

Exercise 11: Working with List Functions

The list functions on ArcPy return Python list objects which contain a collection or list of objects. The individual objects in a list can be accessed through typical Python list access methods. The type of object returned by these methods depends upon the method that is called.

At the end of this exercise you will have learned the following:

- Use various List functions to return lists of GIS data including fields and feature classes
- Use the List functions in conjunction with geoprocessing tools

Step 1: List Fields in a Feature Class

In this step you will create a Python script that lists all the fields in the Hospitals feature class.

- Open PythonWin and create a new script.
- Insert comments including your name, date, and the purpose of this script.
- Save the script with the name ListFields.py.
- Import the ArcPy module

```
import arcpy
```

- Create a try block and set the workspace as seen in the code below

```
import arcpy
try:
    arcpy.env.workspace = "C:\\Users\\Me\\Desktop\\GIS Programming\\Training\\Data"
```

You may need to change the workspace path to another location based on where you would like to save the data.

- Use the ListFields function to return all the fields from a feature class

```
import arcpy
try:
    arcpy.env.workspace = "C:\\Users\\Me\\Desktop\\GIS Programming\\Training\\Data"
    fieldList = arcpy.ListFields("Hospitals.shp")
```


- Loop through all the fields and print out the name

```
import arcpy
try:
    arcpy.env.workspace = "C:\\Users\\Me\\Desktop\\GIS Programming\\Training\\Data"
    fieldList = arcpy.ListFields("Hospitals.shp")
    for fld in fieldList:
        print "%s is a type of %s with a length of %i" % (fld.name, fld.type, fld.length)
```

- Add the except block to check handle any errors

```
import arcpy
try:
    arcpy.env.workspace = "C:\\Users\\Me\\Desktop\\GIS Programming\\Training\\Data"
    fieldList = arcpy.ListFields("Hospitals.shp")
    for fld in fieldList:
        print "%s is a type of %s with a length of %i" % (fld.name, fld.type, fld.length)
except:
    print arcpy.GetMessages()
```

- Save and run the script. You should see the following output.



```
Python
FID is a type of OID with a length of 4
Shape is a type of Geometry with a length of 0
FID_ is a type of Integer with a length of 9
AREA is a type of Double with a length of 13
PERIMETER is a type of Double with a length of 13
CARE_ is a type of Double with a length of 10
CARE_ID is a type of Double with a length of 10
NAME is a type of String with a length of 18
ADDRESS is a type of String with a length of 40
NO_OF_BEDS is a type of SmallInteger with a length of 2
NO_DOCTORS is a type of SmallInteger with a length of 3
NURSING is a type of SmallInteger with a length of 3
CLIN_SUPP is a type of SmallInteger with a length of 3
```

Step 2: List all Feature Classes in a Workspace

The ListFeatureClasses function in ArcPy can be used to return a list of all feature classes in a workspace. In this step you will write a Python script that lists all the feature classes in our exercise data directory.

- Open PythonWin and create a new script.
- Insert comments including your name, date, and the purpose of this script.
- Save the script and give it the name ListFeatureClasses.py.
- Import the ArcPy module

```
import arcpy
```

Create a try block and set the workspace as seen in the code below

```
import arcpy
try:
    arcpy.env.workspace = "C:\\Users\\Me\\Desktop\\GIS Programming\\Training\\Data"
```

You may need to change the workspace path to another location based on where you would like to save the data.

- Use the ListFeatureClasses function on ArcPy to list all feature classes. ListFeatureClasses has two optional arguments that can be passed into the function that will serve to limit the returned list. The first optional argument is a wildcard that can be used to limit the feature classes returned based on name, and the second optional argument can be used to limit the feature classes returned based on data type (point, line, polygon, etc). In this example we want to return all polygon feature classes so we'll apply a data type delimiter.

```
import arcpy
try:
    arcpy.env.workspace = "C:\\Users\\Me\\Desktop\\GIS Programming\\Training\\Data"
    fcList = arcpy.ListFeatureClasses("", "Polygon")
```

- Loop through all feature classes in the list and print the name

```
import arcpy
try:
    arcpy.env.workspace = "C:\\Users\\Me\\Desktop\\GIS Programming\\Training\\Data"
    fcList = arcpy.ListFeatureClasses("", "Polygon")
    for fc in fcList:
        print fc
```

- Add the except code block to handle any errors.

```
import arcpy
try:
    arcpy.env.workspace = "C:\\Users\\Me\\Desktop\\GIS Programming\\Training\\Data"
    fcList = arcpy.ListFeatureClasses("", "Polygon")
    for fc in fcList:
        print fc
except:
    print arcpy.GetMessages()
```

- Save and run the script. The script should print out each of the polygon feature classes to the Interactive Window similar to the output you see below.

```
Communities.shp
```

Step 3: Perform an Operation on the Returned List

The list methods are typically used when you need to perform some type of operation with the list of returned objects. For example, perhaps you'd like to add a new field to all the feature classes returned using the ListFeatureClasses method or convert the data from a personal geodatabase to an enterprise (ArcSDE) geodatabase. In step 3 we'll add to the script we created in step 2 by adding a new field to each of the *polygon* feature classes returned from the ListFeatureClasses method.

- Use the ListFeatureClasses.py file that you created in the last step. Save a new file by clicking File → Save As and name the file ListFeatureClasses_AddField.py.
- Alter your try: block so that it appears as follows:

```
import arcpy
try:
    arcpy.env.workspace = "C:\\Users\\Me\\Desktop\\GIS Programming\\Training\\Data"
    fcList = arcpy.ListFeatureClasses("", "Polygon")
    for fc in fcList:
        print "Adding field to " + fc
        arcpy.AddField_management (fc, "Edited by", "text", "25")
except:
    print arcpy.GetMessages()
```

Inside the for loop we are touching each of the polygon classes and using the ArcToolbox AddField tool to add a field called 'EditedBy' to each feature class. This field could ultimately contain the name of the person who last edited the row.

- Save and run the script.
- Open ArcCatalog or ArcMap and examine the polygon feature classes in the workspace directory to validate that the EditedBy field has been added to each dataset.