

UC



# Python - Tips and Tricks for working with arcpy mapping

jack Horton

## Learn how to use the `arcpy.mapping` module to:

- Alter the contents of map documents such as title, layers, and layouts
- Repair your MXD files when your data is moved
- Work live on your current map document or work on MXD files
- Process multiple MXD files

... and see that ArcGIS Pro has very similar functionality in its `arcpy.mp` module

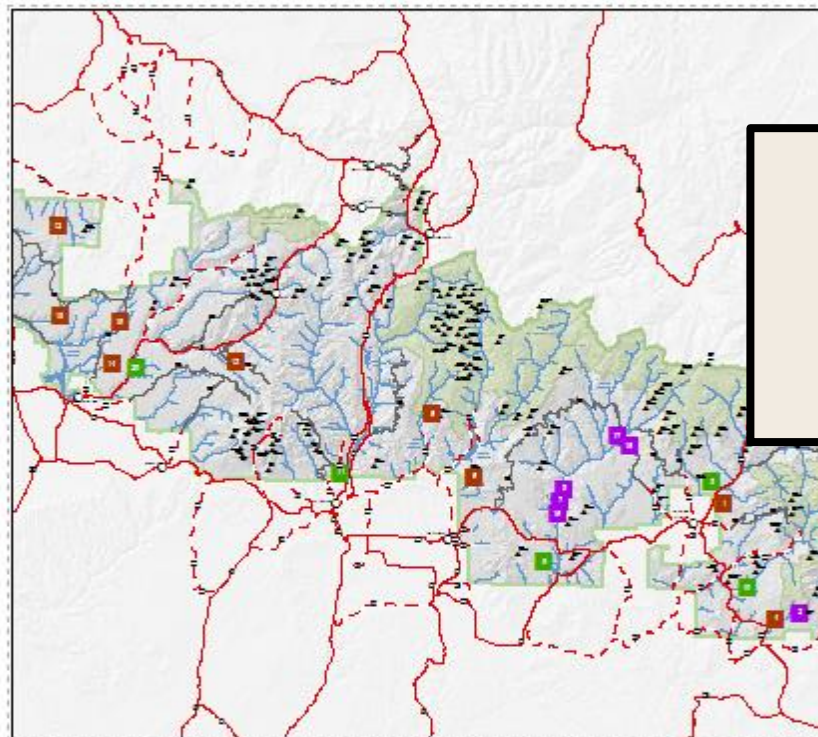




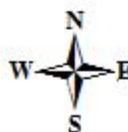
## San Juan

- ☒ LakesAnno
- ☒ WildernessAnno
- ☒ CitiesAnno
- ☒ Spot\_HeightsAnno
- ☒ RoadsAnno
- ☒ Invasive Plants
- ☒ Cities
- ☐ Mountain peaks
- ☒ Roads
  - ROUTE\_TYPE
  - Primary road
  - Secondary road
  - Light duty road
- ☒ Streams
- ☒ Lakes
- ☐ index
- ☒ Wilderness

## Invasive plant treatment sites in San Juan National Forest



0 5 10 20 30 40 50 Miles



## Python

```
>>> import arcpy.mapping as am
>>>
```

**We import arcpy.mapping, so it is now enabled, and we have a variable named am to quickly reference it.**

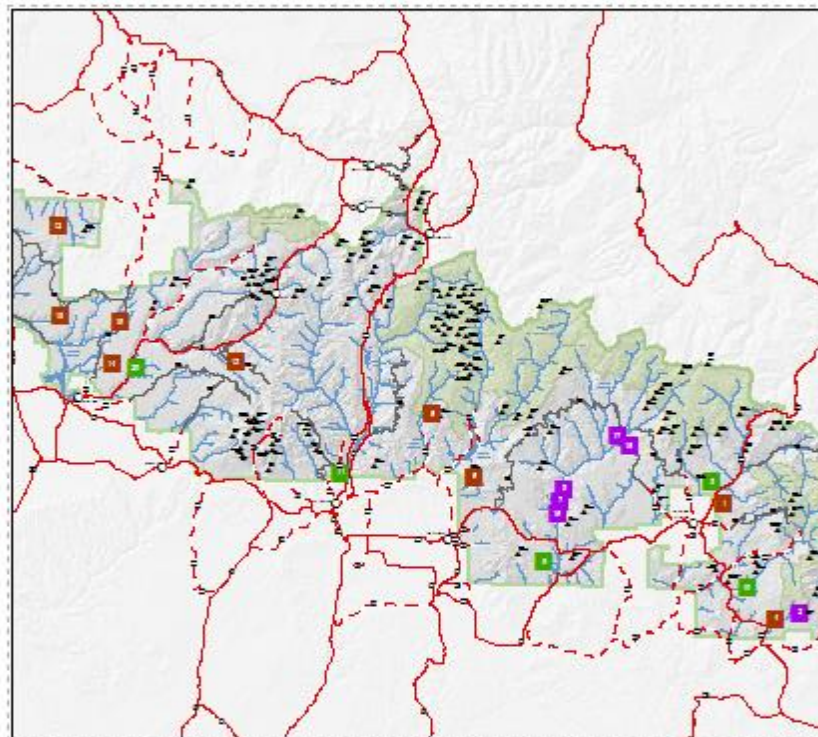
## Table Of Contents



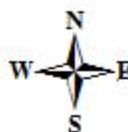
## San Juan

- ☒ LakesAnno
- ☒ WildernessAnno
- ☒ CitiesAnno
- ☒ Spot\_HeightsAnno
- ☒ RoadsAnno
- ☒ Invasive Plants
- ☒ Cities
- ☐ Mountain peaks
- ☒ Roads
  - ROUTE\_TYPE
  - Primary road
  - Secondary road
  - Light duty road
- ☒ Streams
- ☒ Lakes
- ☐ index
- ☒ Wilderness

## Invasive plant treatment sites in San Juan National Forest



0 5 10 20 30 40 50 Miles



## Python

```
>>> import arcpy.mapping as am
>>> am.
```

- ◆ AddLayer
- ◆ AddLayerToGroup
- ◆ AddTableView
- ◆ AnalyzeForMSD
- ◆ AnalyzeForSD
- ◆ ArgAdaptor
- ◆ ConvertToMSD

am.

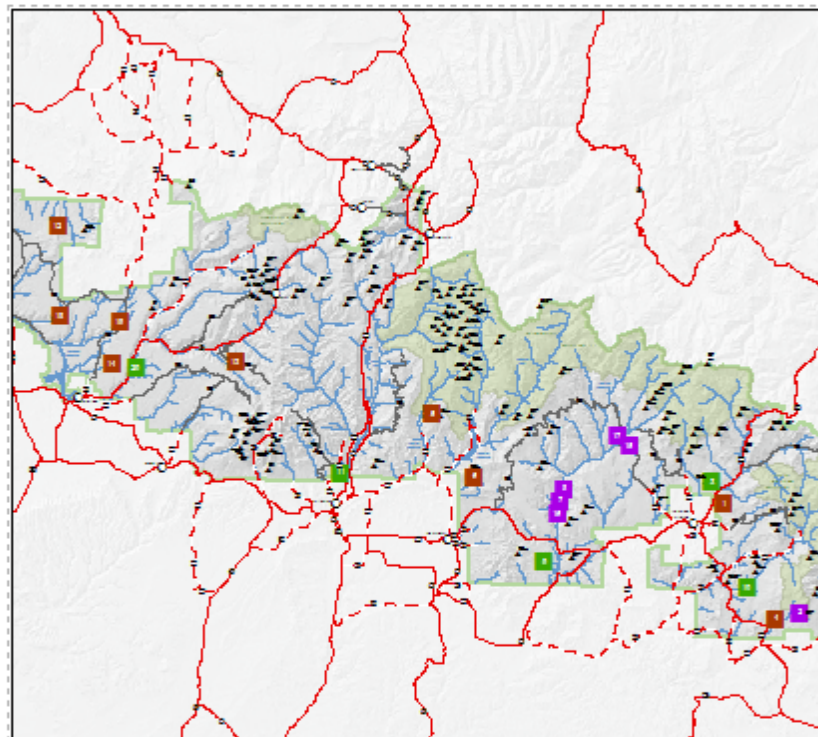
**Our am variable can  
get to lots of things...**



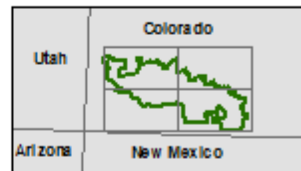
## Table Of Contents

- San Juan**
- ☒ LakesAnno
  - ☒ WildernessAnno
  - ☒ CitiesAnno
  - ☒ Spot\_HeightsAnno
  - ☒ RoadsAnno
  - ☒ Invasive Plants
  - ☒ Cities
  - ☐ Mountain peaks
  - ☒ Roads
    - ROUTE\_TYPE
    - Primary road
    - Secondary road
    - Light duty road
  - ☒ Streams
  - ☒ Lakes
  - ☐ index
  - ☒ Wilderness

## Invasive plant treatment sites in San Juan National Forest



0 5 10 20 30 40 50 Miles



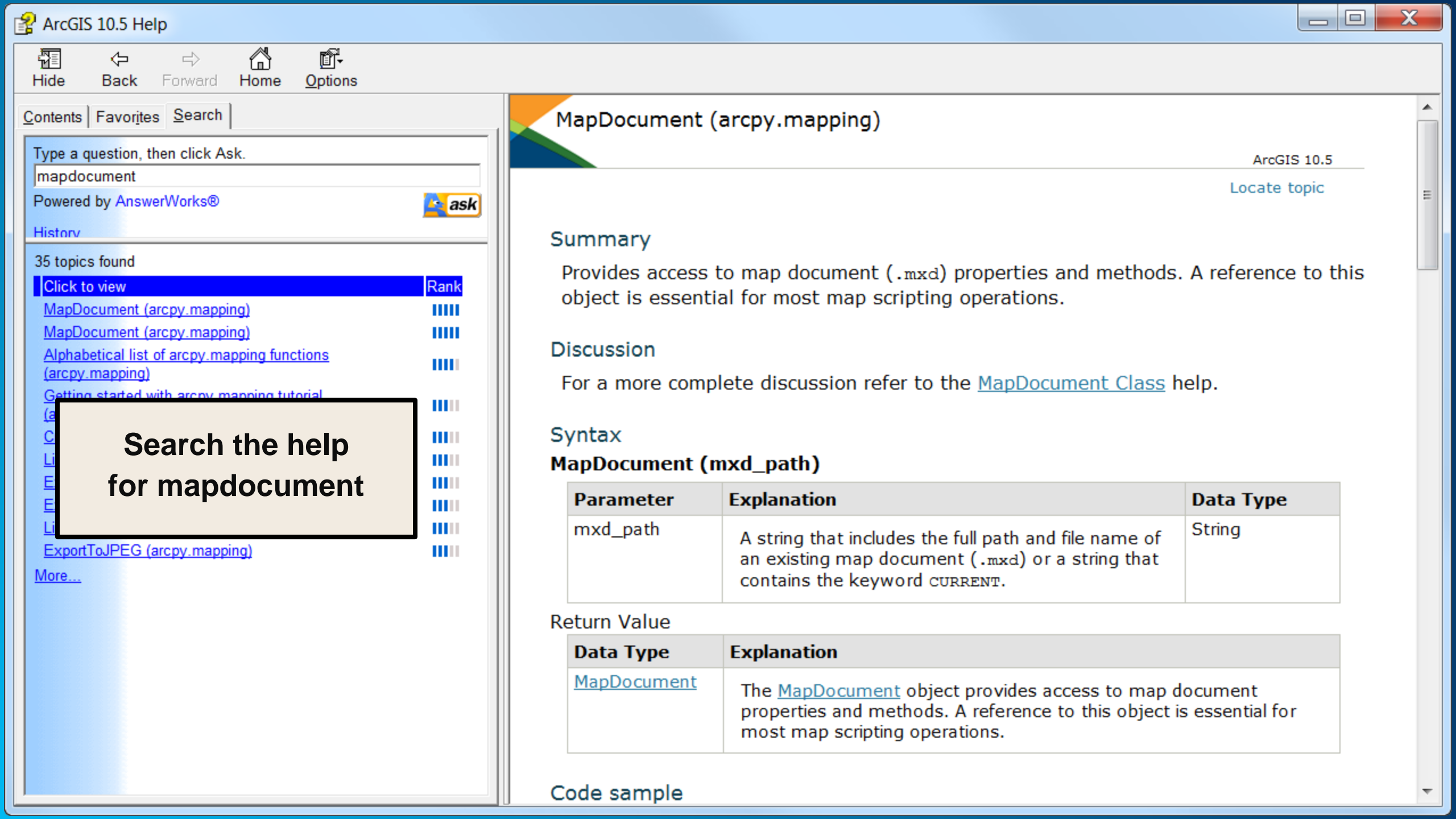
## Python

```
>>> import arcpy.mapping as am  
>>> am.
```

- ◆ ListPrinterNames
- ◆ ListStyleItems
- ◆ ListTableViews
- ◆ MapDocument
- ◆ MoveLayer
- ◆ PDFDocumentCreate
- ◆ PDFDocumentOpen

... including a  
map document

am.



Type a question, then click Ask.

mapdocument

Powered by AnswerWorks®

ask

History

35 topics found

Click to view	Rank
<a href="#">MapDocument (arcpy.mapping)</a>	
<a href="#">MapDocument (arcpy.mapping)</a>	
<a href="#">Alphabetical list of arcpy.mapping functions (arcpy.mapping)</a>	
<a href="#">Getting started with arcpy.mapping tutorial (a</a>	
<a href="#">C</a>	
<a href="#">Li</a>	
<a href="#">E</a>	
<a href="#">E</a>	
<a href="#">Li</a>	
<a href="#">ExportToJPEG (arcpy.mapping)</a>	

[More...](#)

## MapDocument (arcpy.mapping)

ArcGIS 10.5

[Locate topic](#)

### Summary

Provides access to map document (.mxd) properties and methods. A reference to this object is essential for most map scripting operations.

### Discussion

For a more complete discussion refer to the [MapDocument Class](#) help.

### Syntax

#### MapDocument (mxd\_path)

Parameter	Explanation	Data Type
mxd_path	A string that includes the full path and file name of an existing map document (.mxd) or a string that contains the keyword CURRENT.	String

### Return Value

Data Type	Explanation
<a href="#">MapDocument</a>	The <a href="#">MapDocument</a> object provides access to map document properties and methods. A reference to this object is essential for most map scripting operations.

### Code sample

Type a question, then click Ask.

mapdocument

Powered by AnswerWorks®



History

35 topics found

Click to view	Rank
<a href="#">MapDocument (arcpy.mapping)</a>	
<a href="#">MapDocument (arcpy.mapping)</a>	
<a href="#">Alphabetical list of arcpy.mapping functions (arcpy.mapping)</a>	
<a href="#">Getting started with arcpy.mapping tutorial (arcpy.mapping)</a>	
<a href="#">CreateMapSDDraft (arcpy.mapping)</a>	
<a href="#">ListBookmarks (arcpy.mapping)</a>	
<a href="#">ExportToEPS (arcpy.mapping)</a>	
<a href="#">ExportToBMP (arcpy.mapping)</a>	
<a href="#">ListLayoutElements (arcpy.mapping)</a>	
<a href="#">ExportToJPEG (arcpy.mapping)</a>	

[More...](#)

## MapDocument (arcpy.mapping)

### Summary

Provides access to map document (.mxd) object is essential for most map scripting operations.

### Discussion

For a more complete discussion refer to the [MapDocument Class](#) help.

**The MapDocument method takes an mxd\_path and it returns a MapDocument Object**

### Syntax

#### MapDocument (mxd\_path)

Parameter	Explanation	Data Type
mxd_path	A string that includes the full path and file name of an existing map document (.mxd) or a string that contains the keyword CURRENT.	String

### Return Value

Data Type	Explanation
<a href="#">MapDocument</a>	The <a href="#">MapDocument</a> object provides access to map document properties and methods. A reference to this object is essential for most map scripting operations.

### Code sample

# Syntax

## MapDocument (mxd\_path)

Parameter	Explanation	Data Type
mxd_path	A string that includes the full path and file name of an existing map document (.mxd) or a string that contains the keyword CURRENT.	String

# Return Value

Data Type	Explanation
<a href="#">MapDocument</a>	The <a href="#">MapDocument</a> object provides properties and methods. A reference to the current map document is returned by most map scripting operations.

You can provide the pathname to an .mxd file, or you can use “CURRENT” to work live with the current map document



ArcGIS 10.5 Help

Hide

Back

Forward

Home

Options

Contents

Favorites

Search

Type a question, then click Ask.

mapdocument

Powered by AnswerWorks®

ask

History

35 topics found

Click to view

Rank

MapDocument (arcpy.mapping)

MapDocument (arcpy.mapping)

Alphabetical list of arcpy.mapping functions (arcpy.mapping)

Getting started (arcpy.mapping)

CreateMapSeries (arcpy.mapping)

ListBookmarks (arcpy.mapping)

ExportToEPS (arcpy.mapping)

ExportToBMP (arcpy.mapping)

ListLayoutElements (arcpy.mapping)

ExportToJPEG (arcpy.mapping)

More...

application. Note: Python strings cannot end with a backslash, even when the string is preceded by an r. You must use a double backslash. This becomes important when appending dynamic file names to a folder path.

```
import arcpy  
mxd = arcpy.mapping.MapDocument(r"C:\Project\Project.mxd")  
for df in arcpy.mapping.ListDataFrames(mxd):  
    mxd.activeView = df.name  
    mxd.title = df.name  
    mxd.saveACopy(r"C:\Project\Output\\" + df.name + ".mxd")  
del mxd
```

MapDocument example 2

The following script demonstrates how the CURRENT keyword can be used within the Python window. This sample will update the first data frame's name and refresh the table of contents so the change can be seen in the application. Paste the following code into the Python window within a new ArcMap document.

```
mxd = arcpy.mapping.MapDocument("CURRENT")  
arcpy.mapping.ListDataFrames(mxd)[0].name = "New Data Frame Name"  
arcpy.RefreshTOC()  
del mxd
```

The help shows examples of each technique

When pasted into the interactive window it will appear as follows. The three dots to the left of the code block indicate that the lines are a single block of code that will be

File Edit View Bookmarks Insert Selection Geoprocessing Customize Windows Help

1:694,405

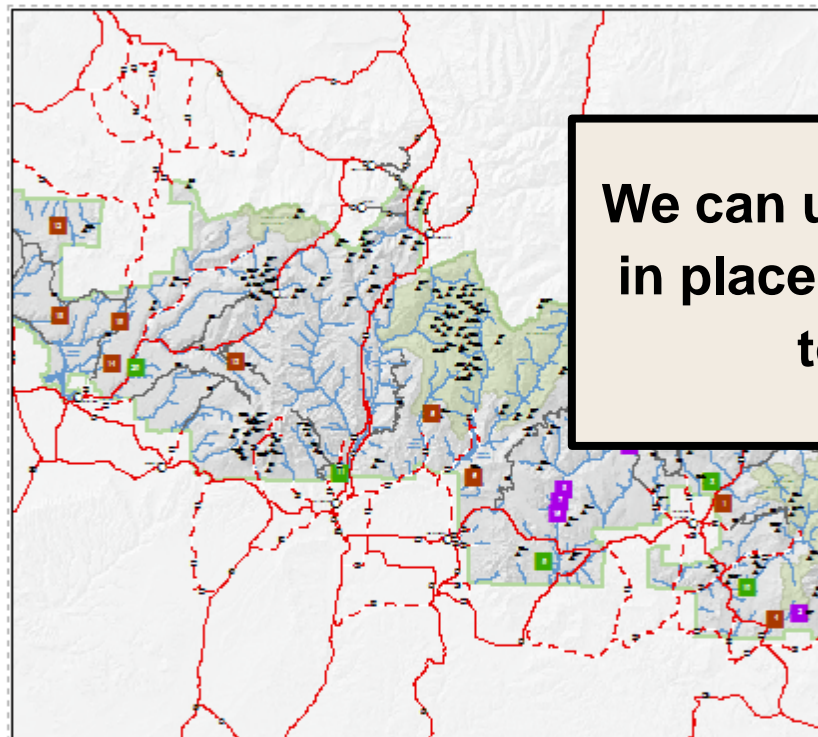
Table Of Contents

↑ ×

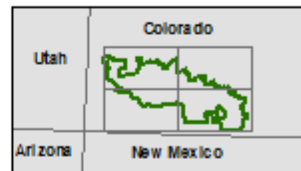
**San Juan**

- ☒ LakesAnno
- ☒ WildernessAnno
- ☒ CitiesAnno
- ☒ Spot\_HeightsAnno
- ☒ RoadsAnno
- ☒ Invasive Plants
- ☒ Cities
- ☐
- ☒ Mountain peaks
- ☒ Roads
  - ROUTE\_TYPE
  - Primary road
  - Secondary road
  - Light duty road
- ☒ Streams
- 
- ☒ Lakes
- 
- ☐ index
- 
- ☒ Wilderness

## Invasive plant treatment sites in San Juan National Forest



0 5 10 20 30 40 50 Miles



Python

↑ ×

```
>>> import arcpy.mapping as am
>>> mxd = am.MapDocument("CURRENT")
>>>
```

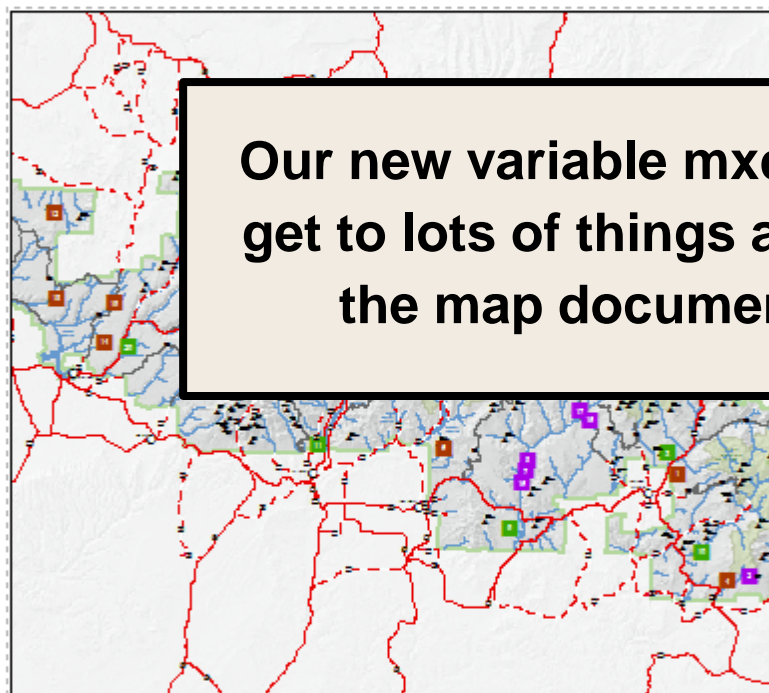
**We can use our am variable  
in place of arcpy.mapping  
to save time**

## Table Of Contents

## San Juan

- ☒ LakesAnno
- ☒ WildernessAnno
- ☒ CitiesAnno
- ☒ Spot\_HeightsAnno
- ☒ RoadsAnno
- ☒ Invasive Plants
- ☒ Cities
- ☐ Mountain peaks
- ☒ Roads
  - ROUTE\_TYPE
  - Primary road
  - Secondary road
  - Light duty road
- ☒ Streams
- ☒ Lakes
- ☐ index
- ☒ Wilderness

## Invasive plant treatment sites in San Juan National Forest



**Our new variable mxd can  
get to lots of things about  
the map document**

## Python

```
>>> import arcpy.mapping as am  
>>> mxd = am.MapDocument("CURRENT")  
>>> mxd.
```

- ◆ activeDataFrame
- ◆ activeView
- ◆ author
- ◆ credits
- ◆ dataDrivenPages
- ◆ dateExported
- ◆ datePrinted

mxd.

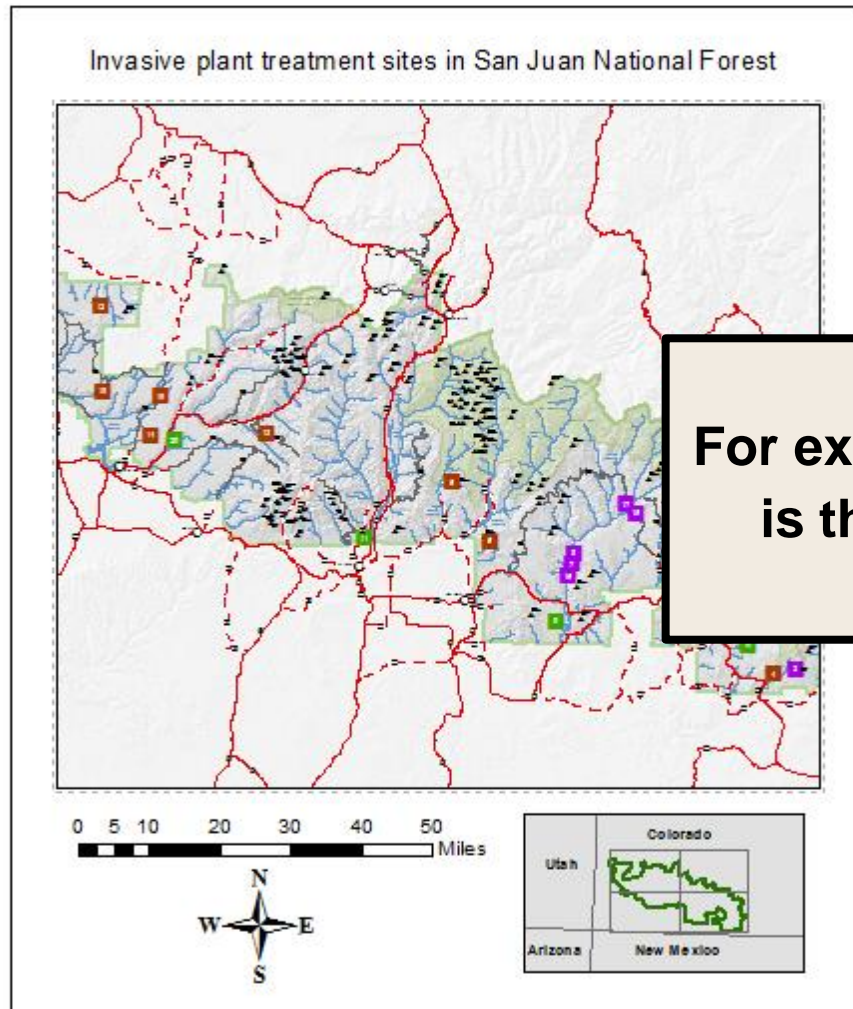
Provides access to map document ( .mxd ) properties and methods. A reference to this object is essential for most map scripting operations.





## San Juan

- ☒ LakesAnno
- ☒ WildernessAnno
- ☒ CitiesAnno
- ☒ Spot\_HeightsAnno
- ☒ RoadsAnno
- ☒ Invasive Plants
- ☒ Cities
- ☐ Mountain peaks
- ☒ Roads
  - ROUTE\_TYPE
  - ☒ Primary road
  - ☒ Secondary road
  - ☒ Light duty road
- ☒ Streams
- ☒ Lakes
- ☐ index
- ☒ Wilderness



```
>>> import arcpy.mapping as am  
>>> mxd = am.MapDocument("CURRENT")  
>>> mxd.title  
u'Invasive plant treatment sites in San  
Juan National Forest'  
>>>
```

**For example, the title property  
is the map document title**

File Edit View Bookmarks Insert Selection Geoprocessing Customize Windows Help

- New... Ctrl+N
- Open... Ctrl+O
- Save Ctrl+S
- Save As...
- Save A Copy...
- Share As
- Add Data
- Sign In...
- ArcGIS Online...
- Page and Print Setup...
- Print Preview...
- Print...
- Export Map...
- Analyze Map...
- Map Document Properties...**

1 D:\jhort...\San\_Juan06.mxd

2 D:\jhort...\San\_Juan05.

3 D:\jhort...\San Diego.n

4 ...\\FeatureAdjustmentF

5 ...\\FeatureAdjustmentF

6 ...\\FeatureAdjustment.

7 ...\\Parcel Fabric Histor

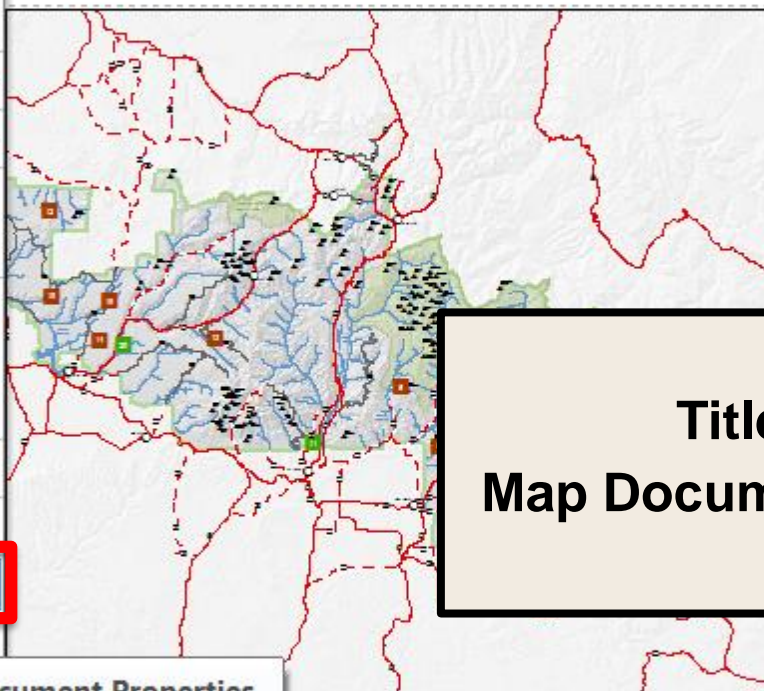
8 ...\\Parcel Fabric Histor

9 D:\jhorton\...\map01.n

**Map Document Properties**

Display or edit the properties of this map document, such as description, author, credits, and specify whether disk-based data it uses will be referenced by relative pathnames.

Invasive plant treatment sites in San Juan National Forest



Title is a  
Map Document Property

Python

```
>>> import arcpy.mapping as am
>>> mxd = am.MapDocument("CURRENT")
>>> mxd.title
u'Invasive plant treatment sites in San
Juan National Forest'
>>>
```



File: C:\Users\jack361\Documents\ArcGIS\Default.gdb\San\_Juan06.mxd

Title: Invasive plant treatment sites in San Juan National Fo

Summary:

Description: Invasive plant treatment sites of San Juan National Forest, CO.

Author:

Credits:

Tags:

Hyperlink base:

Last Saved: 7/2/2017 1:27:13 PM

Last Printed:

Last Exported: 4/16/2010 10:28:11 AM

Default Geodatabase: C:\Users\jack361\Documents\ArcGIS\Default.gdb

Pathnames: ☒ Store relative pathnames to data sources

Thumbnail: Make Thumbnail Delete Thumbnail

OK

Cancel

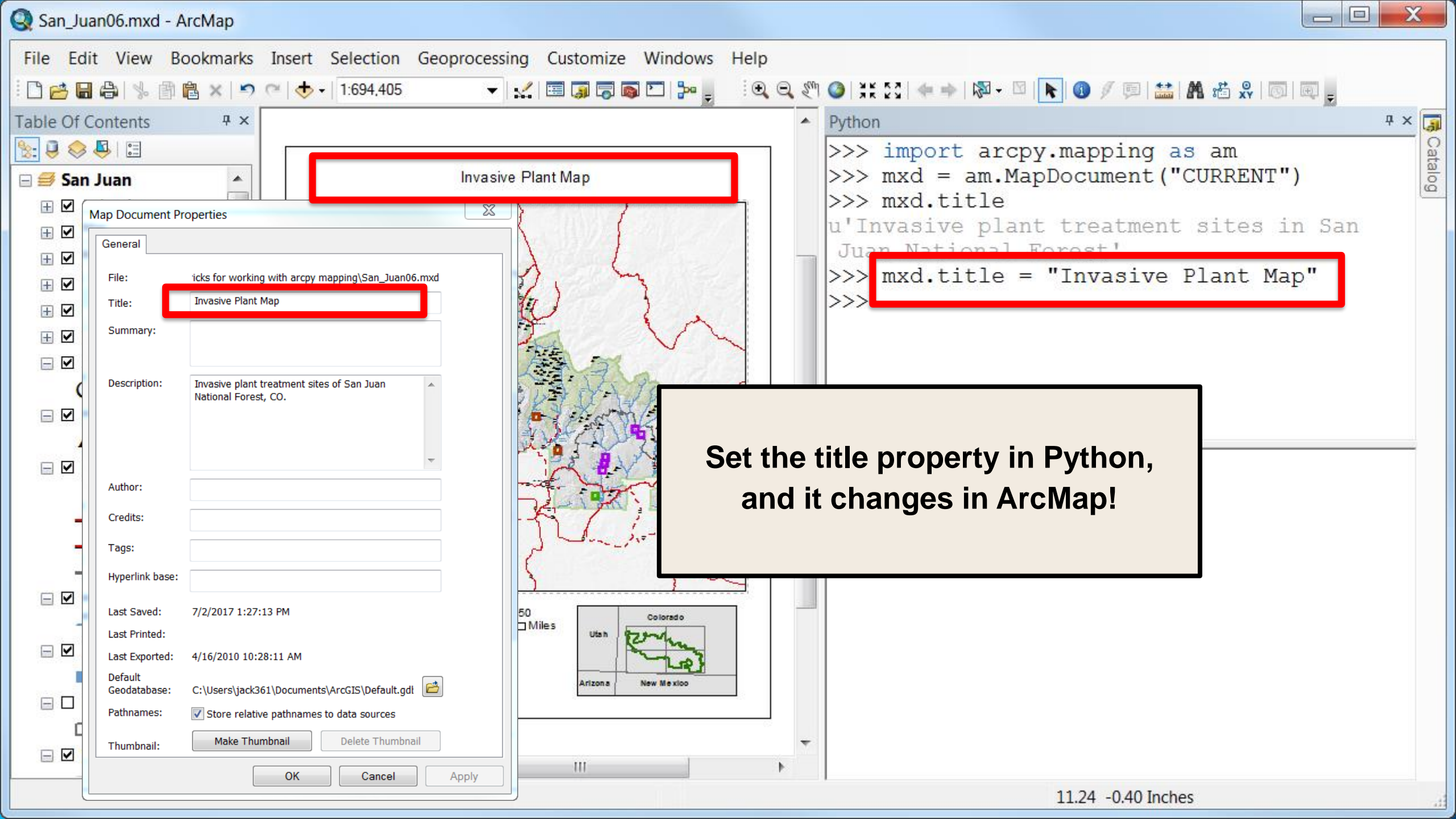
Apply

Invasive plant treatment sites in San Juan National Forest

```
>>> import arcpy.mapping as am
>>> mxd = am.MapDocument("CURRENT")
>>> mxd.title
u'Invasive plant treatment sites in San
Juan National Forest'
```

**It is settable in the Map Document Properties Dialog, and it is used in the Title map element**





Invasive Plant Map

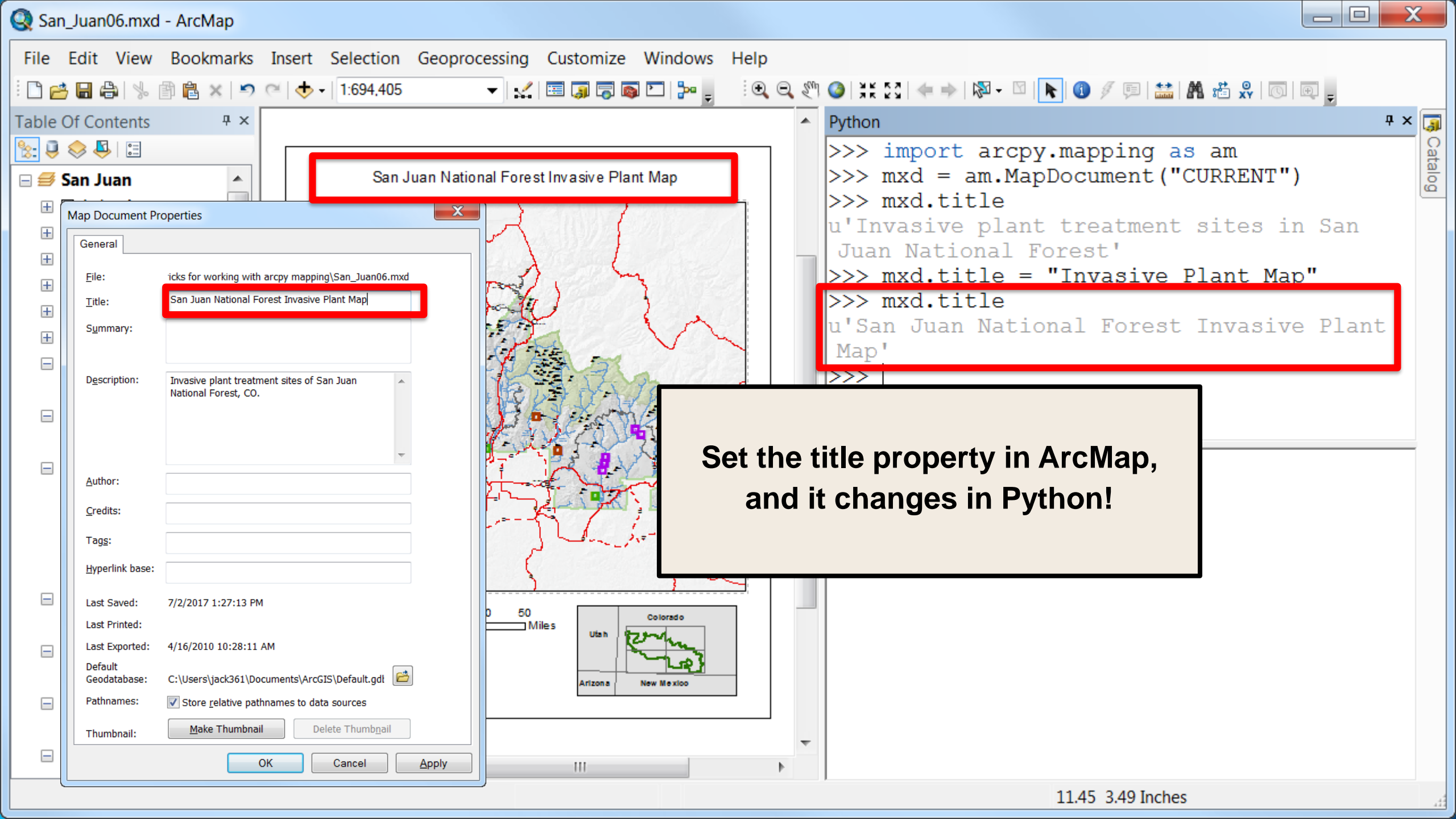
icks for working with arcpy mapping\San\_Juan06.mxd

Invasive Plant Map

Invasive plant treatment sites of San Juan National Forest, CO.

Set the title property in Python, and it changes in ArcMap!

```
>>> import arcpy.mapping as am
>>> mxd = am.MapDocument("CURRENT")
>>> mxd.title
u'Invasive plant treatment sites in San
Juan National Forest'
>>> mxd.title = "Invasive Plant Map"
>>>
```



San Juan National Forest Invasive Plant Map

#### Map Document Properties

##### General

File: c:\Users\jack361\Documents\ArcGIS\San\_Juan06.mxd

Title: San Juan National Forest Invasive Plant Map

Summary:

Description: Invasive plant treatment sites of San Juan National Forest, CO.

Author:

Credits:

Tags:

Hyperlink base:

Last Saved: 7/2/2017 1:27:13 PM

Last Printed:

Last Exported: 4/16/2010 10:28:11 AM

Default Geodatabase: C:\Users\jack361\Documents\ArcGIS\Default.gdb

Pathnames: ☒ Store relative pathnames to data sources

Thumbnail:

OK

Cancel

Apply

#### Python

```
>>> import arcpy.mapping as am
>>> mxd = am.MapDocument("CURRENT")
>>> mxd.title
u'Invasive plant treatment sites in San
Juan National Forest'
>>> mxd.title = "Invasive Plant Map"
>>> mxd.title
u'San Juan National Forest Invasive Plant
Map'
>>>
```

**Set the title property in ArcMap,  
and it changes in Python!**

## San Juan

- ☒ LakesAnno
- ☒ WildernessAnno
- ☒ CitiesAnno
- ☒ Spot\_HeightsAnno
- ☒ RoadsAnno
- ☒ Invasive Plants
- ☒ Cities
- ☐ Mountain peaks
- ☒ Roads
  - ROUTE\_TYPE
  - Primary road
  - Secondary road
  - Light duty road
- ☒ Streams
- ☒ Lakes
- ☐ index
- ☒ Wilderness

## Properties

## Text Size and Position

## Text

<dyn type="document" property="title"/>

Font: Arial 25.00

Angle: 0.00

Character Spacing: 0.00

Leading: 0.00

[About formatting text](#)

Change Symbol...

OK

Cancel

Apply

## Python

```
>>> import arcpy.mapping as am
>>> mxd = am.MapDocument("CURRENT")
>>> mxd.title
u'Invasive plant treatment sites in San
Juan National Forest'
>>> mxd.title = "Invasive Plant Map"
>>> mxd.title
u'San Juan National Forest Invasive Plant
Map'
>>>
```

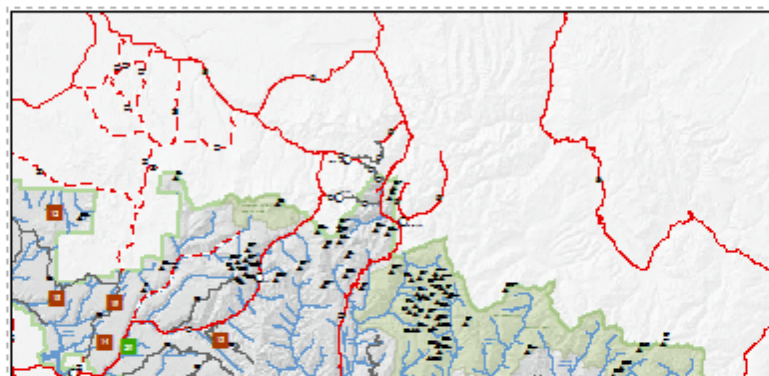
**The map title is Dynamic Text,  
displaying the  
Title property of the Document**



## San Juan

- ☒ LakesAnno
- ☒ WildernessAnno
- ☒ CitiesAnno
- ☒ Spot\_HeightsAnno
- ☒ RoadsAnno
- ☒ Invasive Plants
- ☒ Cities
- ☐ Mountain peaks
- ☒ Roads
  - ROUTE\_TYPE
  - Primary road
  - Secondary road
  - Light duty road
- ☒ Streams
- ☒ Lakes
- ☐ index
- ☒ Wilderness

## San Juan National Forest Invasive Plant Map



There are lots of other useful  
map document properties,  
such as dateSaved

```
>>> import arcpy.mapping as am
>>> mxd = am.MapDocument("CURRENT")
>>> mxd.title
u'Invasive plant treatment sites in San
Juan National Forest'
>>> mxd.title = "Invasive Plant Map"
>>> mxd.title
u'San Juan National Forest Invasive Plant
Map'
>>> mxd.
```

mxd.  
Provides a  
and method  
essential

- dataDrivenPages
- dateExported
- datePrinted
- dateSaved**
- deleteThumbnail
- description
- filePath

mxd ) properties  
object is  
erations.

File Edit View Bookmarks Insert Selection Geoprocessing Customize Windows Help

1:694,405

Table Of Contents

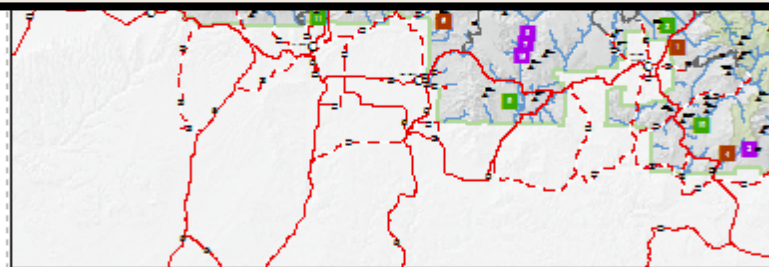
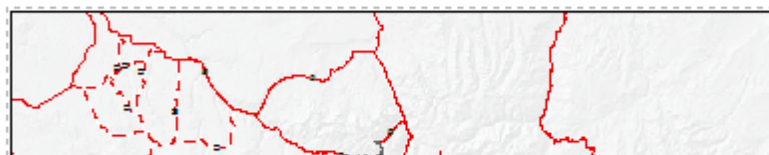


## San Juan

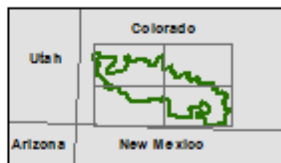
- ☒ LakesAnno
- ☒ WildernessAnno
- ☒ CitiesAnno
- ☒ Spot\_HeightsAnno
- ☒ RoadsAnno
- ☒ Invasive P
- ☒ Cities
- ☐ Mountain
- ☒ Roads
  - ROUTE\_TYPE
  - Primary road
  - Secondary road
  - Light duty road
- ☒ Streams
- ☒ Lakes
- ☐ index
- ☒ Wilderness

**You can read dateSaved,  
but you can not modify it,  
because it is a Read Only property**

## San Juan National Forest Invasive Plant Map



0 5 10 20 30 40 50 Miles



Python

```
>>> mxd.dateSaved  
datetime.datetime(2017, 7, 2, 13, 27, 13)  
>>> mxd.dateSaved = "May 1, 2017"
```

**Runtime error**

```
Traceback (most recent call last):  
  File "<string>", line 1, in <module>  
  File "c:\program files (x86)\arcgis  
    \desktop10.5\arcpy\arcpy\arcobjects  
    \_base.py", line 94, in _set  
    (attr_name, self.__class__.__name__))  
NameError: The attribute 'dateSaved' is  
not supported on this instance of
```

ArcGIS 10.5 Help

Hide

Back

Forward

Home

Options

Contents

Favorites

Search

Type a question, then click Ask.

mapdocument

Powered by AnswerWorks®

ask

History

35 topics found

Click to view

Rank

MapDocument (arcpy.mapping)

MapDocument (arcpy.mapping)

Alphabetical list of arcpy.mapping functions (arcpy.mapping)

Getting started with arcpy.mapping tutorial (arcpy.mapping)

CreateMapSDDraft (arcpy.mapping)

ListBookmarks (arcpy.mapping)

ExportToEPS (arcpy.mapping)

ExportToBMP (arcpy.mapping)

ListLayoutElements (arcpy.mapping)

ExportToJPEG (arcpy.mapping)

More...

		next time the map document is opened, it will be opened in data view, and that particular data frame will be the active data frame.
author (Read and Write)		Provides the ability to either get or set the map document's author information.
credits (Read and Write)		Provides the ability to either get or set the map document's credits or copyright information.
dataDrivenPages (Read Only)		Returns a <a href="#">DataDrivenPages</a> object that can then be used to manage the pages in a Data Driven Pages enabled map document.
<div>The help documentation for MapDocument tells you which properties are Read Only and which are Read Write</div>		
		after the map was printed.
dateSaved (Read Only)		Returns a Python <code>datetime</code> object that reports the date the last time the map document was saved.
description (Read and Write)		Provides the ability to either get or set the map document's description information.
filePath (Read Only)		Returns a string value that reports the fully qualified map document path and file name.
hyperlinkBase (Read and Write)		Provides the ability to either get or set the base path or URL used for field-based hyperlinks to documents or URLs.
isDDPEnabled		



- Exporting and Printing Maps
- Managing Documents and Layers
  - AddLayer
  - AddLayerToGroup
  - AddTableView
  - AnalyzeForMSD
  - AnalyzeForSD
  - ConvertToMSD

Another thing you can do with a map document is list its layout elements

- ListDataFrames
- ListLayers
- ListLayoutElements
- ListMapServices
- ListStyleItems
- ListTableViews
- MapDocument
- MoveLayer
- PDFDocumentCreate
- PDFDocumentOpen
- PublishMSDToServer
- RemoveLayer
- RemoveTableView
- TableView
- UpdateLayer

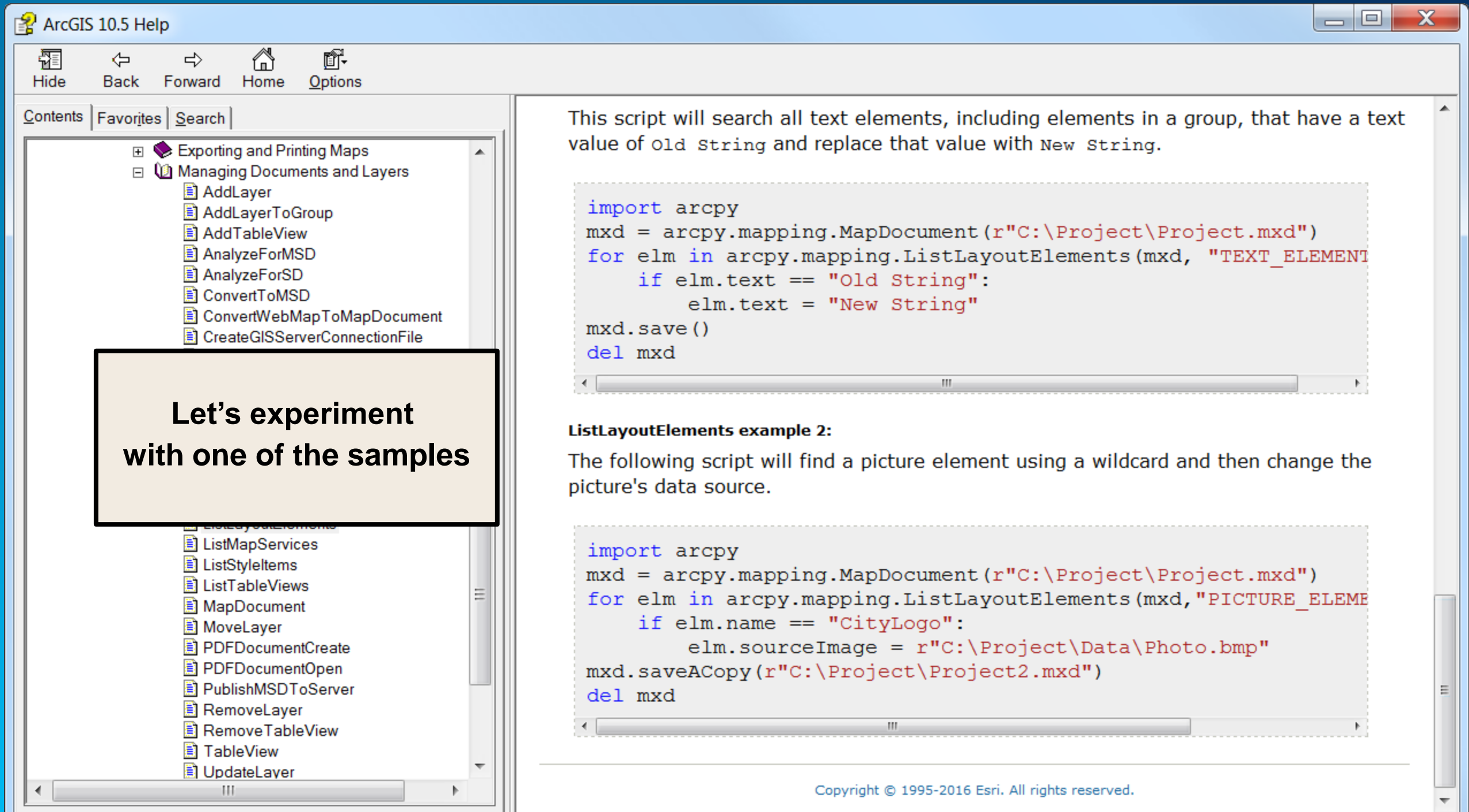
## Syntax

**ListLayoutElements (map\_document, {element\_type}, {wildcard})**

Parameter	Explanation	Data Type
map_document	A variable that references a <a href="#">MapDocument</a> object.	<a href="#">MapDocument</a>
element_type	A string that represents the element type that will be used to filter the returned list of elements. <ul style="list-style-type: none"><li>DATAFRAME_ELEMENT —Dataframe element</li><li>GRAPHIC_ELEMENT —Graphic element</li><li>LEGEND_ELEMENT —Legend element</li><li>MAPSURROUND_ELEMENT —Mapsurround element</li><li>PICTURE_ELEMENT —Picture element</li><li>TEXT_ELEMENT —Text element</li></ul> (The default value is None)	String
wildcard	A combination of asterisks (*) and characters can be used to help limit the results. (The default value is None)	String

## Return Value

Data Type	Explanation
Object	A Python list of page layout elements. The types of objects that can be returned are: <a href="#">DataFrame</a> , <a href="#">GraphicElement</a> , <a href="#">LegendElement</a> , <a href="#">MapsurroundElement</a> , <a href="#">PictureElement</a> , and <a href="#">TextElement</a> .



- Exporting and Printing Maps
- Managing Documents and Layers
  - AddLayer
  - AddLayerToGroup
  - AddTableView
  - AnalyzeForMSD
  - AnalyzeForSD
  - ConvertToMSD
  - ConvertWebMapToMapDocument
  - CreateGISServerConnectionFile

Let's experiment  
with one of the samples

- ListLayoutElements
- ListMapServices
- ListStyleItems
- ListTableViews
- MapDocument
- MoveLayer
- PDFDocumentCreate
- PDFDocumentOpen
- PublishMSDToServer
- RemoveLayer
- RemoveTableView
- TableView
- UpdateLayer

This script will search all text elements, including elements in a group, that have a text value of Old String and replace that value with New String.

```
import arcpy
mxd = arcpy.mapping.MapDocument(r"C:\Project\Project.mxd")
for elm in arcpy.mapping.ListLayoutElements(mxd, "TEXT_ELEMENT"):
    if elm.text == "Old String":
        elm.text = "New String"
mxd.save()
del mxd
```

#### ListLayoutElements example 2:

The following script will find a picture element using a wildcard and then change the picture's data source.

```
import arcpy
mxd = arcpy.mapping.MapDocument(r"C:\Project\Project.mxd")
for elm in arcpy.mapping.ListLayoutElements(mxd, "PICTURE_ELEMENT"):
    if elm.name == "CityLogo":
        elm.sourceImage = r"C:\Project\Data\Photo.bmp"
mxd.saveACopy(r"C:\Project\Project2.mxd")
del mxd
```

ArcGIS 10.5 Help

HideBackForwardHomeOptions

ContentsFavoritesSearch

Exporting and Printing Maps

Managing Documents and Layers

- AddLayer
- AddLayerToGroup
- AddTableView
- AnalyzeForMSD
- AnalyzeForSD
- ConvertToMSD
- ConvertWebMapToMapDocument
- CreateGISServerConnectionFile

ListLayoutElements

ListMapServices

ListStyleItems

ListTableViews

MapDocument

MoveLayer

PDFDocumentCreate

PDFDocumentOpen

PublishMSDToServer

RemoveLayer

RemoveTableView

TableView

UpdateLayer

This script will search all text elements, including elements in a group, that have a text value of Old String and replace that value with New String.

```
import arcpy
mxd = arcpy.mapping.MapDocument(r"C:\Project\Project.mxd")
for elm in arcpy.mapping.ListLayoutElements(mxd, "TEXT_ELEMENT"):
    if elm.text == "Old String":
        elm.text = "New String"
mxd.save()
del mxd
```

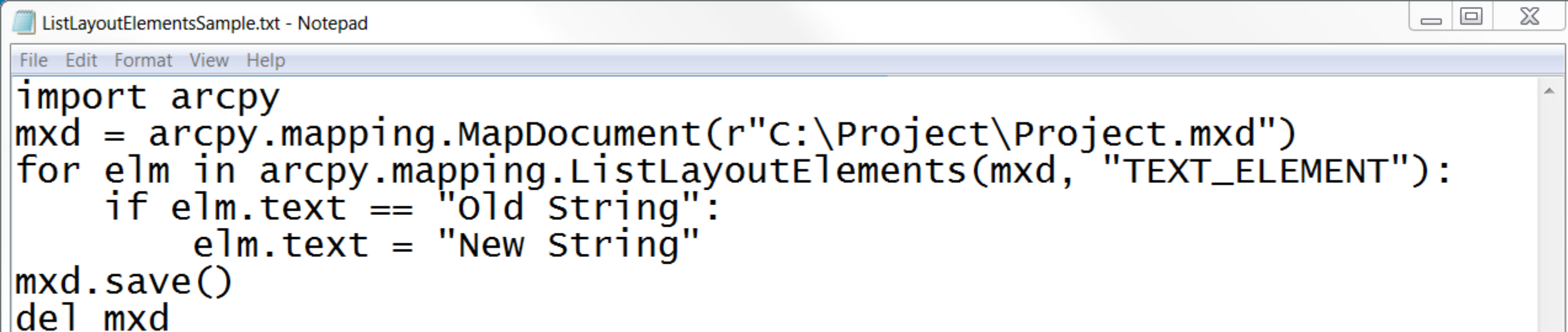
**ListLayoutElements example 2:**

The following script will find a picture element using a wildcard and then change the picture's data source.

```
import arcpy
mxd = arcpy.mapping.MapDocument(r"C:\Project\Project.mxd")
for elm in arcpy.mapping.ListLayoutElements(mxd, "PICTURE_ELEMENT"):
    if elm.name == "CityLogo":
        elm.sourceImage = r"C:\Project\Data\Photo.bmp"
mxd.saveACopy(r"C:\Project\Project2.mxd")
del mxd
```

Copyright © 1995-2016 Esri. All rights reserved.

Copy ...



The image shows a Notepad window titled "ListLayoutElementsSample.txt - Notepad". The menu bar includes "File", "Edit", "Format", "View", and "Help". The text area contains the following Python code:

```
import arcpy
mxd = arcpy.mapping.MapDocument(r"C:\Project\Project.mxd")
for elm in arcpy.mapping.ListLayoutElements(mxd, "TEXT_ELEMENT"):
    if elm.text == "Old String":
        elm.text = "New String"
mxd.save()
del mxd
```

**... and paste to Notepad or your favorite Python editor.  
We need to modify this sample before we use it.**



```
import arcpy
mxd = arcpy.mapping.MapDocument(r"C:\Project\Project.mxd")
for elm in arcpy.mapping.ListLayoutElements(mxd, "TEXT_ELEMENT"):
    if elm.text == "Old String":
        elm.text = "New String"
mxd.save()
del mxd
```

**For example, our mxd variable is already set**

```
for elm in arcpy.mapping.ListLayoutElements(mxd, "TEXT_ELEMENT"):  
    if elm.text == "Old String":  
        elm.text = "New String"  
mxd.save()  
del mxd
```

**... and lets list all the elements,  
not just TEXT\_ELEMENTS**

```
for elm in arcpy.mapping.ListLayoutElements(mxd):  
    if elm.text == "Old String":  
        elm.text = "New String"  
mxd.save()  
del mxd
```

**... And if we are going to list all the elements,  
most of them will not have text properties**

```
for elm in arcpy.mapping.ListLayoutElements(mxd):
```

```
    mxd.save()
```

```
del mxd
```



```
for elm in arcpy.mapping.ListLayoutElements(mxd):  
    print(elm.name + " - " + elm.type)  
mxd.save()  
del mxd
```

**Every element has a name and  
a type, so lets print them out**

```
for elm in arcpy.mapping.ListLayoutElements(mxd):  
    print(elm.name + " - " + elm.type)  
mxd.save()  
del mxd
```

**And we do not need to save the  
map document at this time**

```
for elm in arcpy.mapping.ListLayoutElements(mxd):  
    print(elm.name + " - " + elm.type)
```

```
del mxd
```

**Deleting the mxd variable does not delete the map document.  
It just releases Python's hold on it and clears the lock.  
We don't want to do that either.**

```
for elm in arcpy.mapping.ListLayoutElements(mxd):  
    print(elm.name + " - " + elm.type)
```

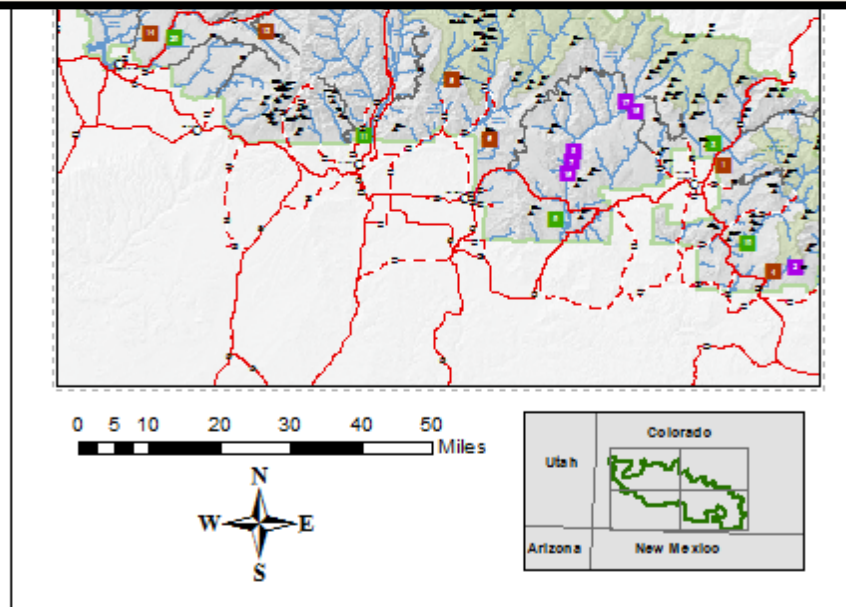
**Lets copy and paste these lines into the  
Python window to run them.**



## San Juan

- ☒ LakesAnd
- ☒ Wilderne
- ☒ CitiesAn
- ☒ Spot\_He
- ☒ RoadsAn
- ☒ Invasive
- ☒ Cities
- ☐ Mountain peaks
- ☒ Roads
  - ROUTE\_TYPE
  - Primary road
  - Secondary road
  - Light duty road
- ☒ Streams
- ☒ Lakes
- ☐ index
- ☒ Wilderness

The text wraps around in the window.  
We pasted a for loop, so we need to  
hit enter twice. That way the Python  
window knows we are done adding lines  
inside the loop.

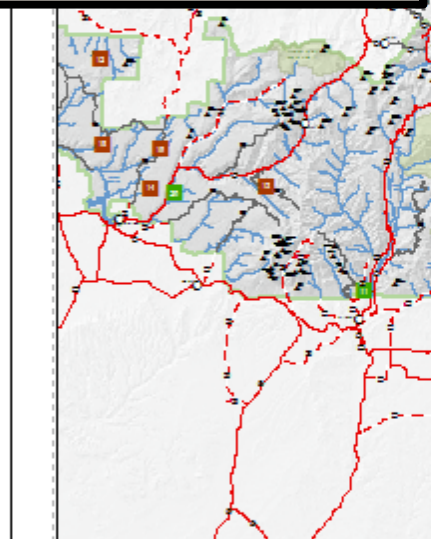


## Python

```
>>> for elm in
    arcpy.mapping.ListLayoutElements(mxd):
...     print(elm.name + " - " +
    elm.type)
...
North Arrow - MAPSURROUND_ELEMENT
- TEXT_ELEMENT
Alternating Scale Bar -
MAPSURROUND_ELEMENT
overview - DATAFRAME_ELEMENT
San Juan - DATAFRAME_ELEMENT
>>>
```

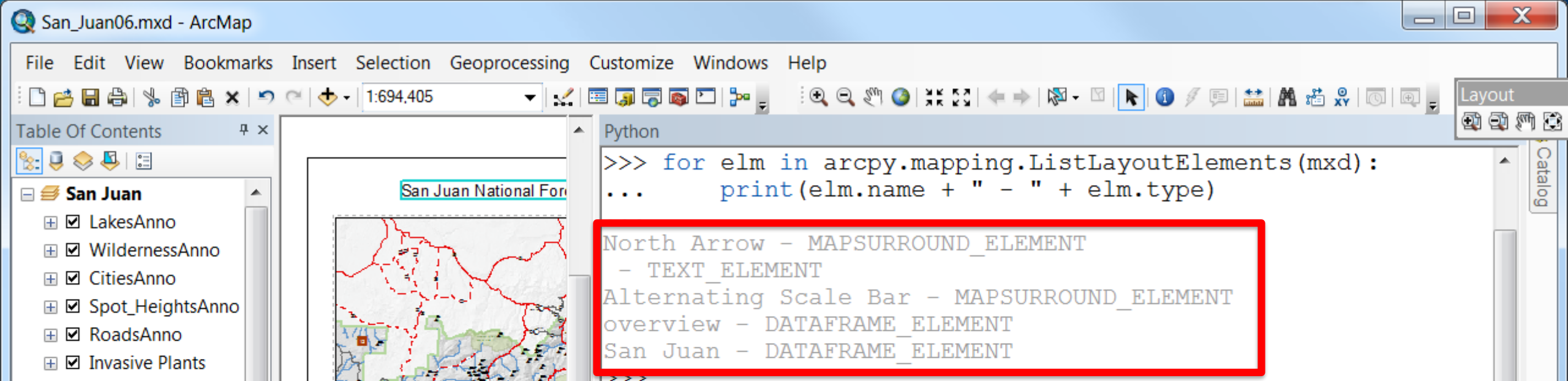
**Lets widen the Python window  
so we can see better**

- ☒ RoadsAnno
- ☒ Invasive Plants
- ☒ Cities
- ☒ Mountain peaks
- ☒ Roads
  - ROUTE\_TYPE
  - Primary road
  - Secondary road
  - Light duty road
- ☒ Streams
- ☒ Lakes
- ☐ index
- ☒ Wilderness



Python

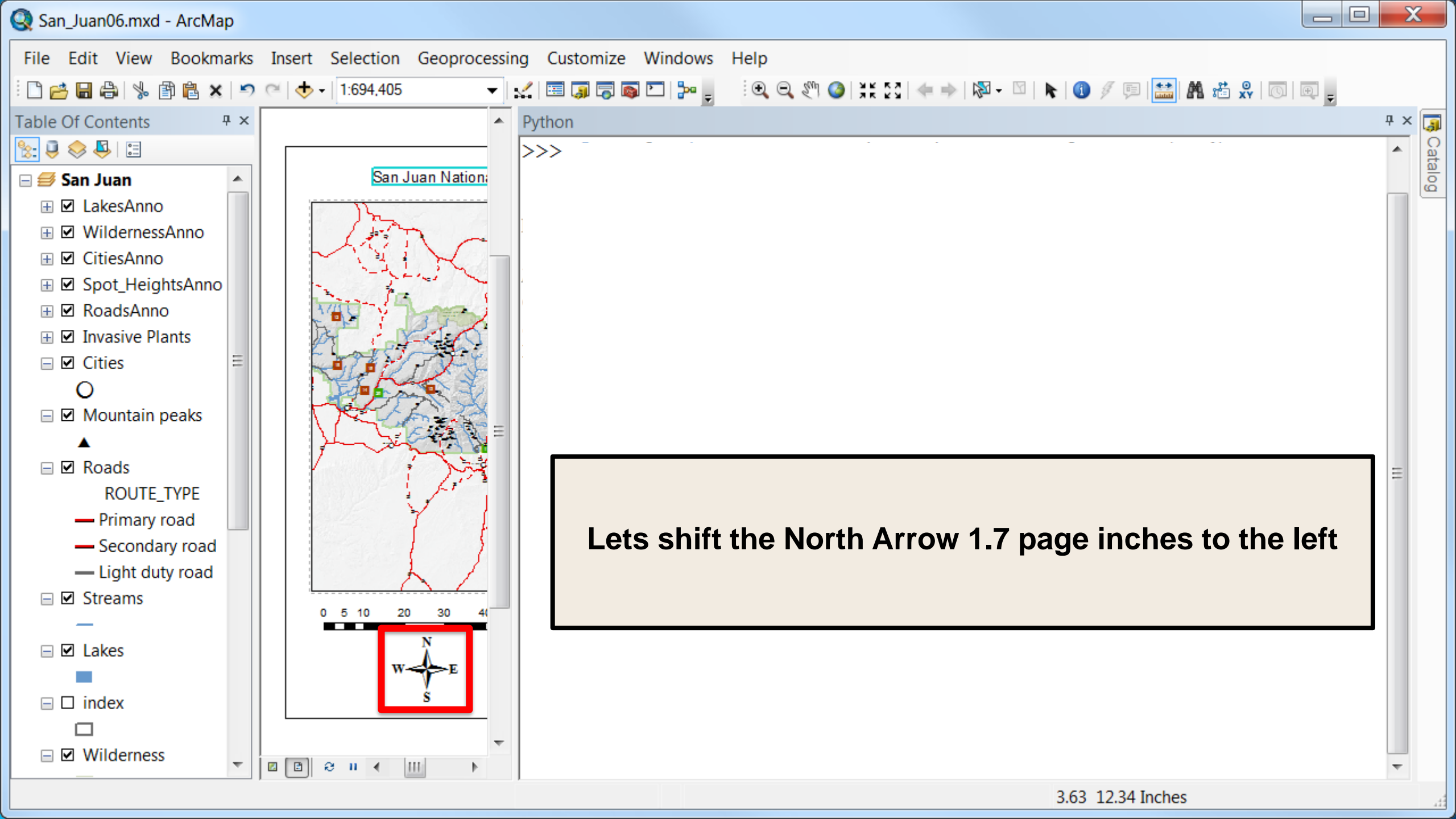
```
>>> for elm in arcpy.mapping.ListLayoutElements(mxd):  
...     print(elm.name + " - " + elm.type)  
...  
North Arrow - MAPSURROUND_ELEMENT  
- TEXT_ELEMENT  
Alternating Scale Bar - MAPSURROUND_ELEMENT  
overview - DATAFRAME_ELEMENT  
San Juan - DATAFRAME_ELEMENT  
>>>
```



**We have five elements on the layout:**

<b>NAME</b>	<b>TYPE</b>
<b>North Arrow</b>	<b>MAPSURROUND_ELEMENT</b>
	<b>TEXT_ELEMENT</b>
<b>Alternating Scale Bar</b>	<b>MAPSURROUND_ELEMENT</b>
<b>overview</b>	<b>DATAFRAME_ELEMENT</b>
<b>San Juan</b>	<b>DATAFRAME_ELEMENT</b>

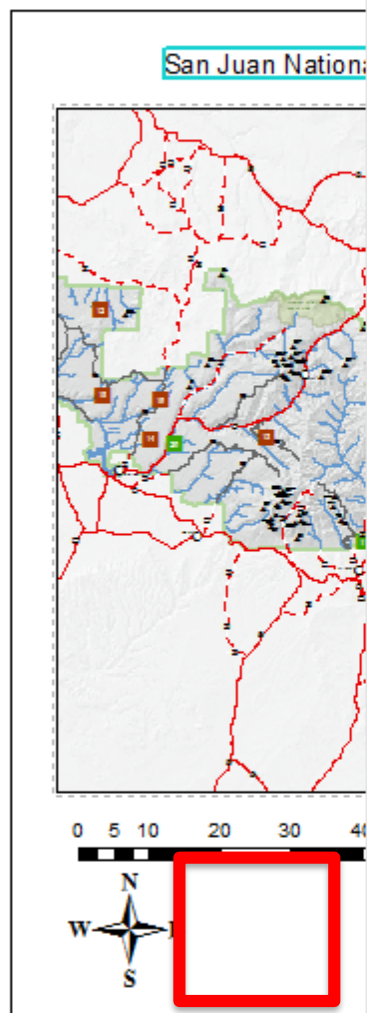
**The name property of the text element does not have anything in it**





## San Juan

- ☒ LakesAnno
- ☒ WildernessAnno
- ☒ CitiesAnno
- ☒ Spot\_HeightsAnno
- ☒ RoadsAnno
- ☒ Invasive Plants
- ☒ Cities
- ☐ Mountain peaks
- ☒ Roads
  - ROUTE\_TYPE
  - Primary road
  - Secondary road
  - Light duty road
- ☒ Streams
- ☒ Lakes
- ☐ index
- ☒ Wilderness

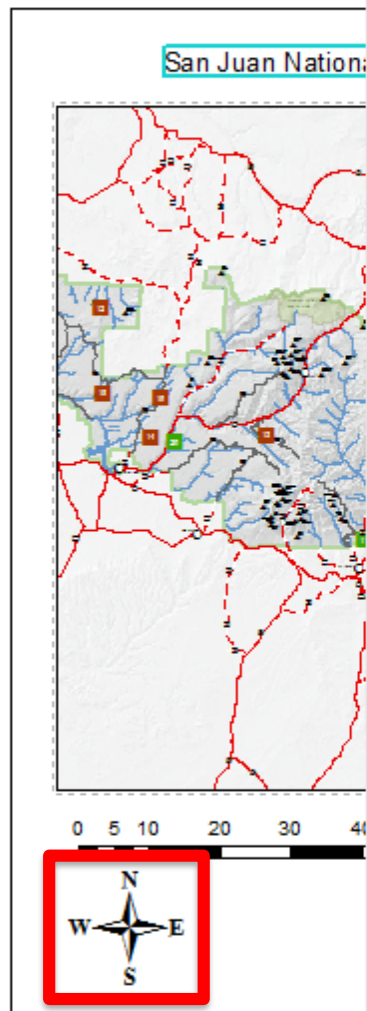


```
>>> for elm in arcpy.mapping.ListLayoutElements(mxd):  
...     if elm.name == "North Arrow":  
...         elm.elementPositionX = elm.elementPositionX - 1.7  
...  
>>> |
```

As you can see, only the North Arrow shifted ...

## San Juan

- ☒ LakesAnno
- ☒ WildernessAnno
- ☒ CitiesAnno
- ☒ Spot\_HeightsAnno
- ☒ RoadsAnno
- ☒ Invasive Plants
- ☒ Cities
- ☐ Mountain peaks
- ☒ Roads
  - ROUTE\_TYPE
  - Primary road
  - Secondary road
  - Light duty road
- ☒ Streams
- ☒ Lakes
- ☐ index
- ☒ Wilderness

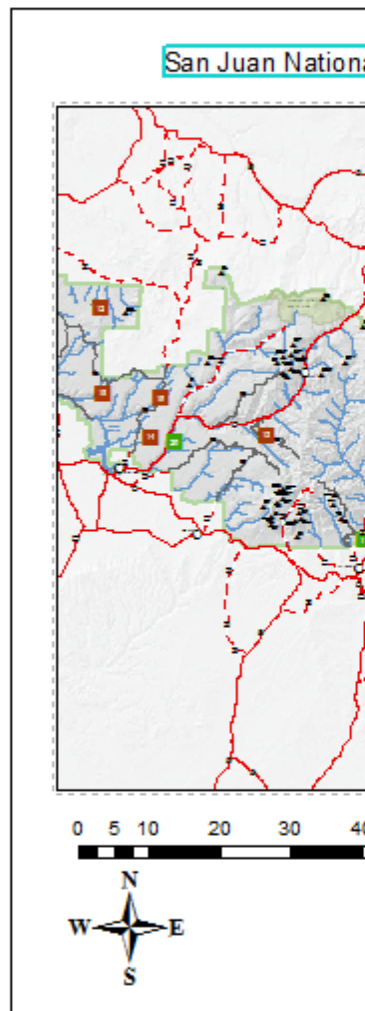


```
>>> for elm in arcpy.mapping.ListLayoutElements(mxd):  
...     if elm.name == "North Arrow":  
...         elm.elementPositionX = elm.elementPositionX - 1.7  
...  
>>> |
```

... and it shifted 1.7 page inches to the left

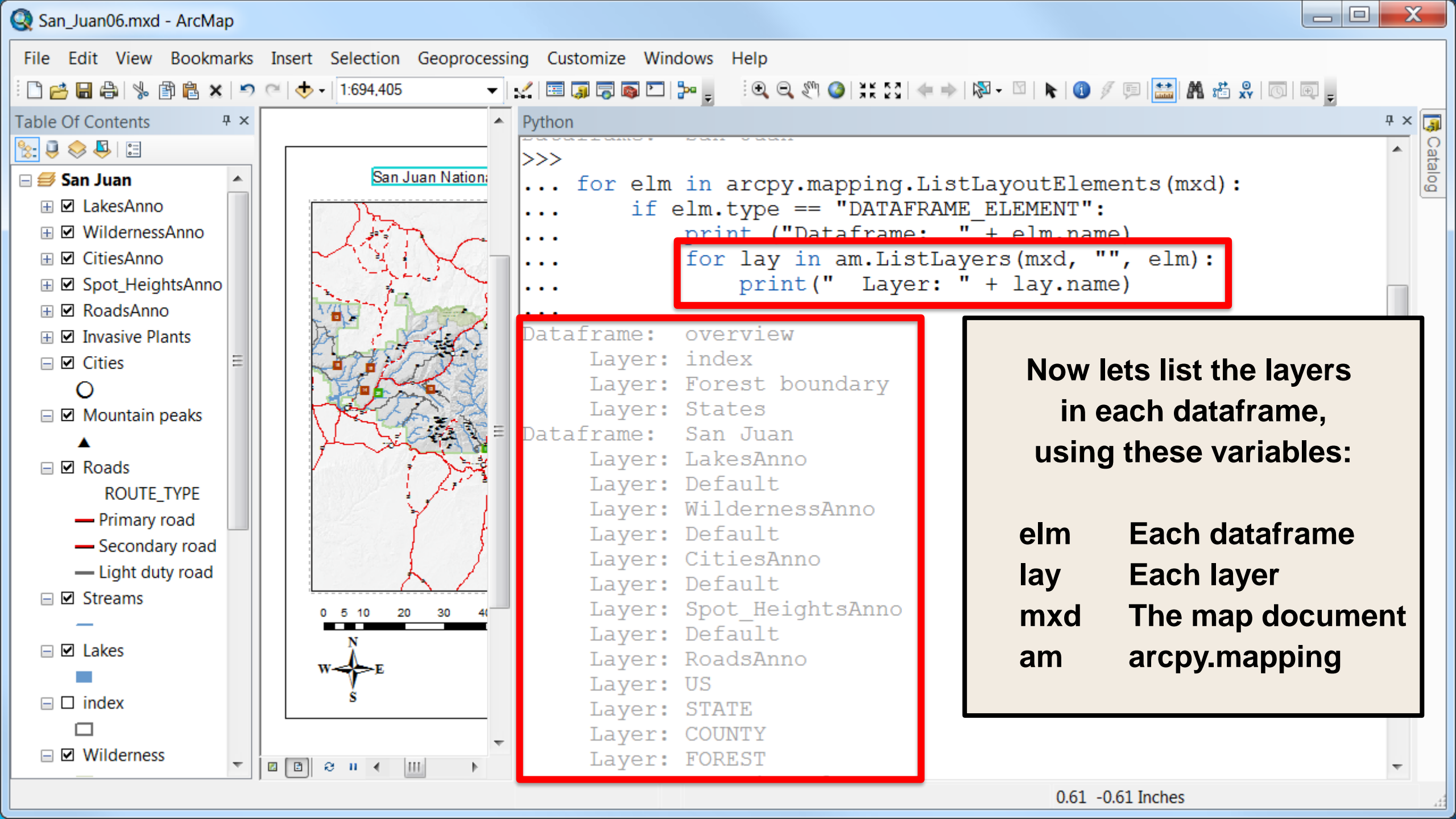
## San Juan

- ☒ LakesAnno
- ☒ WildernessAnno
- ☒ CitiesAnno
- ☒ Spot\_HeightsAnno
- ☒ RoadsAnno
- ☒ Invasive Plants
- ☒ Cities
- ☐ Mountain peaks
- ☒ Roads
  - ROUTE\_TYPE
  - Primary road
  - Secondary road
  - Light duty road
- ☒ Streams
- ☒ Lakes
- ☐ index
- ☒ Wilderness



```
>>> for elm in arcpy.mapping.ListLayoutElements(mxd):  
...     if elm.name == "North Arrow":  
...         elm.elementPositionX = elm.elementPositionX - 1.7  
...  
>>> for elm in arcpy.mapping.ListLayoutElements(mxd):  
...     if elm.type == "DATAFRAME_ELEMENT":  
...         print ("Dataframe: " + elm.name)  
...  
Dataframe: overview  
Dataframe: San Juan  
>>>
```

Lets list the names of just the dataframes



- Updating and fixing data sources with arcpy.m...
  - Classes
    - DataDrivenPages
    - DataFrameTime
    - DataFrame
    - GraduatedColorsSymbology
    - GraduatedSymbolsSymbology
    - GraphicElement
    - LabelClass
    - Layer
    - LayerTime
    - LegendElement
    - MapDocument
    - MapsurroundElement
    - PDFDocument
    - PictureElement
    - RasterClassifiedSymbology
    - StyleItem
    - TableView
    - TextElement
    - UniqueValuesSymbology
  - Functions
    - Exporting and Printing Maps
    - Managing Documents and Layers
      - AddLayer
      - AddLayerToGroup
      - AddTableView
      - AnalyzeForMSD
      - AnalyzeForSD
      - ConvertToMSD
      - ConvertWebMapToMapDocument

Property	Explanation	Data Type
brightness (Read and Write)	Provides the ability to get or set the brightness value. The default, normal brightness, is 0%. Enter any value between +100% and -100%. Enter a plus or minus sign to the left of the value to specify whether it is above or below 0. Not all layers support the <code>brightness</code> property (for example, group layers and feature layers), so it is good practice to test for this ahead of time using the <code>supports</code> method.	Integer
contrast (Read and Write)	Provides the ability to get or set the layer's contrast or contrast information. Not all layers support the <code>contrast</code> property (for example, annotation layers and fabric layers), so it is good practice to test for this ahead of time using the <code>supports</code> method.	
credits (Read and Write)	Provides the ability to either get or set the layer's credits or copyright information.	String
datasetName (Read Only)	Returns the name of the layer's dataset the way it appears in the workspace, not in the TOC. Not all layers support the <code>datasetName</code> property (for example, web services), so it is good practice to test for this ahead of time using the <code>supports</code> method.	String
dataSource (Read Only)	Returns the complete path for the layer's data	String

**There are many kinds of layers and they have many different properties. Not all layers support all properties.**



Updating and fixing data sources with arcpy.m...

Classes

- DataDrivenPages
- DataFrameTime
- DataFrame
- GraduatedColorsSymbology
- GraduatedSymbolsSymbology
- GraphicElement
- LabelClass
- Layer**
- LayerTime
- LegendElement
- MapDocument
- MapsurroundElement

**The supports method lets you test ahead of time and avoid errors**

Functions

- Exporting and Printing Maps
- Managing Documents and Layers
  - AddLayer
  - AddLayerToGroup
  - AddTableView
  - AnalyzeForMSD
  - AnalyzeForSD
  - ConvertToMSD
  - ConvertWebMapToMapDocument

### supports (layer\_property)

Parameter	Explanation	Data Type
layer_property	<p>The name of a particular layer property that will be tested.</p> <ul style="list-style-type: none"><li>BRIGHTNESS —A raster layer's brightness value</li><li>CONTRAST —A raster layer's contrast value</li><li>DATASETNAME —A layers dataset name the way it appears in the workspace</li><li>DATASOURCE —A layer's file path or connection file</li><li>DEFINITIONQUERY —A layer's definition query string</li><li>DESCRIPTION —A layer's description string</li><li>LABELCLASSES —A layer's list of label classes</li><li>LONGNAME —A layer's path including the group layer(s) it may be nested within</li><li>NAME —A layer's name</li><li>SERVICEPROPERTIES —Connection information for SDE and web service layers</li><li>SHOWLABELS —A Boolean indicating if a layer's labels are toggled on or off</li><li>SYMBOLOLOGY —A layer's symbology class</li><li>SYMBOLOLOGYTYPE —A layer's symbology class type</li><li>TIME —A layer's time properties</li><li>TRANSPARENCY —A layer's transparency value</li></ul>	String

```
for elm in arcpy.mapping.ListLayoutElements(mxd):  
    if elm.type == "DATAFRAME_ELEMENT":  
        print ("Dataframe: " + elm.name)  
        for lay in am.ListLayers(mxd, "", elm):  
            print("      Layer: " + lay.name)
```

```
-----  
for elm in arcpy.mapping.ListLayoutElements(mxd):  
    if elm.type == "DATAFRAME_ELEMENT":  
        print ("Dataframe: " + elm.name)  
        for lay in am.ListLayers(mxd, "", elm):  
            if lay.supports("WORKSPACEPATH"):  
                print("      Layer: " + lay.name)  
                print("      Path: " + lay.workspacePath)
```

**For example, a basemap layer does not have a workspace path. Lets modify our script to print out the workspace path of each layer, but only if it has one!**

San\_Juan07.mxd - ArcMap

File Edit View Bookmarks Insert Selection Geoprocessing Customize Windows Help

1:694,405

Table Of Contents

San Juan

LakesAnno

Mountain peaks

Roads

ROUTE\_TYPE

Primary road

Secondary road

Light duty road

Streams

Lakes

For our example, layers have different workspaces, but they are all in the D:\temp folder.

Python

```
>>> import arcpy.mapping as am
... mxd = am.MapDocument("CURRENT")
... for elm in arcpy.mapping.ListLayoutElements(mxd):
...     if elm.type == "DATAFRAME_ELEMENT":
...         print("Dataframe: " + elm.name)
...         for lay in am.ListLayers(mxd, "", elm):
...             if lay.supports("WORKSPACEPATH"):
...                 print("    Layer: " + lay.name)
...                 print("    Path: " + lay.workspacePath)
...                 print("")
...
Dataframe: overview
    Layer: index
    Path: D:\temp\Database


    Layer: Forest boundary
    Path: D:\temp\Database


    Layer: States
    Path: D:\temp\Database\San_Juan.gdb

Dataframe: San Juan
    Layer: LakesAnno
```

Classes

## DataFrameTime

 GraduatedColorsSymbology

 GraphicElement


 Layer

 LegendElement

 **MapsurroundElement**


## PictureElement

 **StyleItem**



 AddTableView

AnalyzeForSD

 ConvertToMSD

 ConvertWebMa

111

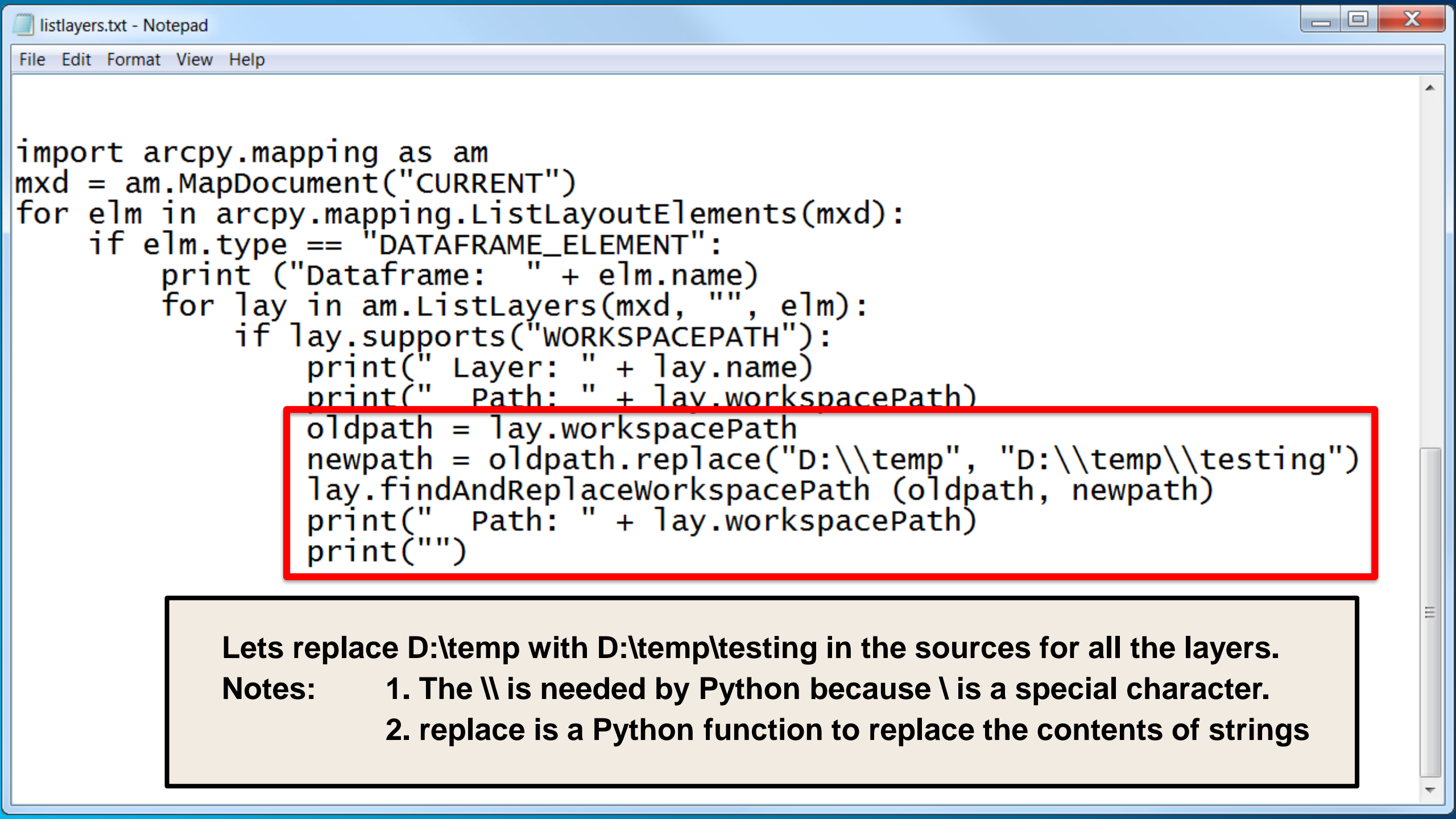
## Lets point the layers to a different folder

```
findAndReplaceWorkspacePath (find_workspace_path, replace_workspace_path,
{validate})
```

Parameter	Explanation	Data Type
find_workspace_path	A string that represents the workspace path or connection file you want to find. If an empty string is passed, then all workspace paths will be replaced with the <code>replace_workspace_path</code> parameter depending on the value of the <code>validate</code> parameter.	String
replace_workspace_path	A string that represents the workspace path or connection file you want to replace.	String
validate	<p>If set to <code>True</code>, the workspace will only be updated if the <code>replace_workspace_path</code> value is a valid workspace. If it is not valid, the workspace will not be replaced. If set to <code>False</code>, the method will set the workspace to match the <code>replace_workspace_path</code>, regardless of a valid match. In this case, if a match does not exist, then the layer's data source would be broken.</p> <p>(The default value is <code>True</code>)</p>	Boolean

For more detailed discussion, parameter information, scenarios, and code samples, please refer to the [Updating and fixing data sources with `arcpy.mapping`](#) help topic.

**getExtent ({symbolized\_extent})**



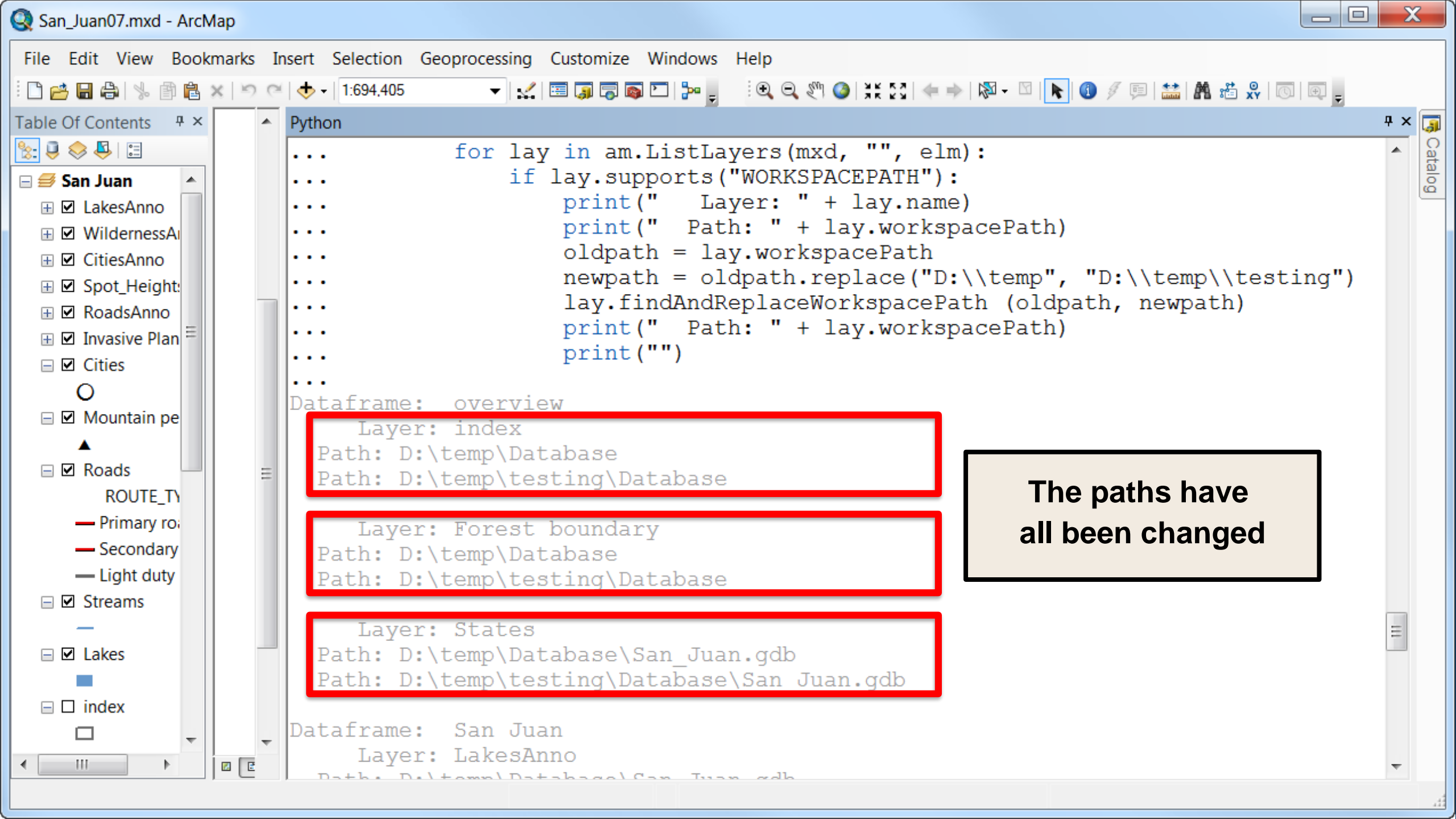
```
import arcpy.mapping as am
mxd = am.MapDocument("CURRENT")
for elm in arcpy.mapping.ListLayoutElements(mxd):
    if elm.type == "DATAFRAME_ELEMENT":
        print ("Dataframe: " + elm.name)
        for lay in am.ListLayers(mxd, "", elm):
            if lay.supports("WORKSPACEPATH"):
                print(" Layer: " + lay.name)
                print(" Path: " + lay.workspacePath)
                oldpath = lay.workspacePath
                newpath = oldpath.replace("D:\\temp", "D:\\temp\\testing")
                lay.findAndReplaceWorkspacePath (oldpath, newpath)
                print(" Path: " + lay.workspacePath)
                print("")
```

**Lets replace D:\\temp with D:\\temp\\testing in the sources for all the layers.**

**Notes:**

- 1. The \\ is needed by Python because \\ is a special character.
- 2. replace is a Python function to replace the contents of strings





Python

```
... for lay in am.ListLayers(mxd, "", elm):
...     if lay.supports("WORKSPACEPATH"):
...         print("    Layer: " + lay.name)
...         print("    Path: " + lay.workspacePath)
...         oldpath = lay.workspacePath
...         newpath = oldpath.replace("D:\\temp", "D:\\temp\\testing")
...         lay.findAndReplaceWorkspacePath (oldpath, newpath)
...         print("    Path: " + lay.workspacePath)
...         print("")
... 
```

Dataframe: overview

Layer: index  
Path: D:\\temp\\Database  
Path: D:\\temp\\testing\\Database

Layer: Forest boundary  
Path: D:\\temp\\Database  
Path: D:\\temp\\testing\\Database

Layer: States  
Path: D:\\temp\\Database\\San\_Juan.gdb  
Path: D:\\temp\\testing\\Database\\San\_Juan.gdb

Dataframe: San Juan

Layer: LakesAnno  
Path: D:\\temp\\Database\\San\_Juan.gdb

The paths have  
all been changed

Type a question, then click Ask.

listfiles

Powered by AnswerWorks®



History

28 topics found

Click to view	Rank
<a href="#">ListFiles (arcpy)</a>	
<a href="#">KML To Layer (Conversion)</a>	
<a href="#">Upload Service Definition (Server)</a>	
<a href="#">Python equivalents to AML directives</a>	
<a href="#">Create lists of data</a>	
<a href="#">Package Result (Data Management)</a>	
<a href="#">Package Layer (Data Management)</a>	
<a href="#">Extract LAS (3D Analyst)</a>	
<a href="#">Extract Package (Data Management)</a>	
<a href="#">Layer 3D To Feature Class (3D Analyst)</a>	

[More...](#)

Now lets use ListFiles() to fix  
all the .mxd files in a folder

## ListFiles (arcpy)

ArcGIS 10.5

[Locate topic](#)

### Summary

Returns a list of files in the current workspace based on a query string. Specifying search conditions can be used to limit the results.

### Discussion

The workspace environment must be set first before using several of the List functions, including [ListDatasets](#), [ListFeatureClasses](#), [ListFiles](#), [ListRasters](#), [ListTables](#), and [ListWorkspaces](#).

### Syntax

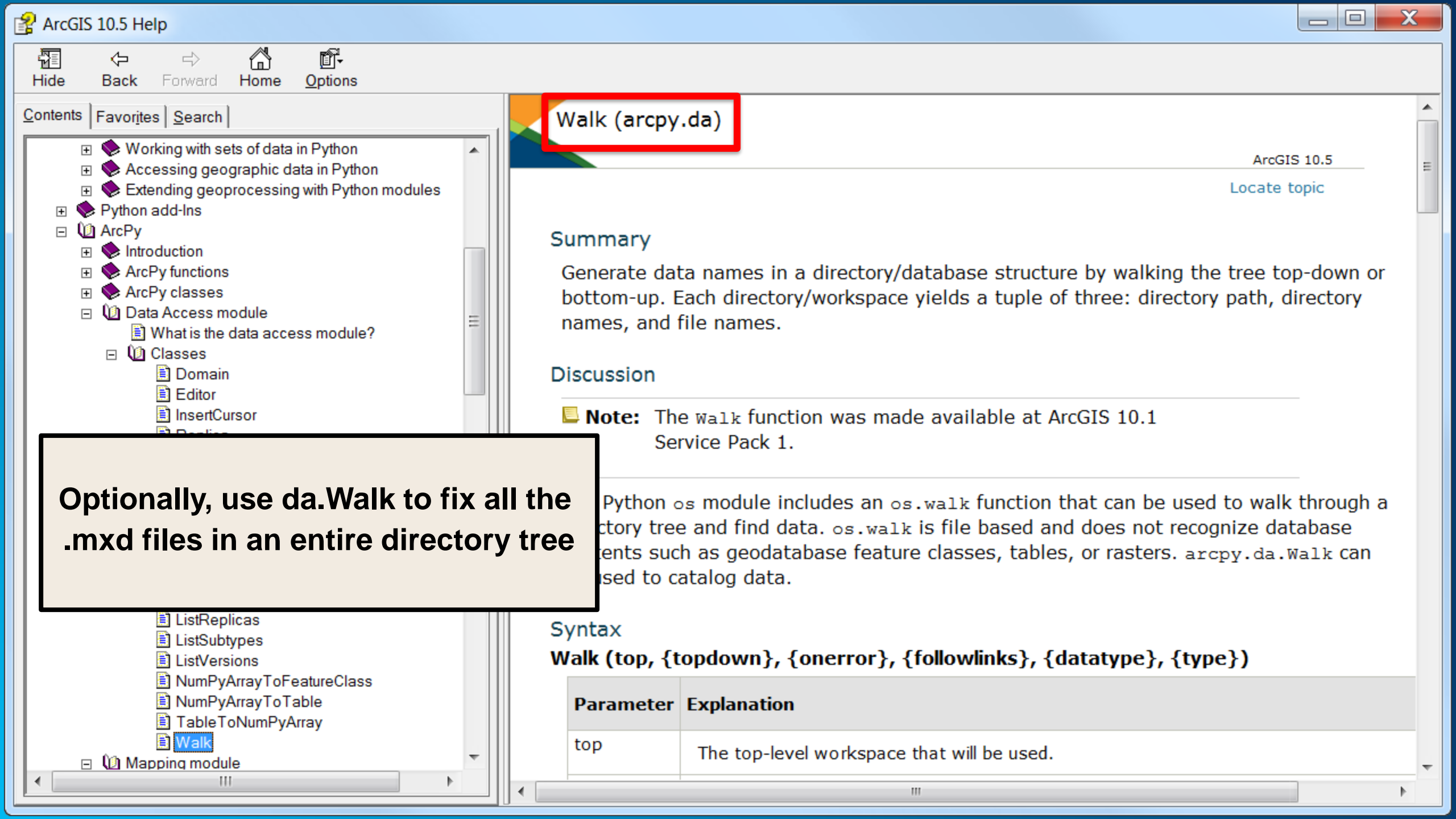
#### ListFiles ({wild\_card})

Parameter	Explanation	Data Type
wild_card	The wild_card limits the results returned. If no wild_card is specified, all values are returned.	String

#### Return Value

Data Type	Explanation
String	A list of files.

### Code sample



Optionally, use `da.Walk` to fix all the `.mxd` files in an entire directory tree

## Walk (arcpy.da)

ArcGIS 10.5

[Locate topic](#)

### Summary

Generate data names in a directory/database structure by walking the tree top-down or bottom-up. Each directory/workspace yields a tuple of three: directory path, directory names, and file names.

### Discussion

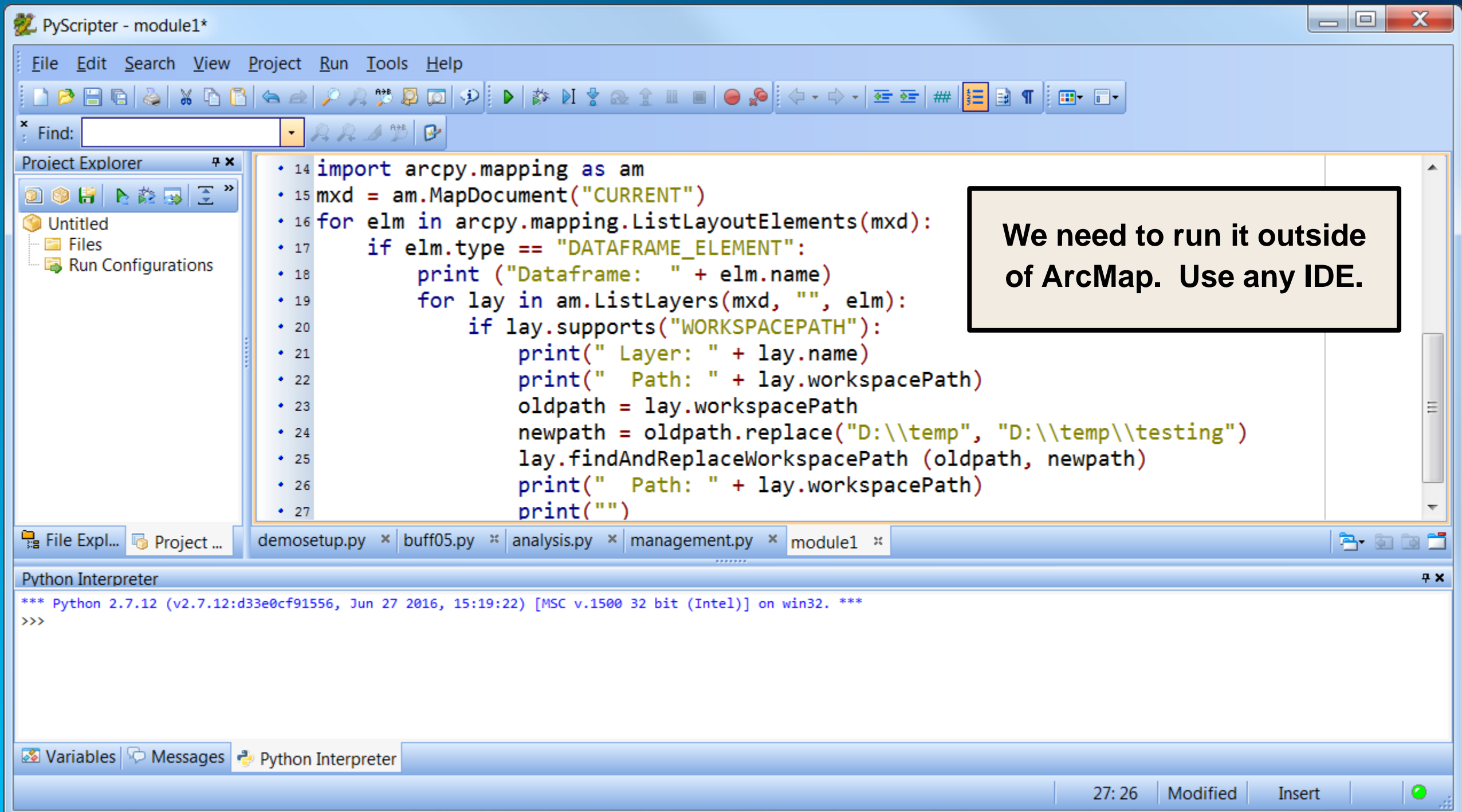
**Note:** The `walk` function was made available at ArcGIS 10.1 Service Pack 1.

The Python `os` module includes an `os.walk` function that can be used to walk through a directory tree and find data. `os.walk` is file based and does not recognize database elements such as geodatabase feature classes, tables, or rasters. `arcpy.da.Walk` can be used to catalog data.

### Syntax

**Walk (top, {topdown}, {onerror}, {followlinks}, {datatype}, {type})**

Parameter	Explanation
top	The top-level workspace that will be used.



File Edit Search View Project Run Tools Help

Find: 

Project Explorer

Untitled  
Files  
Run Configurations

```
10 |
11 import arcpy.mapping as am
12 import arcpy
13 arcpy.env.workspace = "D:\\temp\\mxds to fix"
14 MXDFiles = arcpy.ListFiles("*.mxd")
15 for MXDFile in MXDFiles:
16     mxd = am.MapDocument(MXDFile)
17     print("")
18     print("----- Processing " + MXDFile + " -----")
19     for elm in arcpy.mapping.ListLayoutElements(mxd):
20         if elm.type == "DATAFRAME_ELEMENT":
21             print("Dataframe: " + elm.name)
22             for lay in am.ListLayers(mxd, "", elm):
23                 if lay.supports("WORKSPACEPATH"):
```

File Expl... Project ... demosetup.py x buff05.py x analysis.py x management.py x Fixall.py x \_mapping.py x

Python Interpreter

Path: D:\temp\testing\Database\San\_Juan.gdb

Layer: Hillshade

Path: D:\temp\Database\Elevation

Path: D:\temp\testing\Database\Elevation\

Done

&gt;&gt;&gt;

Variables Messages Python Interpreter

**Add these lines get each .mxd file**

10: 1

Modified

Insert



File Edit Search View Project Run Tools Help

Find:

Project Explorer

Untitled  
Files  
Run Configurations

Indent your original  
script inside  
the for loop

```
• 13 arcpy.env.workspace = "D:\\temp\\mxds to fix"
• 14 MXDFiles = arcpy.ListFiles("*.mxd")
• 15 for MXDFile in MXDFiles:
• 16     mxd = am.MapDocument(MXDFile)
• 17     print("")
• 18     print("----- Processing " + MXDFile + " -----")
• 19     for elm in arcpy.mapping.ListLayoutElements(mxd):
        if elm.type == "DATAFRAME_ELEMENT":
            print("Dataframe: " + elm.name)
            for lay in am.ListLayers(mxd, "", elm):
                if lay.supports("WORKSPACEPATH"):
                    print(" Layer: " + lay.name)
                    print(" Path: " + lay.workspacePath)
                    oldpath = lay.workspacePath
                    newpath = oldpath.replace("D:\\temp", "D:\\temp\\testing")
                    lay.findAndReplaceWorkspacePath (oldpath, newpath)
                    print(" Path: " + lay.workspacePath)
                    print("")
• 27     mxd.save()
• 28
• 29
• 30
• 31
• 32 print("Done")
```

File Expl... Project ... demosetup.py x buff05.py x analysis.py x management.py x Fixall.py x \_mapping.py x

Python Interpreter

18: 54

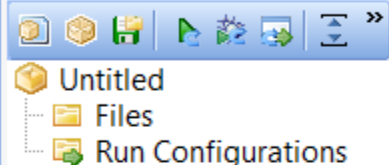
Modified

Insert

File Edit Search View Project Run Tools Help

Find: 

Project Explorer



And save each mxd  
as it is processed

```
• 13 arcpy.env.workspace = "D:\\temp\\mxds to fix"
• 14 MXDFiles = arcpy.ListFiles("*.mxd")
• 15 for MXDFile in MXDFiles:
• 16     mxd = am.MapDocument(MXDFile)
• 17     print("")
• 18     print("----- Processing " + MXDFile + " -----")
• 19     for elm in arcpy.mapping.ListLayoutElements(mxd):
• 20         if elm.type == "DATAFRAME_ELEMENT":
            print ("Dataframe: " + elm.name)
            for lay in am.ListLayers(mxd, "", elm):
                if lay.supports("WORKSPACEPATH"):
                    print(" Layer: " + lay.name)
                    print(" Path: " + lay.workspacePath)
                    oldpath = lay.workspacePath
                    newpath = oldpath.replace("D:\\temp", "D:\\temp\\testing")
                    lay.findAndReplaceWorkspacePath (oldpath, newpath)
                    print(" Path: " + lay.workspacePath)
                    print("")
• 28
• 29
• 30
• 31     mxd.save()
• 32 print("Done")
```

File Expl... Project ... demosetup.py x buff05.py x analysis.py x management.py x Fixall.py x \_mapping.py x

Python Interpreter

18: 54

Modified

Insert

# Review

- Use “CURRENT” as the MapDocument to work live in ArcMap
  - Test live in ArcMap and save to a script file
- Use the MXD file name as the MapDocument to work on MXD files
- Read and sometimes write document properties (Title, dates, etc.)
- Work with elements in a layout
- Work with map layers
- Set data sources on many MXD files at once using ListFiles or da.Walk

# ArcPy.Mapping in Pro

Similar functionality, but Pro is different

- Python 3.5 instead of 2.7
- `arcpy.mp` replaces `arcpy.mapping`
- `.aprx` file replaces `.mxd`
- Multiple layouts
- `Map`, `MapFrame`, and `Camera` objects replace the `Data Frame`

<http://tinyurl.com/promapping>

► Get started

► Geoprocessing and Python

► ArcPy functions

► ArcPy classes

► Data Access module

▼ Mapping module

[Introduction to arcpy.mp](#)

[Migrating from arcpy.mapping to ArcGIS Pro](#)

[Guidelines for arcpy.mp](#)

[Tutorial: Getting started with arcpy.mp](#)

[Alphabetical list of arcpy.mp classes](#)

## Migrating from arcpy.mapping to ArcGIS Pro

- Python 3
- Arcpy.mapping is now arcpy.mp
- The ArcGIS Pro project file (.aprx)
- Many of the list functions have moved
- Export functions have moved
- Layer management functions have moved
- Layer files have changed
- New Map, MapFrame, and Camera objects replace the role of the data frame
- A new Layout object
- The application always refreshes when using CURRENT
- Updating data sources has changed

**Arccpy.mapping** scripts authored with ArcGIS Desktop will need to be modified before they will run in ArcGIS Pro. The changes are straight-forward and logical and can be accomplished mostly with find and replace operations. The sections below highlight many of the significant changes to the **arccpy.mp** API as well as new features that were added.





esri

THE  
SCIENCE  
OF  
WHERE