Python for ArcGIS

Part 2

Ahmad Aburizaiza
Data Services Group - GMU Libraries
Spring 2016

OOP - from previous lecture

To import a class from a module, type in:

```
from moduleName import ClassName
```

OR

```
from moduleName import*
```

OR

```
import moduleName
```

To import a class from a module from a package, type in:

```
from packageName.moduleName import ClassName
```

OR

```
from packageName.moduleName import*
```

OR

```
import packageName.moduleName
```

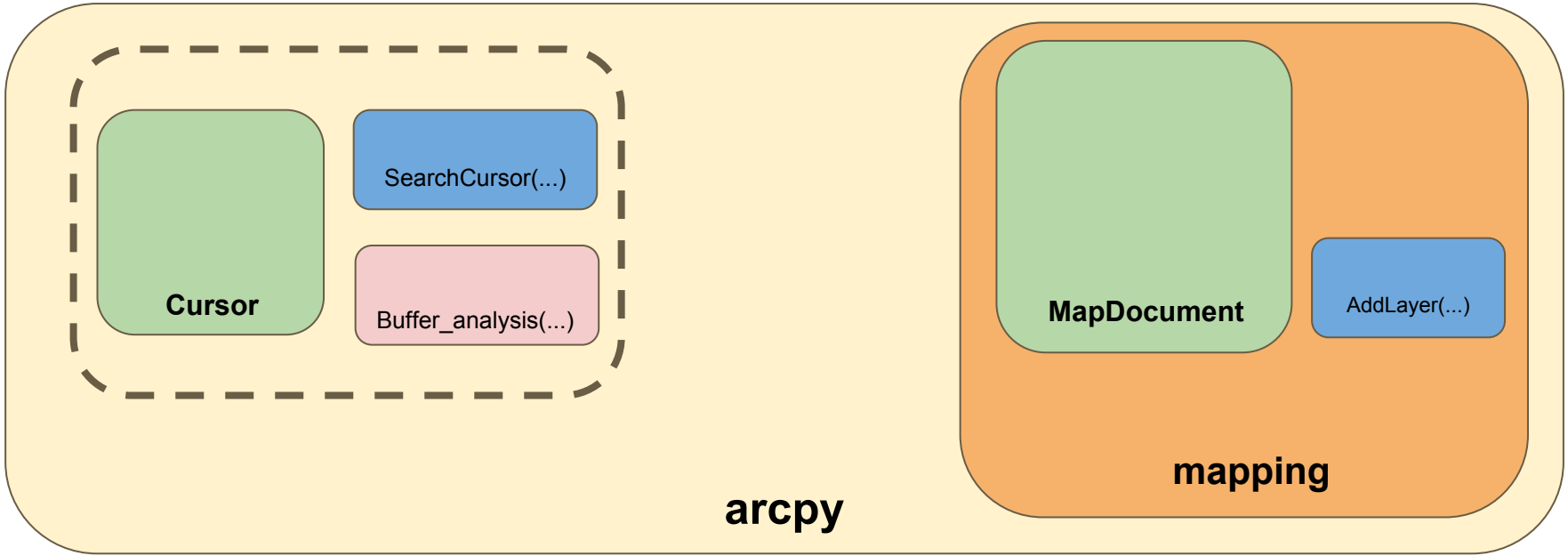
What is arcpy?

- arcpy is a Python package structured differently than common Python packages.
- You can find classes under the arcpy package directly NOT stored in modules.
- You can also find functions under the arcpy package directly and NOT stored in modules. Some of these functions are referred to in arcpy as tools.
- Tools vary in license level. They also output results while regular functions don't "this is particular to arcpy".

More About arcpy

- The reason of this uncommon structure is that ESRI uses classes and tools from their C++ ArcObjects library.
- I am mentioning this to avoid confusion because other Python packages follow the common structure we explained before → class inside module → module inside package.
- Again, you can run Python code in ArcMap, ArcCatalog, and IDLE.
- In ArcMap and ArcCatalog, the style is a shell similar to IDLE's shell window.
- To run scripts, IDLE can create separate (.py) files.

arcpy Uncommon Structure



What to Cover Next?

From the mapping module under the arcpy package:

- The MapDocument class.
- The Layer class.
- The ListLayers(...) function.
- The AddLyer(...) function.

Tools “functions” under arcpy package as toolsets inside toolboxes. We will use tools from analysis toolboxes:

- The Clip_analysis(...) tool from analysis toolset.
- The Buffer_analysis(...) tool from analysis toolset.
- The Intersect_analysis(...) tool from analysis toolset.

Documentation Link

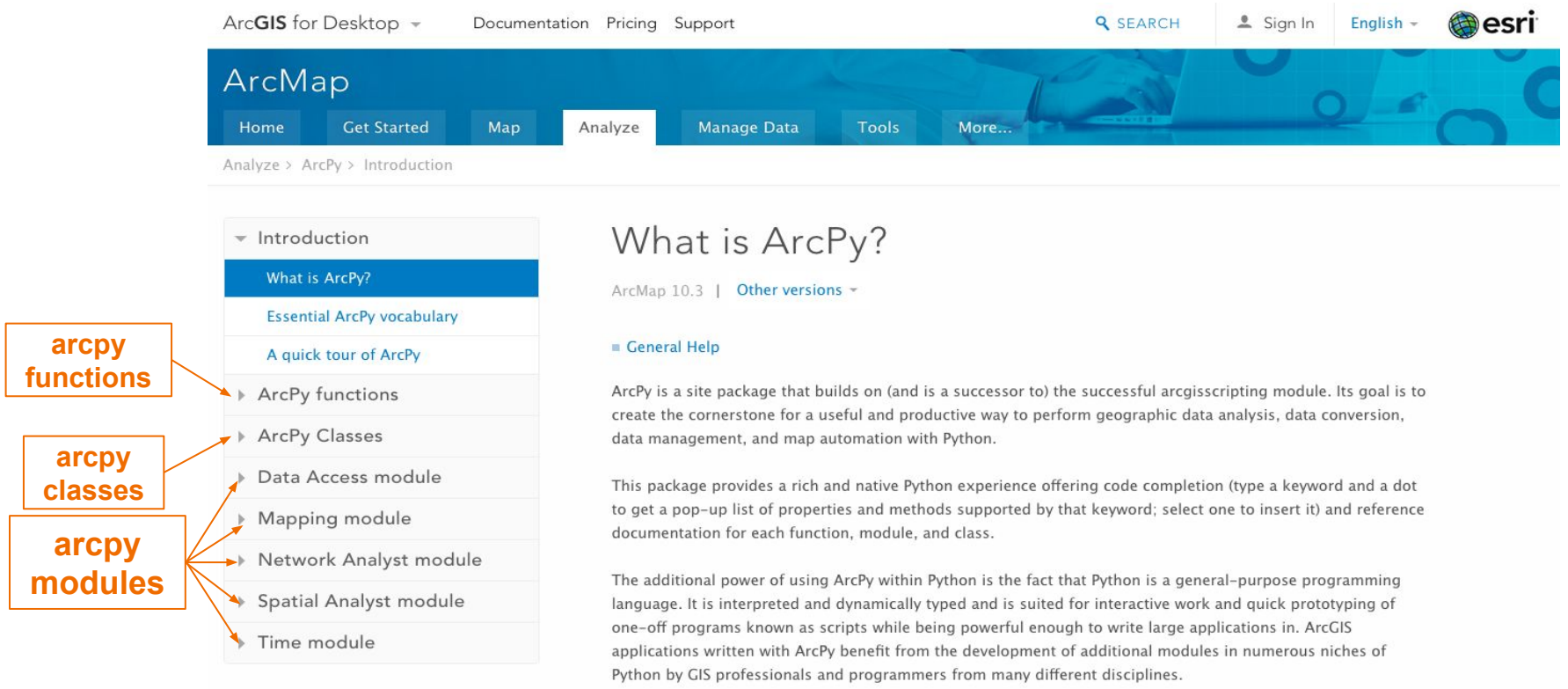
For arcpy modules, classes, and functions:

- Go to:
  http://desktop.arcgis.com/en/arcmap/10.3/analyze/arcpy/what-is-arcpy-.htm

For the analysis toolbox:

- Go to:
  http://desktop.arcgis.com/en/arcmap/10.3/tools/analysis-toolbox/an-overview-of-the-analysis-toolbox.htm

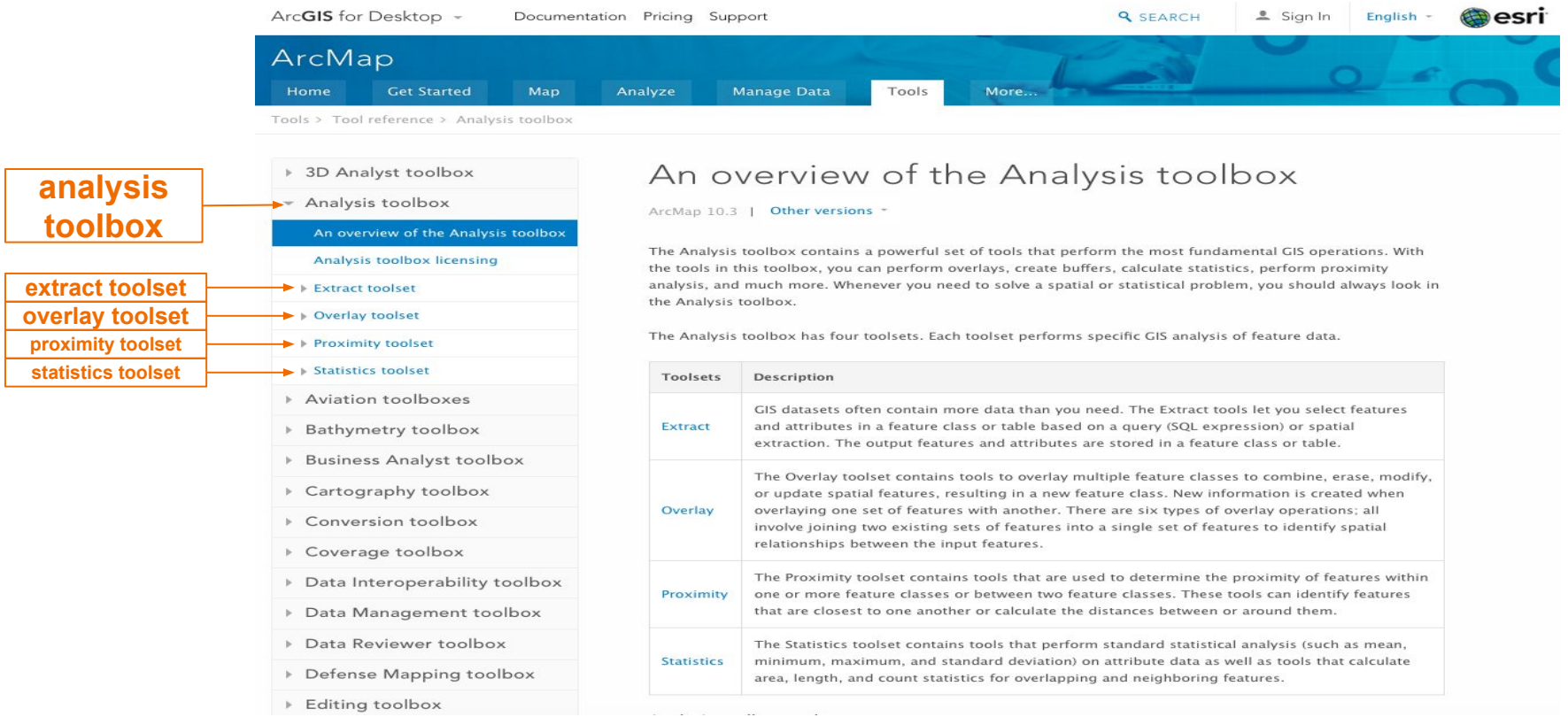
First Link - arcpy modules, classes, and functions



First Link - arcpy modules, classes, and functions

1. Click on “Mapping module” → “Classes” → “MapDocument”
2. This is the documentation for the MapDocument class.
3. Click the browser’s back button → “Layer”.
4. Now this is the documentation of the Layer class.
5. Try to find the function ListTimeZones() in the time module.

Second Link - analysis toolbox



Second Link - analysis toolbox

1. Click on “Extract toolset”.
2. The tools: Clip, Select, Split, and Table Select appears.
3. If you click on any other toolset, you will see its tools listed.
4. If you click on any tool, you will see its detailed documentation.

MapDocument Class Access

arcpy package → mapping module → MapDocument class

Create a MapDocument Object

```
import arcpy

mD = arcpy.mapping.MapDocument('C:/Users/Someuser/Desktop/project.mxd')
mD2 = arcpy.mapping.MapDocument('C:/Users/Someuser/Documents/city.mxd')
mD3 = arcpy.mapping.MapDocument('CURRENT')
```

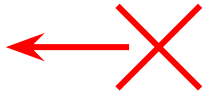
Properties

- Some properties you can read/write

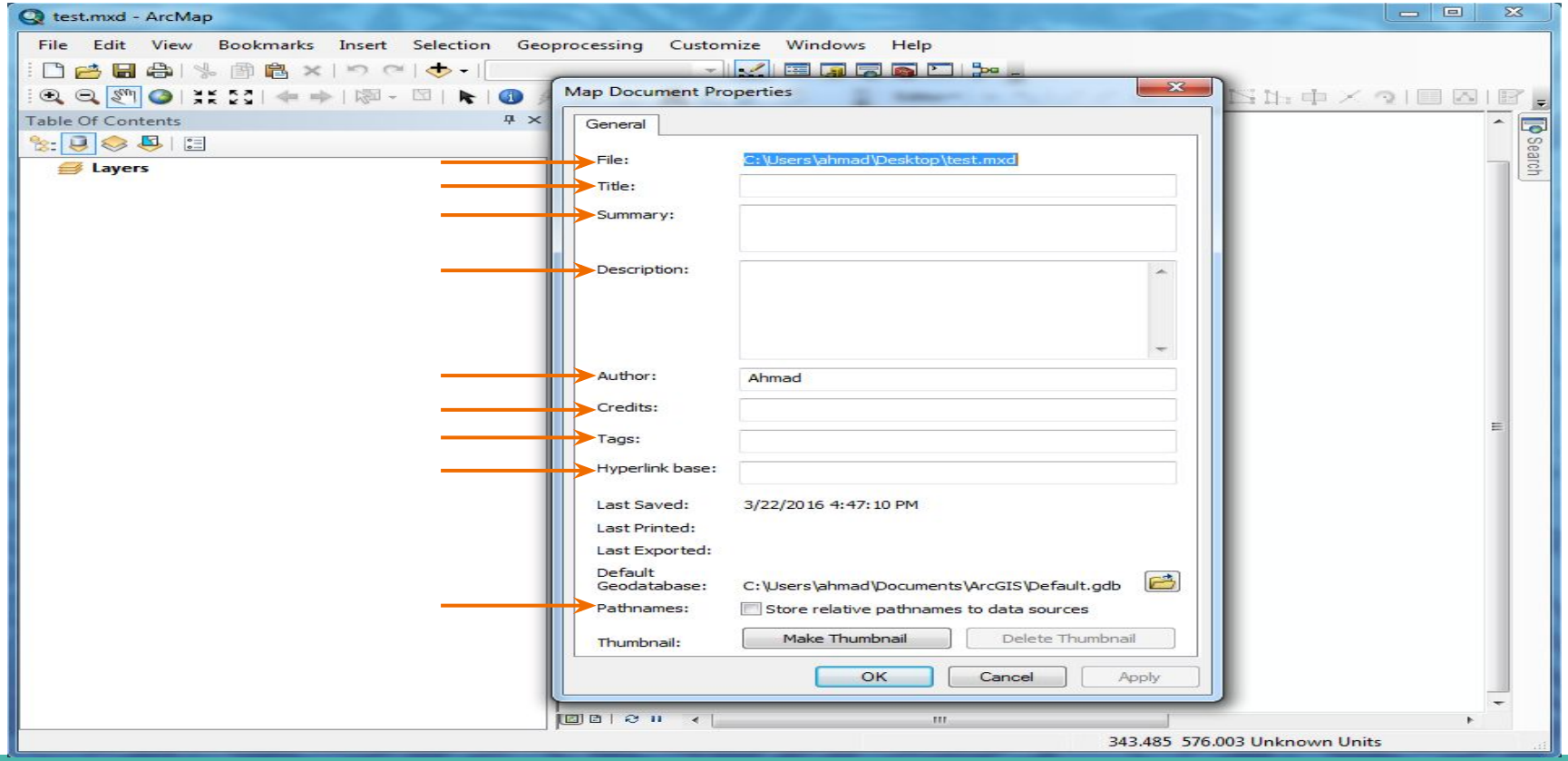
```
print(mD.author)
mD.author = 'Ahmad'
```

- Other properties are read only

```
print(mD.filePath)
mD.filePath = 'C:/temp/test.mxd'
```



Properties


test.mxd - ArcMap
File Edit View Bookmarks Insert Selection Geoprocessing Customize Windows Help
Table Of Contents
Layers
Map Document Properties
General
File:
C:\Users\ahmad\Desktop\test.mxd
Title:
Summary:
Description:
Author:
Ahmad
Credits:
Tags:
Hyperlink base:
Last Saved:
3/22/2016 4:47:10 PM
Last Printed:
Last Exported:
Default Geodatabase:
C:\Users\ahmad\Documents\ArcGIS\Default.gdb
Pathnames:
Store relative pathnames to data sources
Thumbnail:
Make Thumbnail
Delete Thumbnail
OK
Cancel
Apply
Search
343.485 576.003 Unknown Units

Methods

- Some methods do not take parameters

```
mD.save()
```

- Other methods need parameters. Some parameters are required and some are optional.

  In documentation ⟶ methodName(requiredP1,requiredP2.., {optionalP1}, {optionalP2}..)

```
mD.saveACopy('D:/backup/city_3_30_16.mxd')
mD.saveACopy('D:/backup/city_3_30_16.mxd', '8.3')
```

Property Vs. Method in Syntax

objName.propertyName

Vs.

objName.methodName**(...)**

activeDataFrame

- The activeDataFrame is a property of the MapDocument class.
- mD.activeDataFrame → returns a DataFrame object.
- Similarly, mD.author → returns a string.
- The return DataFrame object is based on the DataFrame class.
- We can use the DataFrame class properties and methods.

```
aDF = mD.activeDataFrame
aDF.name = 'Main'
```

❖ **Next lesson, we will learn how to navigate layers in dataframes.**

Layer Class Access

arcpy package → mapping module → Layer class

Create a Layer Object

```
import arcpy

parksShapeLayer = arcpy.mapping.Layer('D:/data/parks.shp')
parksFClassLayer = arcpy.mapping.Layer('D:/data/input.gdb/parks')
parksDotlyrLayer = arcpy.mapping.Layer('D:/data/parks.lyr)
```

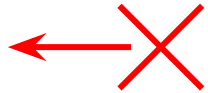
Properties

- Some properties you can read/write

```
print(parksShapeLayer.visible)
parksShapeLayer.visible = False
```

- Other properties are read only

```
print(parksShapeLayer.isFeatureLayer)
parksShapeLayer.isFeatureLayer = False
```



Methods

- Some methods do not take parameters

```
parksShapeLayer.save()
```

- Other methods need parameters. Some parameters are required and some are optional.

  In documentation ⟶ methodName(requiredP1,requiredP2.., {optionalP1}, {optionalP2}..)

```
parksShapeLayer.saveACopy('D:/backup/parks.lyr')
arcpy.CopyFeatures_management(parksShapeLayer,'D:/backup/' + parksShapeLayer.name + '.shp')
```

AddLayer Function

arcpy package → mapping module → AddLayer Function

AddLayer Function

The AddLayer function needs both a Layer and a DataFrame objects to operate. It basically adds a layer to a dataframe:

```
import arcpy

mD = arcpy.mapping.MapDocument('C:/Users/someuser/Desktop/proj.mxd')
dF = mD.activeDataFrame
buildingsLayer = arcpy.mapping.Layer('C:/data/buildings.shp')
roadsLayer = arcpy.mapping.Layer('C:/data/roads.shp')
arcpy.mapping.AddLayer(dF, buildingsLayer, 'TOP')
arcpy.mapping.AddLayer(dF, roadsLayer, 'TOP')
mD.save()
```

ListLayers Function

arcpy package → mapping module → ListLayers Function

ListLayers Function

The ListLayers function returns a list of the layers in an mxd document or in a specific dataframe:

```
import arcpy

mD = arcpy.mapping.MapDocument('C:/Users/someuser/Desktop/proj.mxd')
dF = mD.activeDataFrame
allLayers = arcpy.mapping.ListLayers(mD)
firstDFLayers = arcpy.mapping.ListLayers(mD, None, dF)
```

Analysis Toolbox

There are 4 toolsets inside the analysis toolbox:

1. Extract → here we can find Clip_analysis(...) tool
2. Overlay → here we can find Intersect_analysis(...) tool
3. Proximity → here we can find Buffer_analysis(...) tool
4. Statistics

Let's look into the ESRI's documentation again for details on these tools.

Analysis Toolbox Example 1

```
.....
.....
intersect = '....path..../intersect.shp' # this shape file does not exist yet.
union = '....path..../union.shp' # this shape file also does not exist yet
arcpy.Intersect_analysis(['layer1' , 'layer2'] , intersect)
arcpy.Union_analysis(['layer1' , 'layer2'] , union)
.....
.....
```

Analysis Toolbox Example 2

```
.....
for l in layersList:
        print('Creating the buffer shapefile of ' + l.name + ' ...')
        arcpy.Buffer_analysis(l.dataSource, 'output/' + l.name + 'buf.shp','100 Feet')
        newL = arcpy.mapping.Layer('output/' + l.name + 'buf.shp')
        print('Adding the buffered shapefile of ' + l.name + ' ,named ' + l.name + 'buf.shp to the table of
contents ...')
        arcpy.mapping.AddLayer(df,newL,'BOTTOM')

print('Saving the mxd ...')
m.save()
print('Scripting done')
.....
```

Conclusion

Python can also be used with QGIS.

Also, it can be used in GeoDjango, a web GIS environment.

arcpy package has many other modules, classes, and functions to explore.

Thank you for listening