Chapter 8: Mapping with ArcPy

- New to ArcGIS 10 Python scripting library
- Allows you to:
    - Manage and manipulate map documents (.mxd)
    - Manage and manipulate layer files (.lyr)
        - Both layer packages (.lpk) and layer files (.lyr) are created using ArcGIS Desktop. Layer files store the layer properties, including cartography, and point to the source data. Layer packages encapsulate the data and layer file. Layer packages are ideal for sharing, especially on ArcGIS Online.

    - Manage and manipulate data within map documents and layer files
        - For example, if you have a set of map document files whose data sources have been changed. You could use the Arcpy mapping module in a script that automatically updates the contents of the map document files to point to the new data sources.
        - Batch change symbology in mxd files

    - Automation of printing and exporting map documents
    - Automate map production and creation of pdf map books
    - Publish tasks to ArcGIS Server
- Can be used to accomplish many of the day to day tasks performed by the GIS Analyst

1. Managing map documents and layers (and data within these files)
2. Printing, exporting, map automation and creation of map books
3. Publication to ArcGIS Server

- The functions in an arcpy.mapping script closely mimic the actions performed inside the ArcMap application. Consider the following simple ArcMap workflow:

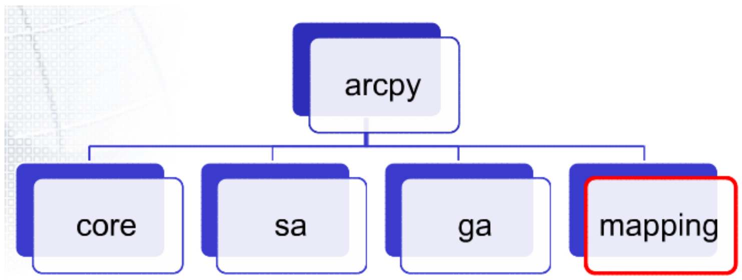
    1. Open the map document located at C:\GIS\TownCenter_2009.mxd.
    2. Find any layout text elements that read GIS Services Division 2009 and change them to read GIS Services Division 2010.
    3. Export the updated map layout to PDF.

An arcpy.mapping script expresses these steps as follows:

This simple find and replace script changes 2009 to 2010, and then exports the layout to a PDF file.

```
mxd = arcpy.mapping.MapDocument(r"C:\GIS\TownCenter_2009.mxd")
for textElement in arcpy.mapping.ListLayoutElements(mxd, "TEXT_ELEMENT"):
  if textElement.text == "GIS Services Division 2009":
    textElement.text = "GIS Services Division 2010"
arcpy.mapping.ExportToPDF(mxd, r"C:\GIS\TownCenterUpdate_2010.pdf")
del mxd
```

- The geoprocessing arcpy.mapping module is intended for use by anyone who needs to automate an ArcMap workflow.
- While it looks like programming and is driven by the powerful Python programming language, you do not need to be a GIS software developer to create arcpy.mapping scripts.
- The intent of arcpy.mapping is to automate tedious things, allowing you to focus on your important creative and analytic work.
- Where the mapping module fits in



8.1 Running Arcpy.Mapping scripts

- ArcPy.Mapping scripts can be run from a variety of environments just like any other geoprocessing script that you've developed with Python for use with the ArcGIS geoprocessor.
- The new Python Window in ArcMap is a great interface for writing and testing small code blocks.
- Once tested, these code blocks are often moved into **stand-alone Python scripts which can be executed from an IDE such as PythonWin, IDLE, or Wing**, but they can also be executed from **custom script tools in ArcToolbox, the command line or as scheduled tasks**.
- Many people choose to attach their scripts to script tools in ArcMap or ArcCatalog to provide a visual interface for the scripts within the familiar ArcGIS Desktop environment.
- Finally, scripts can also be published as **geoprocessing tasks in ArcGIS Server**.

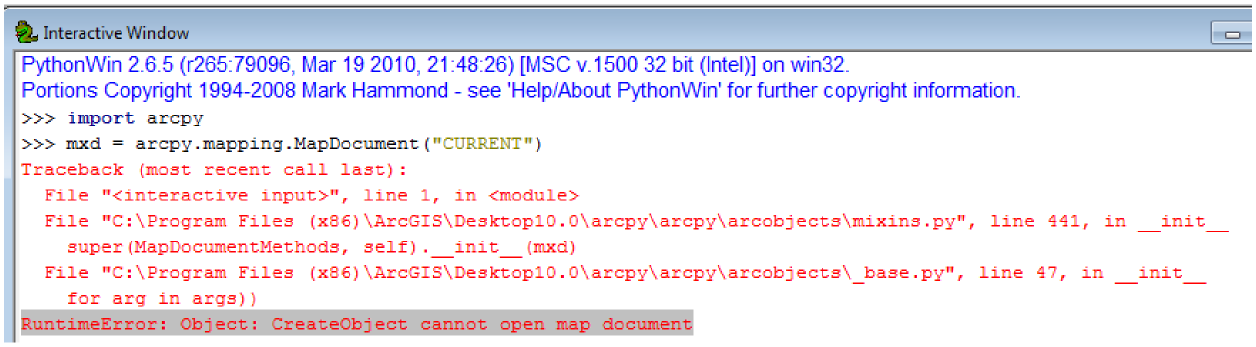
8.1 Working with map documents

- Before you can actually perform any operations on a map document file you need to get a reference to it in your Python script.
- This is done by calling the MapDocument() method on the arcpy.mapping module.
- You can reference either the currently running document or a document at a specific location. To reference the currently active document you simply supply the keyword "CURRENT" as a parameter to the MapDocument() function.
- ArcMap should be running

Import arcpy

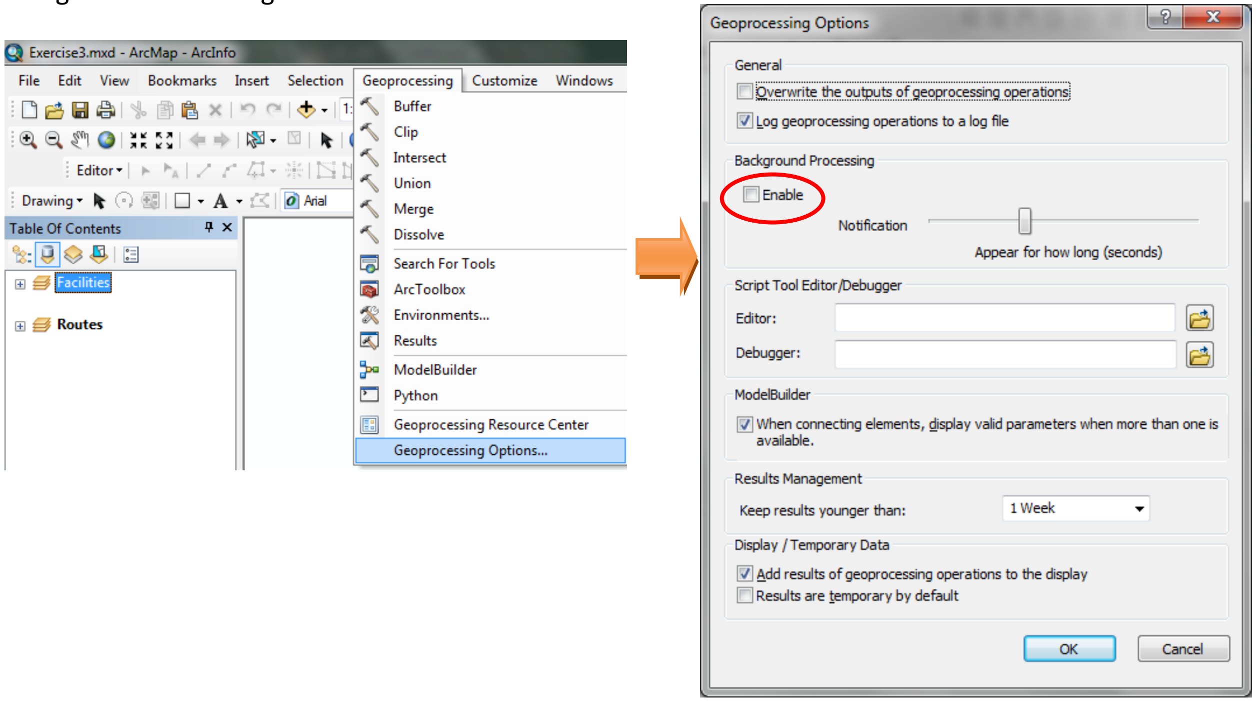
mapDoc = arcpy.mapping.MapDocument("CURRENT")

  - Possible errors: Runtime



```
PythonWin 2.6.5 (r265:79096, Mar 19 2010, 21:48:26) [MSC v.1500 32 bit (Intel)] on win32.
Portions Copyright 1994-2008 Mark Hammond - see 'Help/About PythonWin' for further copyright information.
>>> import arcpy
>>> mxd = arcpy.mapping.MapDocument("CURRENT")
Traceback (most recent call last):
  File "<interactive input>", line 1, in <module>
  File "C:\Program Files (x86)\ArcGIS\Desktop10.0\arcpy\arcpy\arcobjects\mixins.py", line 441, in __init__
    super(MapDocumentMethods, self).__init__(mxd)
  File "C:\Program Files (x86)\ArcGIS\Desktop10.0\arcpy\arcpy\arcobjects\_base.py", line 47, in __init__
    for arg in args))
RuntimeError: Object: CreateObject cannot open map document
```

This error arises if background processing in ArcGIS is enabled. To correct this error you will need to disable background processes in ArcGIS. (Geoprocessing→Geoprocessing Options). Make sure Background Processing is UNCHECKED.



Geoprocessing Options

General
- [ ] Overwrite the outputs of geoprocessing operations
- [x] Log geoprocessing operations to a log file

Background Processing
- [ ] Enable

Notification
Appear for how long (seconds)

Script Tool Editor/Debugger
Editor:
Debugger:

ModelBuilder
- [x] When connecting elements, display valid parameters when more than one is available.

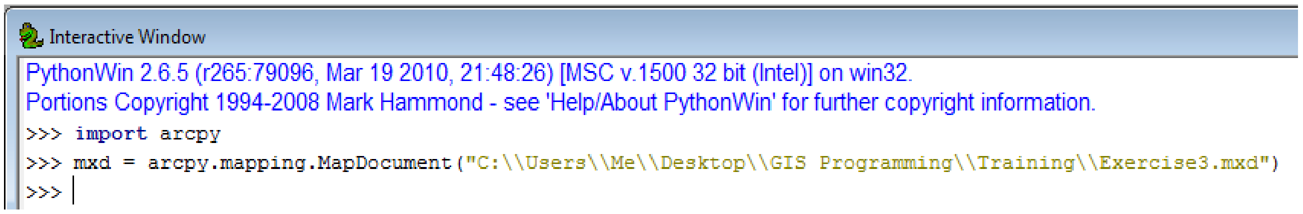
Results Management
Keep results younger than: 1 Week

Display / Temporary Data
- [x] Add results of geoprocessing operations to the display
- [ ] Results are temporary by default

OK Cancel

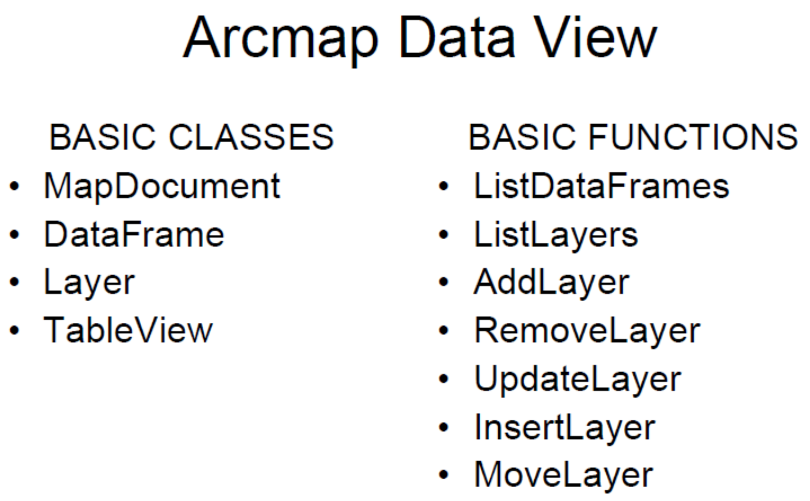
See http://help.arcgis.com/en/arcgisdesktop/10.0/help/index.html#//00210000003q000000.htm for more information.

- To reference a map document on disk you simply supply the path to the map document as well as the map document name as a parameter to MapDocument().



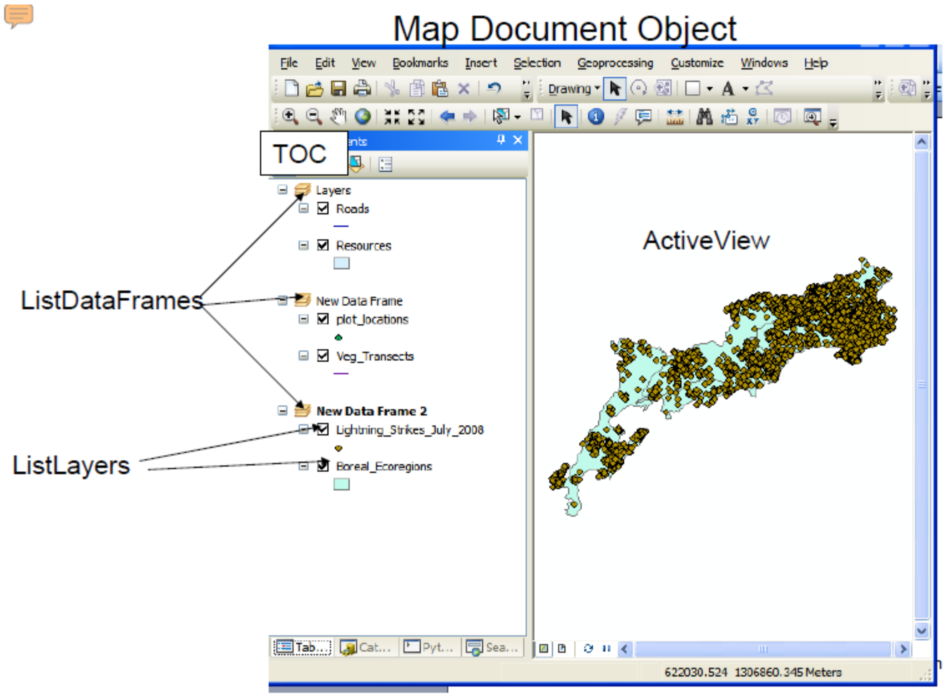
```
PythonWin 2.6.5 (r265:79096, Mar 19 2010, 21:48:26) [MSC v.1500 32 bit (Intel)] on win32.
Portions Copyright 1994-2008 Mark Hammond - see 'Help/About PythonWin' for further copyright information.
>>> import arcpy
>>> mxd = arcpy.mapping.MapDocument("C:\\Users\\Me\\Desktop\\GIS Programming\\Training\\Exercise3.mxd")
>>> 
```

8.2 Mapping basics



Arcmap Data View

BASIC CLASSES	BASIC FUNCTIONS
• MapDocument	• ListDataFrames
• DataFrame	• ListLayers
• Layer	• AddLayer
• TableView	• RemoveLayer
	• UpdateLayer
	• InsertLayer
	• MoveLayer

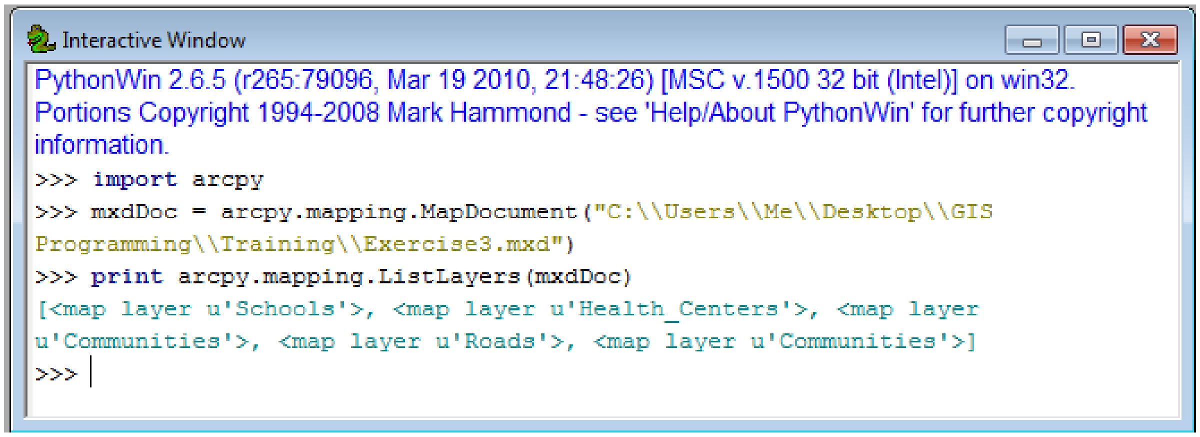


8.3 List functions

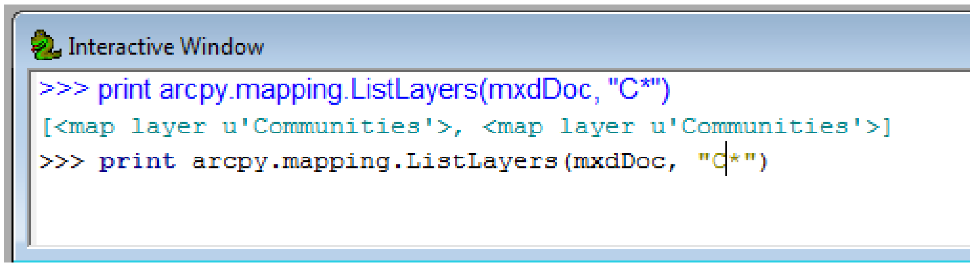
- **All list functions are returned in a Python list**
- Usually just the first step of a multi-step approach. For example, you may need to iterate between a list of layers with broken links and fix each one.
- The ListLayers() function
    - Returns all layers within an mxd, dataframe within an mxd or lyr file
- Can iterate list with a for loop
- Syntax

    ListLayers (map_document_or_layer, {wildcard}, {data_frame})

- Wildcards are used on the name property and are not case sensitive.
- A wildcard string of "so*" will return a layer with the name Soils.
- Wildcards can be skipped in the scripting syntax simply by passing an empty string (""), an asterisk (*), or entering wildcard=None, or nothing at all if it is the last optional parameter in the syntax.

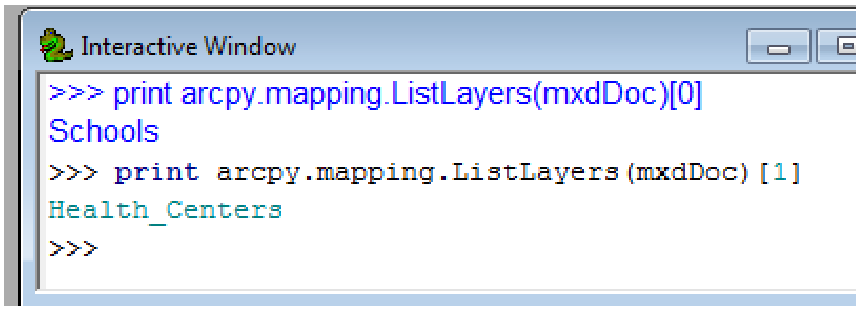


```
PythonWin 2.6.5 (r265:79096, Mar 19 2010, 21:48:26) [MSC v.1500 32 bit (Intel)] on win32.
Portions Copyright 1994-2008 Mark Hammond - see 'Help/About PythonWin' for further copyright
information.
>>> import arcpy
>>> mxdDoc = arcpy.mapping.MapDocument("C:\\Users\\Me\\Desktop\\GIS
Programming\\Training\\Exercise3.mxd")
>>> print arcpy.mapping.ListLayers(mxdDoc)
[<map layer u'Schools'>, <map layer u'Health_Centers'>, <map layer
u'Communities'>, <map layer u'Roads'>, <map layer u'Communities'>]
>>>
```



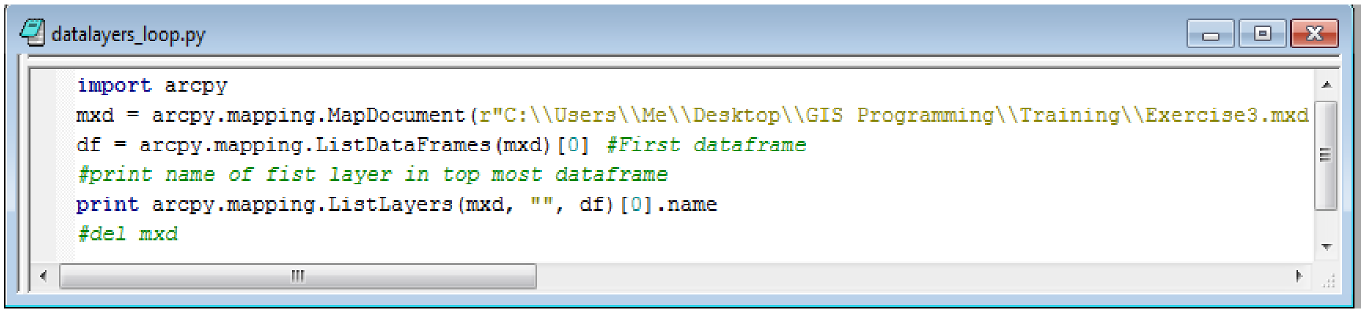
```
>>> print arcpy.mapping.ListLayers(mxdDoc, "C*")
[<map layer u'Communities'>, <map layer u'Communities'>]
>>> print arcpy.mapping.ListLayers(mxdDoc, "C*")
```

- Results are returned in a list

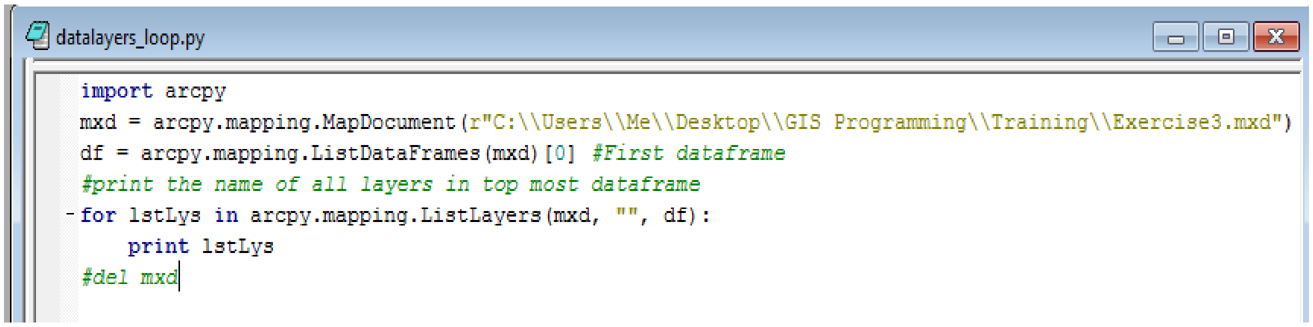


```
>>> print arcpy.mapping.ListLayers(mxdDoc)[0]
Schools
>>> print arcpy.mapping.ListLayers(mxdDoc)[1]
Health_Centers
>>>
```

- {DataFrame}
  - **A variable that references a DataFrame object.**
  - ListLayers always returns a Python list object even if only one layer is returned. In order to return a Layer object, an index value must be used on the list (e.g., lyr = arcpy.mapping.ListLayers(mxd)[0]). An Index value must also be used with the DataFrame


datalayers_loop.py

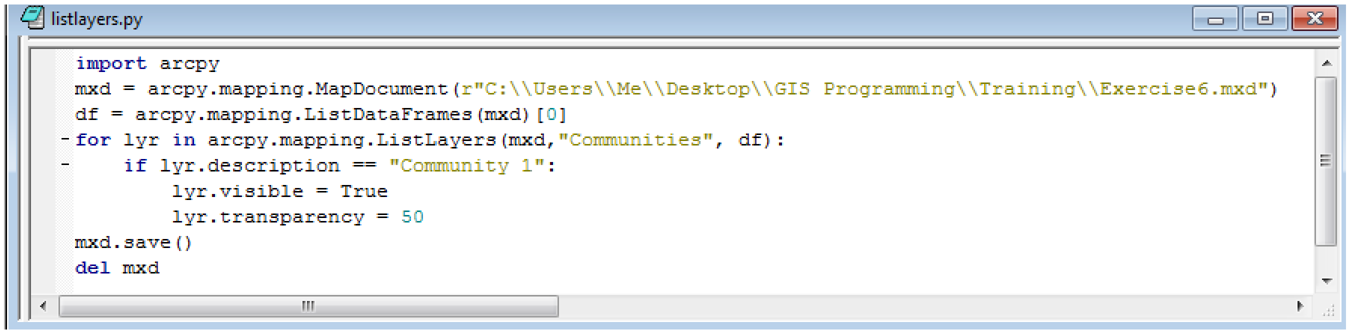
```
import arcpy
mxd = arcpy.mapping.MapDocument(r"C:\\Users\\Me\\Desktop\\GIS Programming\\Training\\Exercise3.mxd
df = arcpy.mapping.ListDataFrames(mxd)[0] #First dataframe
#print name of fist layer in top most dataframe
print arcpy.mapping.ListLayers(mxd, "", df)[0].name
#del mxd
```


datalayers_loop.py

```
import arcpy
mxd = arcpy.mapping.MapDocument(r"C:\\Users\\Me\\Desktop\\GIS Programming\\Training\\Exercise3.mxd")
df = arcpy.mapping.ListDataFrames(mxd)[0] #First dataframe
#print the name of all layers in top most dataframe
for lstLys in arcpy.mapping.ListLayers(mxd, "", df):
    print lstLys
#del mxd
```

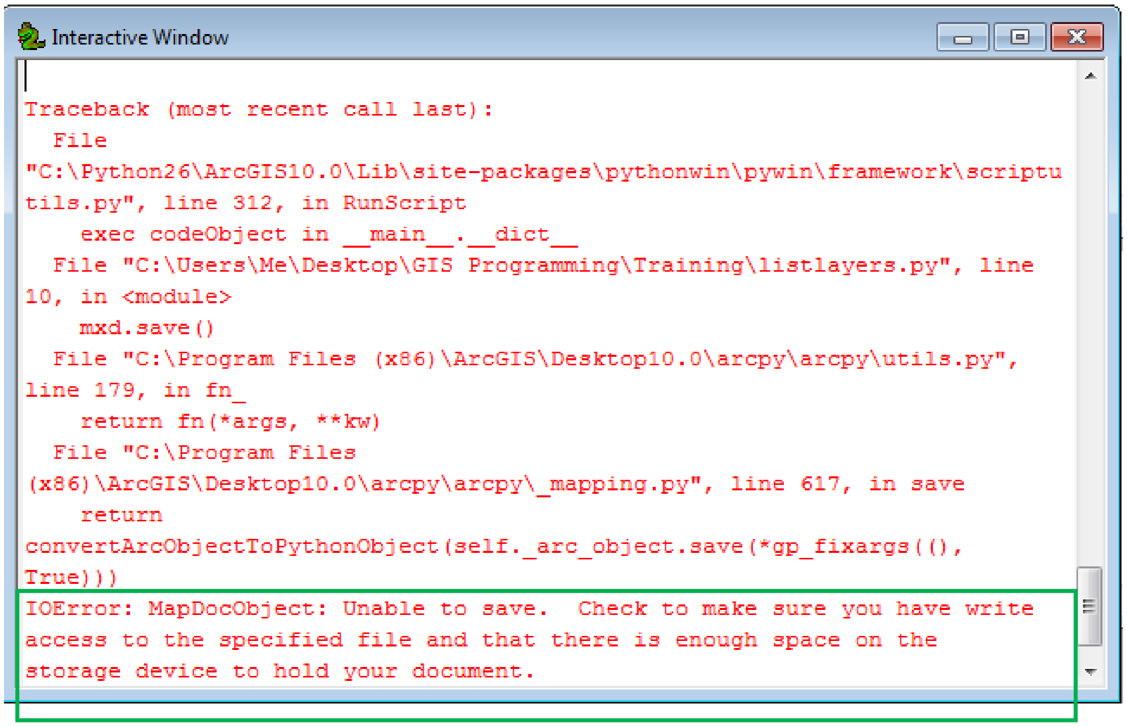
Example 1:listlayers.py

The following script will find a layer called Communities in a data frame named Facilities, turn the layer on (to be visible) and set the transparency to 50%. An issue is that there happens to be two layers with the same name in the same data frame so the description property is used to further isolate the layer of interest. Ideally, all layers would have a unique name but that isn't always the case so other properties have to be used to isolate the layer of interest. In this example, the description is used. Make sure that the Communities layer in the Facilities dataframe is unchecked in the Table of Contents in ArcMap to test that the script really worked.


listlayers.py

```
import arcpy
mxd = arcpy.mapping.MapDocument(r"C:\\Users\\Me\\Desktop\\GIS Programming\\Training\\Exercise6.mxd")
df = arcpy.mapping.ListDataFrames(mxd)[0]
for lyr in arcpy.mapping.ListLayers(mxd,"Communities", df):
    if lyr.description == "Community 1":
        lyr.visible = True
        lyr.transparency = 50
mxd.save()
del mxd
```

Make sure that ArcGIS is close before the script is run in PythonWin. Failure to do so will result in the following error:

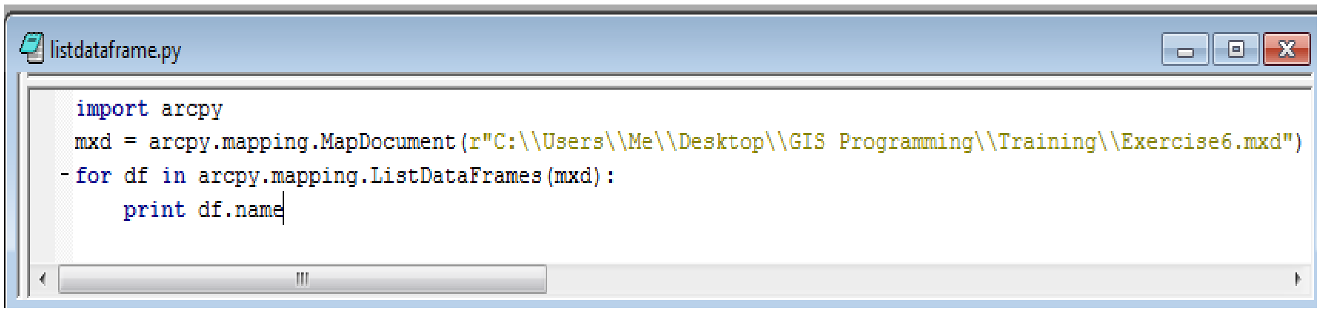


```
Traceback (most recent call last):
  File
"C:\Python26\ArcGIS10.0\Lib\site-packages\pythonwin\pywin\framework\scriptu
tils.py", line 312, in RunScript
    exec codeObject in __main__.__dict__
  File "C:\Users\Me\Desktop\GIS Programming\Training\listlayers.py", line
10, in <module>
    mxd.save()
  File "C:\Program Files (x86)\ArcGIS\Desktop10.0\arcpy\arcpy\utils.py",
line 179, in fn_
    return fn(*args, **kw)
  File "C:\Program Files
(x86)\ArcGIS\Desktop10.0\arcpy\arcpy\_mapping.py", line 617, in save
    return
convertArcObjectToPythonObject(self._arc_object.save(*gp_fixargs((),
True)))
IOError: MapDocObject: Unable to save.  Check to make sure you have write
access to the specified file and that there is enough space on the
storage device to hold your document.
```

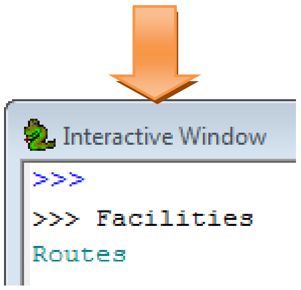
- ListDataFrames() returns a Python list of data frames in a map document file.
- An integer index value can be used to access an individual data frame or you can test for a specific data frame name before applying further processing to the data within the data frame.
- You can also use a wildcard to filter the data frames that are returned.
- The index value isolates an individual frame in the instance that there are multiple frames with the same name.

Example: listdataframe.py

Prints the name of each dataframe in the map document referenced



```
import arcpy
mxd = arcpy.mapping.MapDocument(r"C:\\Users\\Me\\Desktop\\GIS Programming\\Training\\Exercise6.mxd")
for df in arcpy.mapping.ListDataFrames(mxd):
    print df.name
```



```
>>>
>>> Facilities
Routes
```

8.4 Exercise 7