcpy/functions/exists.htm

```
Python Syntax

arcpy.management.Delete(in_data, {data_type})
```

ArcGIS Pro Help: https://pro.arcgis.com/en/pro-app/latest/tool-reference/data-management/delete.htm

7. Line 15
   Use the **`Make Feature Layer`** function to create an in-memory feature layer **SelStates** of the feature class **States.shp**.

```
Python Syntax

arcpy.management.MakeFeatureLayer(in_features, out_layer,
{where_clause}, {workspace}, {field_info})
```

ArcGIS Pro Help: https://pro.arcgis.com/en/pro-app/latest/tool-reference/data-management/make-feature-layer.htm

8. Lines 17 – 18
   Report the number of features (records) in the feature layer **SelStates**. It should be the same number as the features in the feature class **States.shp**, since the previous line does not contain any query restriction.

9. Lines 20 – 21
   Use the **`Exists`** and **`Delete`** functions to check the existence of the feature layer and remove it if necessary.

10. Lines 23 – 24
    Construct the query expression, then use it to make the feature layer based on the input feature class.

    - Build the query expression by placing the escape character (\) in front of the single quote(**'**). The Python will then recognize the single quote as part of the string, not the bounding character of the string.
    - Utilize the **`Make Feature Layer`** tool to create an in-memory feature layer based on the query restriction by adding the query variable as the third parameter.

The SQL expression will be read like the following in ArcGIS.

```
"SUB_REGION" = 'Pacific'
```

11. Lines 26 – 27
    Report the number of features (records) in the feature layer **SelStates** with the query restrictions.

12. Lines 29 – 30
    It is good practice to free up the computer memory and avoid the data locks by deleting the in-memory feature layer.

13. Lines 32 - 34
    The exception block to handle ArcGIS geoprocessing tool error.

    - The **`ExecuteError`** class is used to specifically handle any geoprocessing tool errors.
      `ExecuteError` (ArcPy Exception Class): https://pro.arcgis.com/en/pro-app/latest/arcpy/classes/executeerror.htm

    - The **`GetMessages()`** function returns the geoprocessing from a tool by a specified severity level. With the input parameter **`2`**, it will only return the Error message.

```
Python Syntax

GetMessages ({severity})
```

      ArcGIS Pro Help: https://pro.arcgis.com/en/pro-app/latest/arcpy/functions/getmessages.htm

    - The **`AddError()`** function sends the message on the tool dialog box. Though this is not a scrip tool, and you will not see the message.

```
Python Syntax

AddError (message)
```

      ArcGIS Pro Help: https://pro.arcgis.com/en/pro-app/latest/arcpy/functions/adderror.htm

    - The **`print()`** function sends the message to the Python editor *Shell* window.

14. Lines 36 – 48
    The exception block to handle other error.

    - The **`sys.exc_info()`** function returns a tuple of three values that give information about the exception that is currently being handled.
        - If no exception is being handled anywhere on the stack, a tuple containing three None values is returned. Otherwise, the values returned are (type, value, traceback).
        - Type gets the exception type of the exception being handled (a class object).

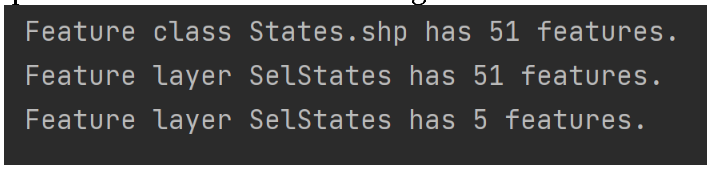
- `sys.exc_info()[0]`
- Value gets the exception parameter.
- `sys.exc_info()[1]`
- Traceback gets a traceback object, which encapsulates the call stack at the point where the exception originally occurred.
- `sys.exc_info()[2]`

- The `traceback.format_tb()` function returns a list of strings from the traceback message in format.

Tasks

1. Use Python IDE, type the above codes into the editor. Change the environment setting `arcpy.env.Workspace` to correspond with the path name in your computer. Save the script as **..\Practice10\Scripts\Practice10-1.py**.

2. Run the script. You should see the print out results as the following.



```
Feature class States.shp has 51 features.
Feature layer SelStates has 51 features.
Feature layer SelStates has 5 features.
```

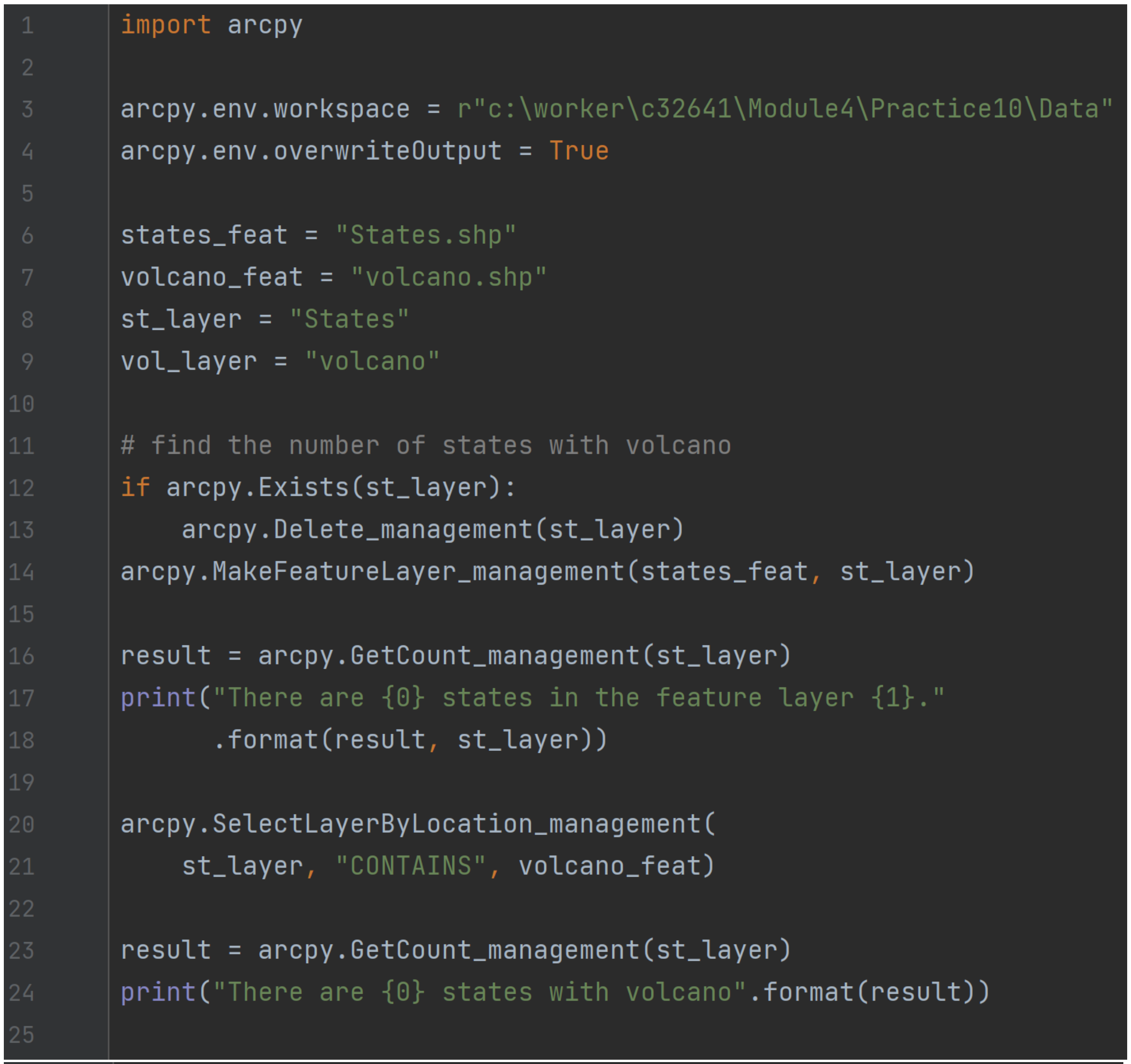
   - If you encounter any error, read the message printed in the *Shell* window and fix the problem.
   - If you did not encounter any error, try to change the script to create Python error and ArcGIS error, so you can observe the print out message, and practice the debugging skill.

3. Fix any bug and add comments for the script.

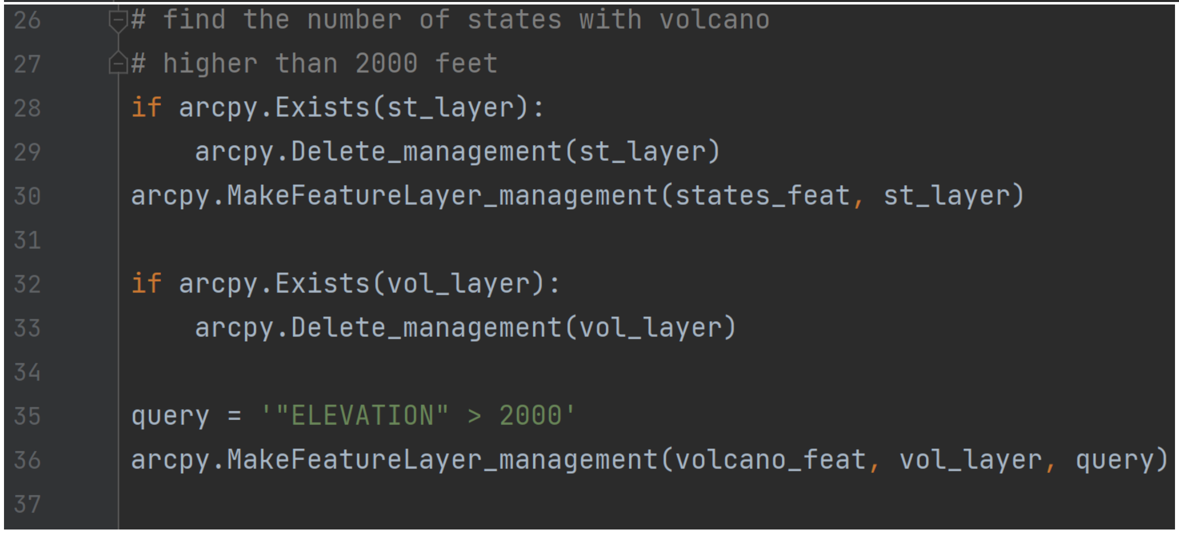
Part II: The combination of various tools for query

The script shows how to utilize the combination of the *Make Feature Layer*, *Select Layer By Attribute*, and *Select Layer By Location* tools to create feature layer.

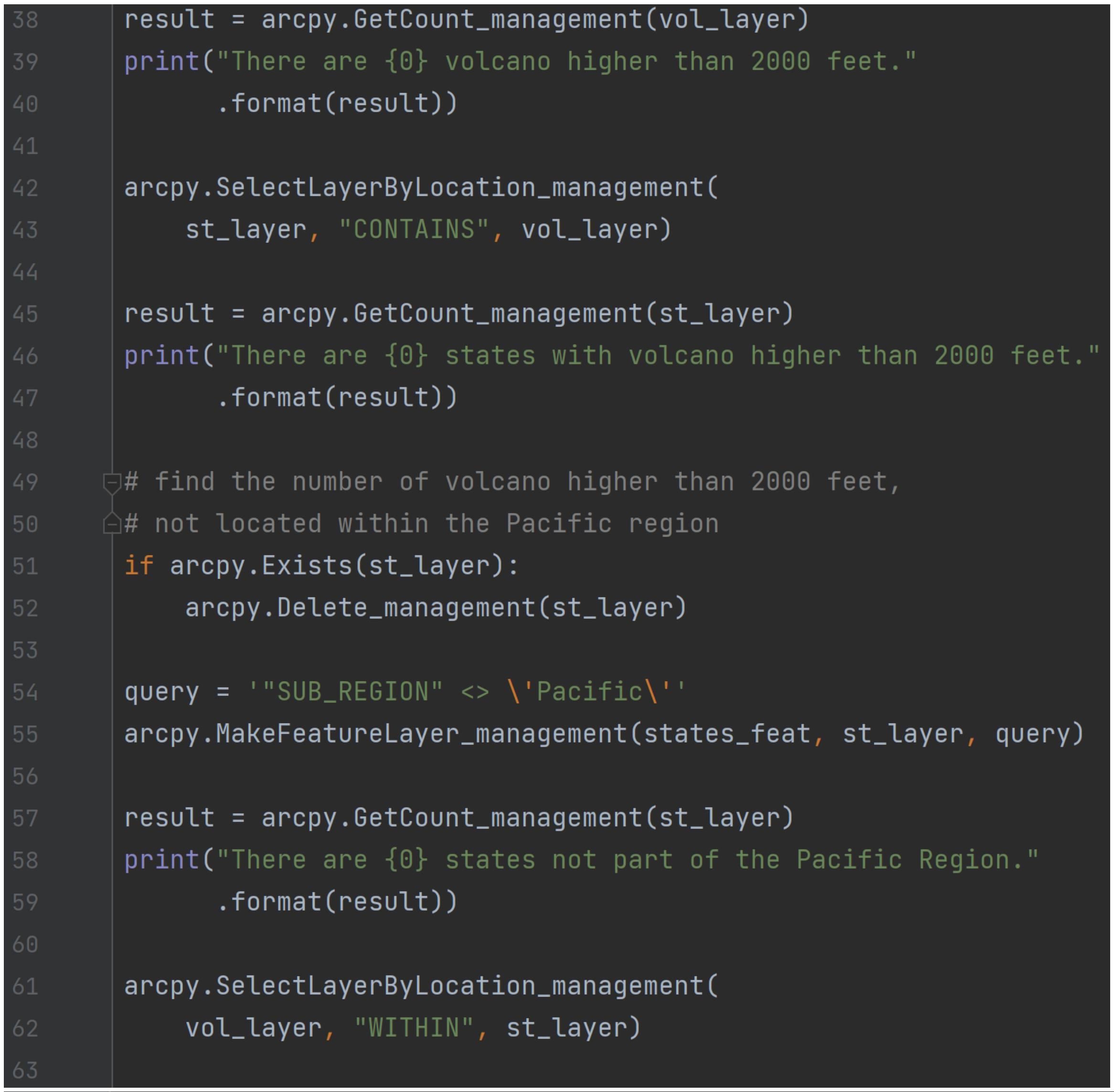
- Use the `MakeFeatureLayer` tool to limit the records when you create the feature layer.
- Utilize the `SelectLayerByLocation` tool retrieve the feature records based on the spatial relationship.
- Apply `SelectLayerByAttribute` tool on the feature layer to further confine the selection set.
- Use escape character for a single quote ('`\'`) and as line continuation character.



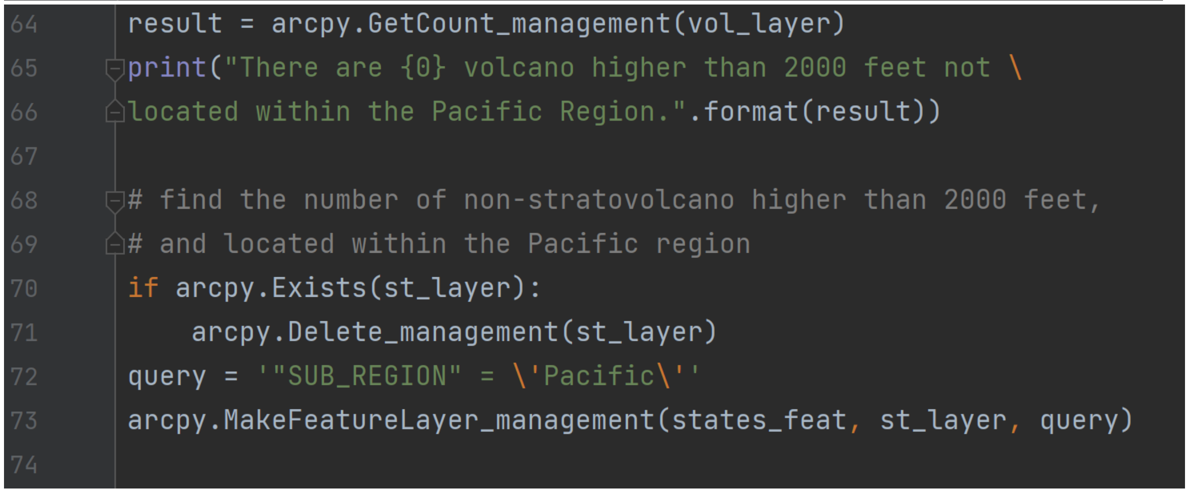
```python
import arcpy

arcpy.env.workspace = r"c:\worker\c32641\Module4\Practice10\Data"
arcpy.env.overwriteOutput = True

states_feat = "States.shp"
volcano_feat = "volcano.shp"
st_layer = "States"
vol_layer = "volcano"

# find the number of states with volcano
if arcpy.Exists(st_layer):
    arcpy.Delete_management(st_layer)
arcpy.MakeFeatureLayer_management(states_feat, st_layer)

result = arcpy.GetCount_management(st_layer)
print("There are {0} states in the feature layer {1}."
      .format(result, st_layer))

arcpy.SelectLayerByLocation_management(
    st_layer, "CONTAINS", volcano_feat)

result = arcpy.GetCount_management(st_layer)
print("There are {0} states with volcano".format(result))

```



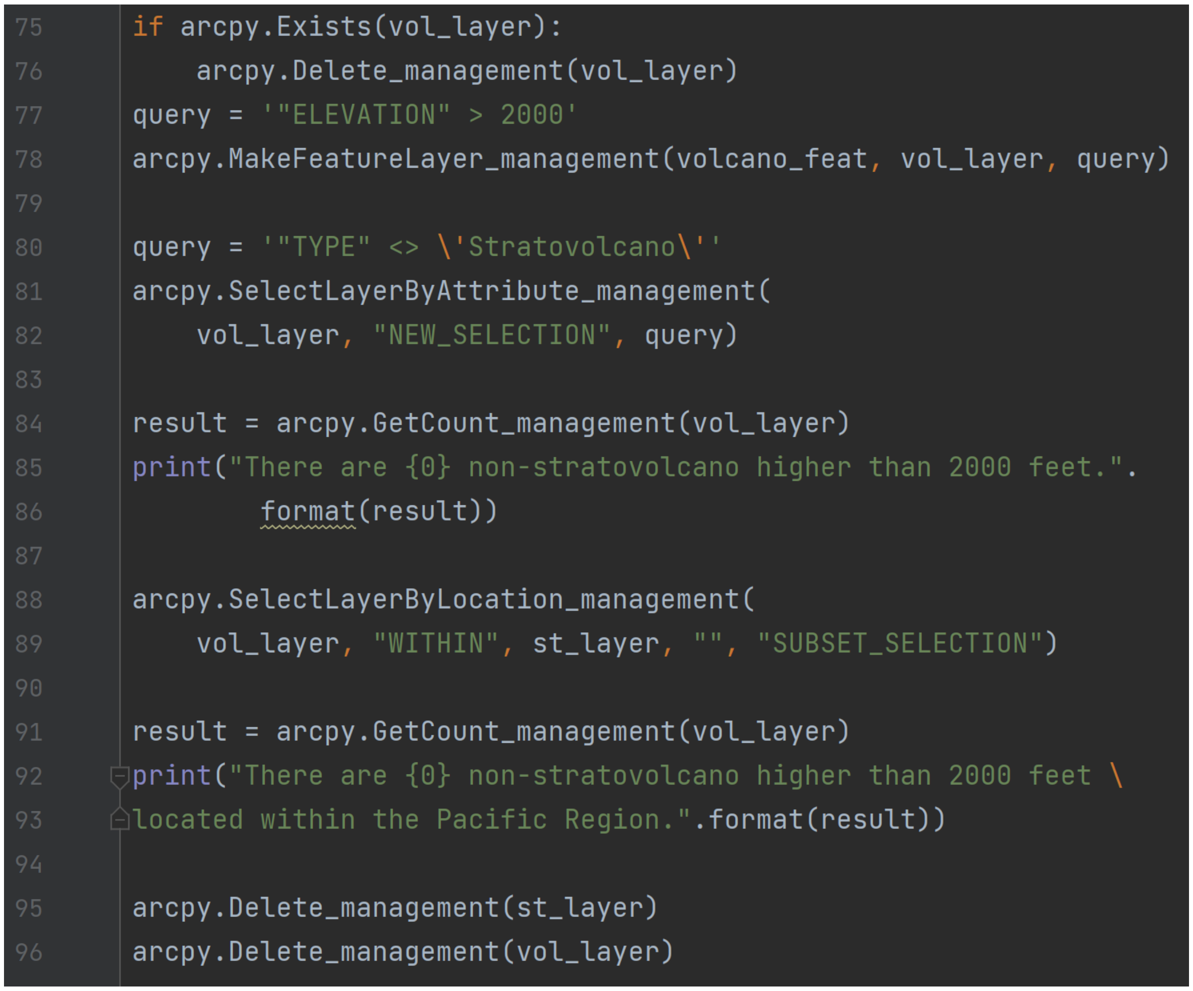
```python
# find the number of states with volcano
# higher than 2000 feet
if arcpy.Exists(st_layer):
    arcpy.Delete_management(st_layer)
arcpy.MakeFeatureLayer_management(states_feat, st_layer)

if arcpy.Exists(vol_layer):
    arcpy.Delete_management(vol_layer)

query = '"ELEVATION" > 2000'
arcpy.MakeFeatureLayer_management(volcano_feat, vol_layer, query)

```



```python
result = arcpy.GetCount_management(vol_layer)
print("There are {0} volcano higher than 2000 feet."
      .format(result))

arcpy.SelectLayerByLocation_management(
    st_layer, "CONTAINS", vol_layer)

result = arcpy.GetCount_management(st_layer)
print("There are {0} states with volcano higher than 2000 feet."
      .format(result))

# find the number of volcano higher than 2000 feet,
# not located within the Pacific region
if arcpy.Exists(st_layer):
    arcpy.Delete_management(st_layer)

query = '"SUB_REGION" <> \'Pacific\''
arcpy.MakeFeatureLayer_management(states_feat, st_layer, query)

result = arcpy.GetCount_management(st_layer)
print("There are {0} states not part of the Pacific Region."
      .format(result))

arcpy.SelectLayerByLocation_management(
    vol_layer, "WITHIN", st_layer)

```



```python
result = arcpy.GetCount_management(vol_layer)
print("There are {0} volcano higher than 2000 feet not \
located within the Pacific Region.".format(result))

# find the number of non-stratovolcano higher than 2000 feet,
# and located within the Pacific region
if arcpy.Exists(st_layer):
    arcpy.Delete_management(st_layer)
query = '"SUB_REGION" = \'Pacific\''
arcpy.MakeFeatureLayer_management(states_feat, st_layer, query)

```



```
75  if arcpy.Exists(vol_layer):
76      arcpy.Delete_management(vol_layer)
77  query = '"ELEVATION" > 2000'
78  arcpy.MakeFeatureLayer_management(volcano_feat, vol_layer, query)
79
80  query = '"TYPE" <> \'Stratovolcano\''
81  arcpy.SelectLayerByAttribute_management(
82      vol_layer, "NEW_SELECTION", query)
83
84  result = arcpy.GetCount_management(vol_layer)
85  print("There are {0} non-stratovolcano higher than 2000 feet.".
86          format(result))
87
88  arcpy.SelectLayerByLocation_management(
89      vol_layer, "WITHIN", st_layer, "", "SUBSET_SELECTION")
90
91  result = arcpy.GetCount_management(vol_layer)
92  print("There are {0} non-stratovolcano higher than 2000 feet \
93  located within the Pacific Region.".format(result))
94
95  arcpy.Delete_management(st_layer)
96  arcpy.Delete_management(vol_layer)
```

Code Explanations:

1. I did not place the `Try/Except` blocks for error handling in this script for saving space. You should use error handling block to help you, if needed.

2. Lines 1 – 9
   Import module and specify the workspace. Assign the variables (`states_feat`, `volcano_feat`) with feature class name and initialize variables (`st_layer`, `vol_layer`) with feature layer name.

3. Lines 11 – 24
   This block of codes shows how to find the number of states with volcanos.

   a) Lines 12 – 14
      First use the `Exists` and `Delete` functions to check the existence of the feature layer **States** and remove it if necessary. Then, create a feature layer **States** (variable `st_layer`) based on the feature class **States.shp** (variable `states_feat`) without query restriction.

b) Lines 16 – 18
   Report the number of features (records) in the feature layer **States** (variable **`st_layer`**). It should be the same number as the features in the feature class **States.shp** (variable **`states_feat`**).

c) Lines 20 – 21
   Utilize the **`SelectLayerByLocation`** function to select features in the feature layer **States** based whether there is any feature CONTAINS the features in the feature class **volcano.shp** (variable **`volcano_feat`**).

```
Python Syntax

arcpy.management.SelectLayerByLocation(in_layer, {overlap_type},
{select_features}, {search_distance}, {selection_type},
{invert_spatial_relationship})
```

ArcGIS Pro Help: https://pro.arcgis.com/en/pro-app/latest/tool-reference/data-management/select-layer-by-location.htm

- Notice that the **`SelectLayerByLocation`** function directly uses the feature class as the *"select_features"* parameter because we did not perform any query or operation on the feature class **volcano.shp** (variable **`volcano_feat`**). Therefore, there will not be any potential data lock.

d) Lines 23 – 24
   Report the number of selected features (records) in the feature layer **States** (variable **`st_layer`**), which should be the number of states contain any volcano within its boundary.

4. Lines 26 – 47
   This block of codes shows how to find the number of states with volcanos higher than 2000 feet within them.

   - It is essentially the same as the previous block, except we create a feature layer **volcano** (variable **`vol_layer`**) based on the selected feature from the feature class **volcano.shp** (variable **`volcano_feat`**).

   a) Lines 28 – 30
      Check the existence of the feature layer **States** (variable **`st_layer`**) and remove it if necessary. Create a new feature layer **States** (variable **`st_layer`**).

   b) Lines 32 – 33
      Check the existence of the feature layer **volcano** (variable **`vol_layer`**) and remove it if necessary.

c) Lines 35 – 36
Build a query expression to select the features with ELEVATION higher than 2000. Create a feature layer **volcano** (variable `vol_layer`) with the selected features from the feature class **volcano.shp** (variable `volcano_feat`) based on the query expression.

d) Lines 38 – 40
Report the number of selected features (records) in the feature layer **volcano** (variable `vol_layer`) , which should be the number of volcanos higher than 2000 feet.

e) Lines 42 – 43
Utilize the `SelectLayerByLocation` function to find the features in the feature layer **States** (variable `st_layer`) CONTAINS any feature from the feature layer **volcano** (variable `vol_layer`).

- Notice that the `SelectLayerByLocation` function uses the feature layer **volcano** (variable `vol_layer`) as the *"select_features"* parameter which will limit the selected features to only with the volcanos higher than 2000 feet.

f) Lines 45 – 47
Report the number of selected features (records) in the feature layer **States** (variable `st_layer`), which should be the number of states contain any volcano higher than 2000 feet within its boundary.

5. Lines 49 – 66
This block of codes shows how to find <u>the number of volcanos higher than 2000 feet but not within the Pacific Region</u>.

- You should notice that the input feature and select features are reversed in the `SelectLayerByLocation` function because we want to query about the **volcano** feature layer (variable `vol_layer`) based on its spatial relationship with the **States** feature layer (variable `st_layer`).

a) Lines 51 – 52
Check the existence of the feature layer **States** and remove it if necessary.

b) Lines 54 – 55
Build a query expression to select the features not belong to the Pacific Region. Create a feature layer **States** (variable `st_layer`) based on the query expression.

c) Lines 57 – 59
d) Report the number of selected features (records) in the feature layer **States** (variable `st_layer`), which should be the number of states, not belong to the Pacific Region.

- Notice that there is no code to create the feature layer for volcanos because we already create the feature layer of volcanos higher than 2000 feet in the previous block of code. In the previous block, the feature layer **volcano** (variable `vol_layer`) is used as the select features parameter in the `SelectLayerByLocation` function, and there is no other operation to change its value or selected set. Therefore, we can still use the same feature layer in this block of code.

e) Lines 61 – 62
Utilize the `SelectLayerByLocation` function to select features in the feature layer **volcano** (variable `vol_layer`) (the volcanos higher than 2000 feet) based whether the feature is located WITHIN the feature layer **States** (variable `st_layer`).

f) Lines 64 – 66
Report the number of selected features (records) in the feature layer **volcano** (variable `vol_layer`) which should be the number of volcanos higher than 2000 feet and located within the states not belong to the Pacific Region.

- Notice that the escape character (`\`) is used as line continuation character in the `print` function, so Python will recognize the statement continues to the next line.

6. Lines 68 – 93
This block of codes shows how to the combination of the *Make Feature Layer*, *Select Layer By Location*, and *Select Layer By Attribute* tools to perform a complex query.

- The goal is to find the number of non-stratovolcano type volcanos higher than 2000 feet within the Pacific Region.

- There is one attribute restriction (need to be the Pacific Region) on the feature layer **States** (variable `st_layer`), two attribute constraints (elevation higher than 2000 feet and not stratovolcano type) of the feature layer **volcano** (variable `vol_layer`), and a selection based on the spatial relationship between these two layers.

a) Lines 70 – 73
Check the existence of the feature layer **States** (variable `st_layer`) and remove it if necessary. Build a query expression to select the features belong to the Pacific Region. Create a new feature layer **States** (variable `st_layer`) based on the query expression.

b) Lines 75 – 78
Check the existence of the feature layer **volcano** (variable `vol_layer`) and remove it if necessary. Build a query expression to select the features with elevation higher than 2000. Create a new feature layer **volcano** (variable `vol_layer`) based on the query expression.

c) Lines 80 – 82

Build a query expression to select the features which are not Stratovolcano type. Utilize the `SelectLayerByAttribute` function to select features in the feature layer **volcano** (variable `vol_layer`) (the volcanos higher than 2000 feet) based whether the feature is Stratovolcano type.

- The `SelectLayerByAttribute` function uses the select type as NEW_SELECTION, which creates a subset of selection on the feature layer **volcano** (variable `vol_layer`) (the volcanos higher than 2000 feet). It will lead to different selection type when you apply the `SelectLayerByLocation` function later.

```
Python Syntax

arcpy.management.SelectLayerByAttribute(in_layer_or_view,
{selection_type}, {where_clause}, {invert_where_clause})
```

ArcGIS Pro Help: https://pro.arcgis.com/en/pro-app/latest/tool-reference/data-management/select-layer-by-attribute.htm

d) Lines 84 – 86
Report the number of selected features (records) in the feature layer **volcano** (variable `vol_layer`) which should be the number of non-stratovolcano type volcanos higher than 2000 feet.

e) Lines 88 - 89
Utilize the `SelectLayerByLoation` function to select features in the feature layer **volcano** (variable `vol_layer`) (the non- stratovolcano type volcanos higher than 2000 feet) based whether the feature is located within the Pacific Region (the feature layer **States**, (variable `st_layer`)).

- The select type of the `SelectLayerByLoation` function is SUBSET_SELECTION because the feature layer volcano contains a subset of selection from the `SelectLayerByAttribute` function. We want to select the features from this selected subset not the entire feature layer.

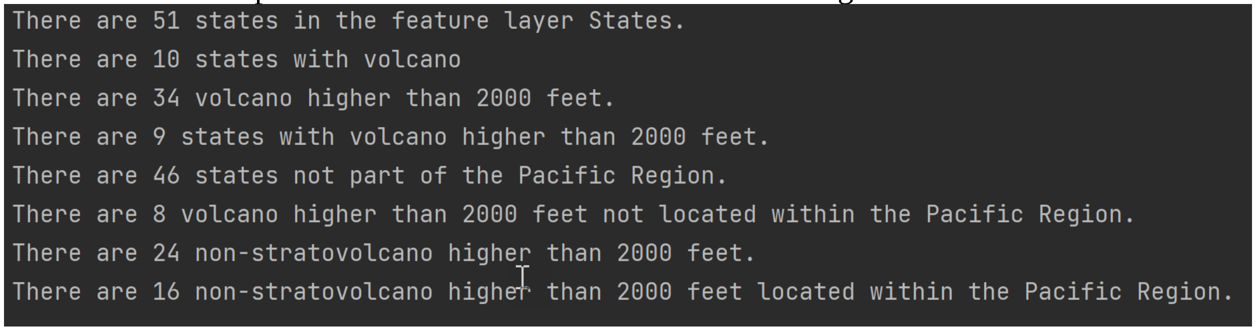
f) Lines 91 – 93
Report the number of selected features (records) in the feature layer **volcano**, which should be the number of non-stratovolcano type volcanos higher than 2000 feet located within the Pacific Region.

7. Lines 95 – 96
Good practice to free up the computer memory and avoid the data locks by deleting the in-memory feature layer.

Tasks

1. Use Python IDE, type the above codes into the editor. Change the environment setting `arcpy.env.Workspace` to correspond with the path name in your computer. Save the script as **..\Practice10\Scripts\Practice10-2.py**.

2. Run the script. You should see the results as the following in the interactive window.



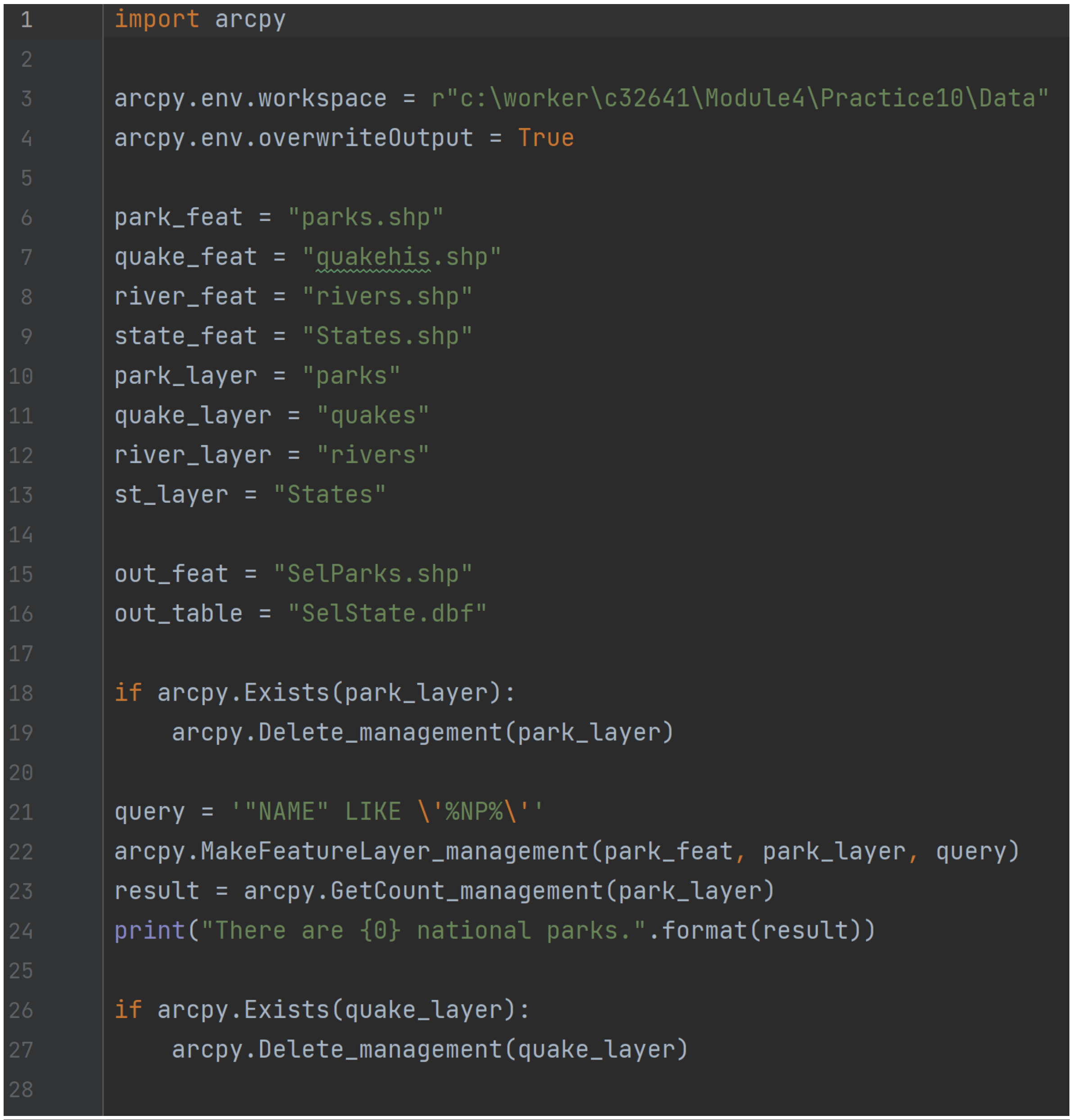
```
There are 51 states in the feature layer States.
There are 10 states with volcano
There are 34 volcano higher than 2000 feet.
There are 9 states with volcano higher than 2000 feet.
There are 46 states not part of the Pacific Region.
There are 8 volcano higher than 2000 feet not located within the Pacific Region.
There are 24 non-stratovolcano higher than 2000 feet.
There are 16 non-stratovolcano higher than 2000 feet located within the Pacific Region.
```

3. Fix any bug and add comments for the script.

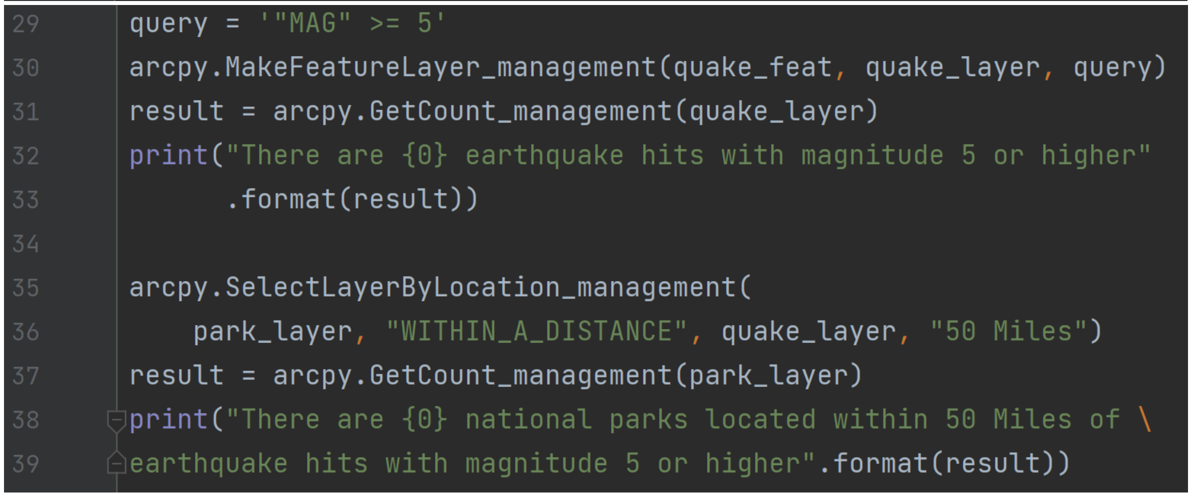
<u>Part III: Practice with various query expressions</u>

The script shows how more on the combination of queries, and then outputs the results as feature class and table

- Utilize <u>Search Distance</u> parameter in the `SelectLayerByLocation` tool to select features.
- Use the `CopyFeatures` tool to write the result into a new feature class.
- Apply `CopyRows` tool to copy the attribute table into a table.

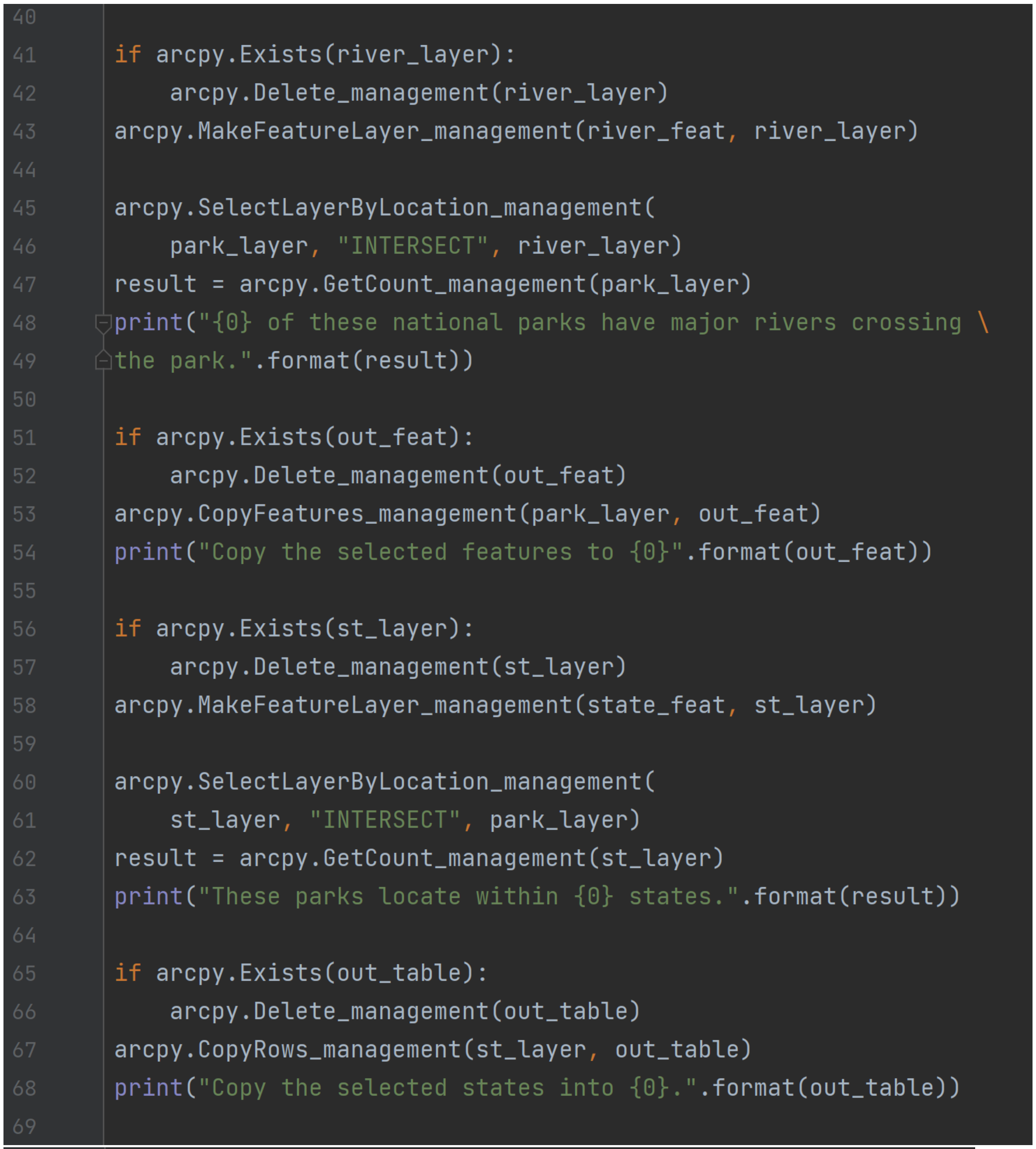


```python
import arcpy

arcpy.env.workspace = r"c:\worker\c32641\Module4\Practice10\Data"
arcpy.env.overwriteOutput = True

park_feat = "parks.shp"
quake_feat = "quakehis.shp"
river_feat = "rivers.shp"
state_feat = "States.shp"
park_layer = "parks"
quake_layer = "quakes"
river_layer = "rivers"
st_layer = "States"

out_feat = "SelParks.shp"
out_table = "SelState.dbf"

if arcpy.Exists(park_layer):
    arcpy.Delete_management(park_layer)

query = '"NAME" LIKE \'%NP%\''
arcpy.MakeFeatureLayer_management(park_feat, park_layer, query)
result = arcpy.GetCount_management(park_layer)
print("There are {0} national parks.".format(result))

if arcpy.Exists(quake_layer):
    arcpy.Delete_management(quake_layer)

```

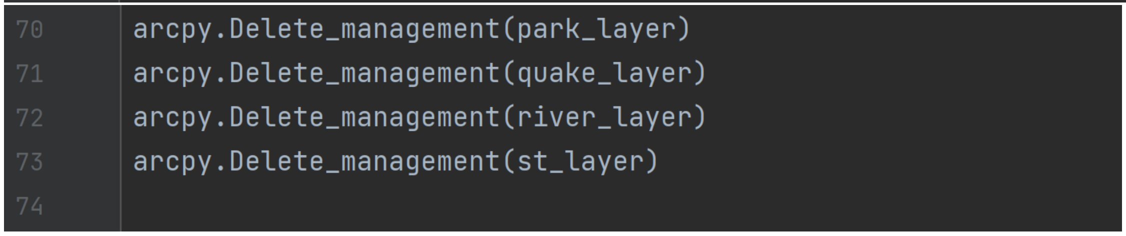


```python
query = '"MAG" >= 5'
arcpy.MakeFeatureLayer_management(quake_feat, quake_layer, query)
result = arcpy.GetCount_management(quake_layer)
print("There are {0} earthquake hits with magnitude 5 or higher"
      .format(result))

arcpy.SelectLayerByLocation_management(
    park_layer, "WITHIN_A_DISTANCE", quake_layer, "50 Miles")
result = arcpy.GetCount_management(park_layer)
print("There are {0} national parks located within 50 Miles of \
earthquake hits with magnitude 5 or higher".format(result))
```

Code Explanations:



```
if arcpy.Exists(river_layer):
    arcpy.Delete_management(river_layer)
arcpy.MakeFeatureLayer_management(river_feat, river_layer)

arcpy.SelectLayerByLocation_management(
    park_layer, "INTERSECT", river_layer)
result = arcpy.GetCount_management(park_layer)
print("{0} of these national parks have major rivers crossing \
the park.".format(result))

if arcpy.Exists(out_feat):
    arcpy.Delete_management(out_feat)
arcpy.CopyFeatures_management(park_layer, out_feat)
print("Copy the selected features to {0}".format(out_feat))

if arcpy.Exists(st_layer):
    arcpy.Delete_management(st_layer)
arcpy.MakeFeatureLayer_management(state_feat, st_layer)

arcpy.SelectLayerByLocation_management(
    st_layer, "INTERSECT", park_layer)
result = arcpy.GetCount_management(st_layer)
print("These parks locate within {0} states.".format(result))

if arcpy.Exists(out_table):
    arcpy.Delete_management(out_table)
arcpy.CopyRows_management(st_layer, out_table)
print("Copy the selected states into {0}.".format(out_table))

```



```
arcpy.Delete_management(park_layer)
arcpy.Delete_management(quake_layer)
arcpy.Delete_management(river_layer)
arcpy.Delete_management(st_layer)

```

Code Explanations:

1. Lines 1 – 16
   Import module and specify the workspace. Assign the variables (`park_feat`, `quake_feat`, `river_feat`, `state_feat`) with feature classes (**parks.shp**, **quakehis.shp**, **rivers.shp**, **States.shp**) and initialize variables (`park_layer`, `quake_layer`, `river_layer`, `st_layer`) with feature layer name. Initial two variables (`out_feat`, `out_table`) for the output feature class and the output table.

2. Lines 18 - 24
   Find the national parks based on the name with NP and create a new feature layer **parks** (variable `park_layer`).

   - The query expression uses `LIKE` operator and `%` wildcard character to bound the word NP. It means any string text in the NAME field contains the word NP will be selected. The escape character is use to escape the single quote.

```
query = '"NAME" LIKE \'%NP%\''
```

3. Lines 26 – 33
   Find the earthquake hits with magnitude 5 or higher and create a new feature layer **quakes** (variable `quake_layer`).

4. Lines 35 – 39
   Utilize the `SelectLayerByLoation` function to select features in the feature layer **parks** (variable `park_layer`) (only national parks) based whether the feature is located within 50 miles of earthquakes with magnitude 5 and higher (the feature layer **quakes** (variable `quake_layer`)).

   - The `SelectLayerByLoation` function uses the *Search Distance* parameter, which will generate a buffer around the *Select Features* (in this case, the feature layer **quake** (variable `quake_layer`)), then finds the overlapping park features.

```
arcpy.SelectLayerByLocation_management(
    park_layer, "WITHIN_A_DISTANCE", quake_layer, "50 Miles")
```

5. Lines 41 – 49
   Within these selected national parks (the feature layer **parks** (variable `park_layer`)), choose the ones with major rivers crossing through the park area.

   - The `SelectLayerByLoation` function uses INTERSECT as overlap type. The input feature layer is the selected subset of feature layer **parks** (variable `park_layer`) from previous `SelectLayerByLoation` function.

6. Lines 51 – 54
   Use the **`Exists`** to check the existence of the output feature class. If yes, then utilize the **`Delete`** function to delete it from the computer disk. If not, then just skip to the next line to copy the selected feature into the new feature class **SelParks.shp** (variable **`out_feat`**) with the **`CopyFeatures`** function.

```
Python Syntax

arcpy.management.CopyFeatures(in_features, out_feature_class,
{config_keyword}, {spatial_grid_1}, {spatial_grid_2},
{spatial_grid_3})
```

   ArcGIS Pro Help: https://pro.arcgis.com/en/pro-app/latest/tool-reference/data-management/copy-features.htm

7. Lines 56 – 63
   Create a new feature layer **States** (variable **`state_layer`**) based on the feature class **States.shp** (variable **`state_feat`**). Utilize the **`SelectLayerByLoation`** function to find the states overlap with any previous selected parks.

   - The input feature layer is still the selected subset of feature layer **parks** (variable **`park_layer`**) from previous **`SelectLayerByLoation`** function.

8. Lines 65 – 68
   Use the **`Exists`** to check the existence of the output table. If yes, then utilize the **`Delete`** function to delete it from the computer disk. If not, then just skip to the next line to copy the attribute table of the selected feature into the new table **SelStates** (variable **`out_table`**) with the **`CopyRows`** function.

```
Python Syntax

arcpy.management.CopyRows(in_rows, out_table, {config_keyword})
```

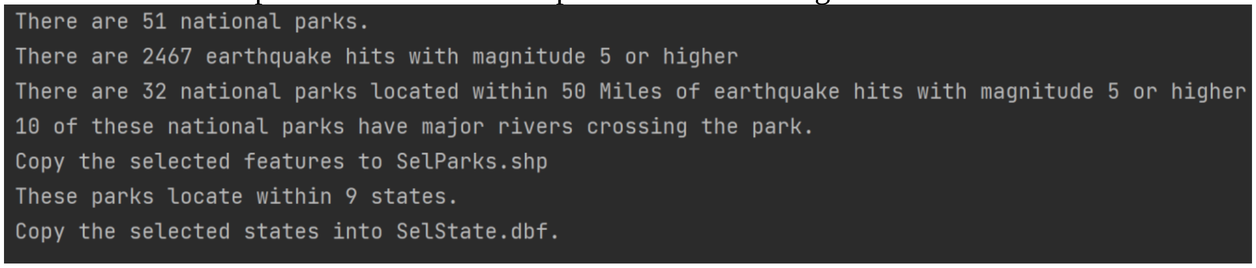
   ArcGIS Pro Help: https://pro.arcgis.com/en/pro-app/latest/tool-reference/data-management/copy-rows.htm

9. Lines 70 – 73
   Good practice to free up the computer memory and avoid the data locks by deleting the in-memory feature layer.

Tasks

1. Use Python IDE, type the above codes into the editor. Change the environment setting `arcpy.env.Workspace` to correspond with the path name in your computer. Save the script as **..\Practice10\Scripts\Practice10-3.py**.

2. Run the script. You should see the print as the following in the interactive window.



```
There are 51 national parks.
There are 2467 earthquake hits with magnitude 5 or higher
There are 32 national parks located within 50 Miles of earthquake hits with magnitude 5 or higher
10 of these national parks have major rivers crossing the park.
Copy the selected features to SelParks.shp
These parks locate within 9 states.
Copy the selected states into SelState.dbf.
```

   - There should also be a new feature class named **SelParks.shp** and a new table named **SelState.dbf** in your **Practice10\Data** folder.

3. Fix any bug and add comments for the script.

Turn In:

- Compress the scripts as a zip file for submission.
  - There should 3 scripts files.