Practice 11: Search Cursor

Objectives:

- Utilize cursor to search records in the vector feature class.
- Create query expression for {where_clause} in cursor syntax

Datasets:

Data folder

MOcity.shp	point shapefile of Missouri cities
MOcnty.shp	polygon shapefile of Missouri counties
Scripts folder	folder to store the scripts

Data Source:

Esri

Part I: Use the search cursor to retrieve the field value

In the lecture note, you already see the example to utilize **`ListFields()`** function to retrieve the field name and how to apply search cursor to get the value from a field. This script will show you the combination of **`ListFields()`** and search cursor with the loop to list the field name and field value. It also shows you how to utilize **`SHAPE@`** token to print out the geometric information.

```
import arcpy
feat_Class = r"c:\worker\c32641\Module4\Practice11\Data\MOcity.shp"

fields = arcpy.ListFields(feat_Class, "", "String")
fieldName = []
for fld in fields:
    fieldName.append(fld.name)
fieldName.append("SHAPE@XY")

with arcpy.da.SearchCursor(feat_Class, fieldName) as srCursor:
    for row in srCursor:
        i = 0
        for fld in fields:
            print("{0}: Value = {1}".format(fld.name, row[i]))
            i = i + 1
        print("XY coordinate = {0}".format(row[i]))

del srCursor
```

Code Explanations:

1. Lines 1 – 2
   Import **`arcpy`** module and assign the feature class into variable **`feat_Class`**.

2. Line 4
   Use the **`ListFields()`** function to create a list of field object and use the *"field_type"* parameter (the third argument) to restrict the list for only string type field.

   - The output is a list of fields, then you will need to retrieve the field name later.

```
Python Syntax

ListFields (dataset, {wild_card}, {field_type})
```

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

3. Lines 5 – 8
   Create a list of field names that you want to display as the output. In this case, the script will restrict the list of only string type field (from the previous step) and the XY coordinates of the point features.

   - Line 5 ➔ Declare an empty list object as the variable **`fieldName`**.

   - Lines 6 – 7 ➔ Iterate through the list of fields, as variable **`fields`**, add the field name (**`fld.name`**) into the list.
     a) The variable **`fld`** is for each field object in the list **`fields`**, and **`name`** (the name of the field) is the property of the field object.

   - Line 8 ➔ Add the **`SHAPE@`** token for XY coordinates into the list **`fieldName`**.
     a) The **`SHAPE@`** token is a geometry token can be used to access full geometry object. You will learn more about geometry object later, now you only need to know that **`SHAPE@XY`** represents the X, Y coordinates of point features.

4. Line 10
   Combine **`WITH`** statement with cursor creation to avoid the data lock.

   - Create a search cursor over the input feature class with only the fields listed in **`fieldName`**. Use the second argument (the variable **`fieldName`**) to restrict the number of fields in this cursor.

```
Python Syntax

SearchCursor (in_table, field_names, {where_clause},
{spatial_reference}, {explode_to_points}, {sql_clause},
{datum_transformation})
```

ArcGIS Pro Help: https://pro.arcgis.com/en/pro-app/latest/arcpy/data-access/searchcursor-class.htm

5. Lines 11 – 16
   Create nested **FOR** loops. The outer **FOR** loop iterates through the record in the feature class **feat_Class**, and the inner **FOR** loop iterates through the string-type attribute fields in each record **fields**.

   - Line 12 ➔ Create an outer **FOR** loop to iterate each record in the cursor until the end of the record.

     a) The variable **row** is the current tuple assessed through the cursor.
     b) The variable **srCursor** is for the search object.

   - Line 13 ➔ Create a counting variable **i** to keep track on how many fields in the list **fieldName**. Initialize it as 0.

     a) It is necessary to keep track on the number of fields in the list because we will need to use it as index position of each field in the list to retrieve its value.

   - Line 14 ➔ Create an inner **FOR** loop to iterate each field (a field object) in the list until the end of the list.

     a) The variable **fld** is for the field object.

   - Line 15 ➔ Print out of the field name and the field value.

     a) **row[i]** is the field value. Remember the advantage of using **arcpy.da** cursor is that you can directly access the field value based in the index position in the list from creating the cursor.

   - Line 16 ➔ Increase the counting variable by one since you already print out the first variable. This is the end of the inner **FOR** loop.

   - Line 17 ➔ After executing the inner **FOR** loop to print out all field values as the string type fields, then print out the XY coordinates.

6. Line 18

Delete the search cursor **srCursor**, though search cursor does not create lock on the data, it is always a good practice to delete cursor.

Tasks

1. Use Python IDE, type the above codes into the editor. Change the **feat_class** variable setting to correspond with the path name in your computer. Save the script as **..\Practice11\Scripts\Practice11-1.py**.

2. Run the script. You should see the results similar as the following, but much more, in the interactive window.

```
NAME: Value = Poplar Bluff
STATEABB: Value = US-MO
COUNTRY: Value = USA
XY coordinate = (731314.3596304352, 4071301.4163697166)
NAME: Value = Sikeston
STATEABB: Value = US-MO
```

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

Part II: Use the search cursor to perform analysis based on the field value

This script finds the average population for counties in a dataset. To find the average, the script needs to divide the total population by the number of counties. Then loop through each record and keeps a running total of the population and the count of records. Once all the records have been read, only one line of division is necessary to find the average.

```
1   import arcpy
2
3   inputFC = r"c:\worker\c32641\Module4\Practice11\Data\MOcnty.shp"
4   fieldList = ["NAME", "POP2000"]
5
6   arcpy.env.workspace = arcpy.Describe(inputFC).path
7   feat_Class = arcpy.Describe(inputFC).file
8
9   average = 0
10  totalPopulation = 0
11  recordCount = 0
12
13  with arcpy.da.SearchCursor(feat_Class, fieldList) as srCursor:
14      for row in srCursor:
15          print("{0} has population {1}".format(row[0], row[1]))
16          totalPopulation = totalPopulation + row[1]
17          recordCount = recordCount + 1
18
19  average = totalPopulation / recordCount
20  print("Average population for a county is {0}".format(average))
21
22  del srCursor
```

Code Explanations:

1. Lines 3 – 4
   Assign the feature class and the list of field name into variables.

   - The **`fieldList`** limits the number of attribute fields in the cursor to only for "`NAME`" and "`POP2000`". The "`NAME`" field has the index position in the list as `[0]`. The "`POP2000`" field has the index position as `[1]`.
   - Once the script run, you can change these two variables as user input parameter by **`arcpy.GetParameterAsText()`** to create script tool. It is not required for this practice, just why it is a good practice to list and assign variables at the beginning of the script.

2. Lines 6 – 7
   Retrieve the file path of the feature class and assign it as the workspace **`arcpy.env.workspace`**. Retrieve the file name and assign it to the feature class variable **`feat_class`**.

```
Python Syntax

Describe (value, {datatype})
```

ArcGIS Pro Help: https://pro.arcgis.com/en/pro-app/latest/arcpy/data-access/describe.htm

- The **path** property of the `Describe` object will provide the file path of the input feature class.
- The **file** property of the `Describe` object will return the file name of the input feature class.

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

3. Lines 9 – 11
   Initial variables.

   - The variable **average** is for the average population
   - The variable **totalPopulation** is for the current total population
   - The variable **recordCount** is for the count of records.

4. Line 13
   Combine **WITH** statement with cursor creation to avoid the data lock. Create a search cursor over the input feature class with only the fields listed in **fieldList**.

5. Lines 14 – 17
   Iteration through the records in the search cursor **srCursor**.

   - Line 14 ➔ Create a **FOR** loop to iterate each record in the cursor until the end of the record.

   - Line 15 ➔ Print out the message for the population of each state.

     a) In this example, **{0}** will be replaced by the first value in the **format()**, which is **row[0]**. **row[0]** is value of the first item in the **fieldList**, which is "NAME" as state name.
     b) **{1}** will be replaced by the second value in the **format()**, which is **row[1]**. **row[1]** is value of the second item in the **fieldList**, which is "POP2000" as population.

   - Line 16 ➔ Calculate the total population by adding the population of the current record into the variable **totalPopulation**.

     a) **row[1]** is value of the second item in the **fieldList**, which is "POP2000" as population.

- Line 17 ➔ Add one into the record count variable **`recordCount`**.

6. Lines 19 – 20
   Calculate the average population of the states and print out the message.

Tasks

1. In the Python editor *Script* window, type the above codes into the window. Change the **`inputFC`** variable setting to correspond with the path name in your computer. Save the script as **..\Practice11\Scripts\Practice11-2.py**.

2. Run the script. You should see the results similar as the following, but much more, in the interactive window.

```
Taney has population 39703
Ozark has population 9542
McDonald has population 21681
Dunklin has population 33155
Pemiscot has population 20047
Average population for a county is 48654.00869565218
```

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

Part III: Retrieving records with a spatial query

This script shows how to perform a spatial query to retrieving the specific records. ArcGIS utilizes the concept of temporary feature layers to represent in-memory sets of records from a feature class. It can avoid the data lock conflicts between the users, and save the physical storage space to create a new feature class to hold the query results. The *Make Feature Layer* tool creates a feature layer from some or all of the records in a feature class. A SQL expression can be applied in running *Make Feature Layer* to narrow down the records included in the feature layer based on attributes. Then, subsequently use *Select Layer By Location* tool to narrow down the records in the feature layer based on some spatial criteria.

The following script will open a search cursor on a county in interest (e.g., **Ozark**) and all counties bordering it.

```
1   import arcpy
2
3   county = "Ozark"
4   inputFC = r"c:\worker\c32641\Module4\Practice11\Data\MOcnty.shp"
5   fieldName = "NAME"
6
7   arcpy.env.workspace = arcpy.Describe(inputFC).path
8
9   arcpy.MakeFeatureLayer_management(inputFC, "AllCntyLyr")
10
11  query = "\"{0}\" = \'{1}\'".format(fieldName, county)
12  arcpy.MakeFeatureLayer_management(inputFC, "SelCntyLyr", query)
13
14  arcpy.SelectLayerByLocation_management(
15      "AllCntyLyr", "BOUNDARY_TOUCHES", "SelCntyLyr")
16
17  print("The neighboring counties of {0} are: ".format(county)),
18  with arcpy.da.SearchCursor("AllCntyLyr", fieldName) as srCursor:
19      for row in srCursor:
20          if row[0] != county:
21              print (row[0]),
22      print("\n")
23
24  arcpy.Delete_management("AllCntyLyr")
25  arcpy.Delete_management("SelCntyLyr")
26  del srCursor
```

Code Explanations:

1. Lines 3 – 5
   Assign the parameters into variables.

   - The variable **`county`** is for the county in question, which should be set as string data type in the tool dialog.
   - The variable **`inputFC`** is the input feature class for query, which should be set as feature class data type in the tool dialog.
   - The variable **`fieldName`** is the attribute field stored the name of the counties in the input feature class.

2. Line 7

Retrieve the file path of the feature class and assign it as the workspace `arcpy.env.workspace`.

3. Line 9
   Make a feature layer of the input feature class without restriction. Name the feature layer as **AllCntyLyr**.

4. Line 11
   Create query expression to select only the county name (variable `fieldName`) as **Ozark** (variable `county`).

   - In this example, we utilize Python string `format()` method to make code simpler, but a little difficult to read at first.

     - The query expression in ArcGIS field calculator should need to look like this:
       ```
       "NAME" = 'Ozark'
       ```

     - There need to be double quote surrounding the field name, and the single quote surrounding the field value.

     - However, you can't directly place the variables within double quote or single quote. Once the variable being placed inside the quote, it will be treated as string. Therefore, you will need to use single quote surrounding the double quote for making it as string, keep the variable as variable, use double quote surrounding the single quote. Use "+" to concatenate the strings and variables.

       ```
       '"' + str(fieldName) + '" = ' + "'" + str(county) + "'"
       ```

     - Use Python string `format()` method with escaping character `\`, we can create the exact query expression with less quote marks. Place the escaping character `\` in front of quote mark, Python will treat the quote mark as string, not special character.
       ```
       "\"{0}\" = \'{1}\'".format(fieldname, county)
       ```
       

5. Line 12
   Make a feature layer of the input feature class only to the chosen county based on the query expression. Name the feature layer as **SelCntyLyr**.

6. Lines 14 – 15
   Use the `SelectLayerByLocation()` function to select any record/feature in the feature layer **AllCntyLyr** has touched the boundary with the records/features in the feature layer **SelCntyLyr**.

7. Line 17

Print out a message first to explain the following results.

- Notice that there is a **comma** at the end of `print()` function, it is called as trailing comma. It will stop Python `print()` to create a new line after this message, so the next message will be in the same line as this one.

8. Line 18
   Combine `WITH` statement with cursor creation to avoid the data lock. Create a search cursor over the input feature class with only the field listed in `fieldName`.

   - When you work on any function with query expression, it will be best to utilize the print statement and `GetCount_management()` to make sure that you select the correct feature records.

9. Line 19 – 21
   Create a `FOR` loop to iterate the search cursor over the feature layer **AllCntyLyr** and print out the value in the field. The script will exit the loop when there is no more record.

   - `row[0]` is value of the first item in the `fieldName`, which is "NAME" as county name.

   - Lines 20 – 21 ➔ Because ArcGIS *Select By Location* function will select all counties touch the boundary of **Ozark** including **Ozark**, we will need to remove **Ozark** from the print out message.

     - Use `IF` statement to determine whether the county name is Ozark or not. If not, then print out the county name. Notice the trailing comma, it will make sure all print out are in the same line.

```
if row[0] != county:
    print (row[0]),
```

10. Line 21
    After all county names are printed out (end of the `FOR` loop), then create a new line.

11. Lines 24 – 26
    Delete the feature layer and the search cursor.

Tasks

1. In the Python editor *Script* window, type the above codes into the window. Change the `inputFC` variable setting to correspond with the path name in your computer. Save the script as **..\Practice11\Scripts\Practice11-3.py**.

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

```
The neighboring counties of Ozark are:
Douglas
Howell
Taney
```

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.