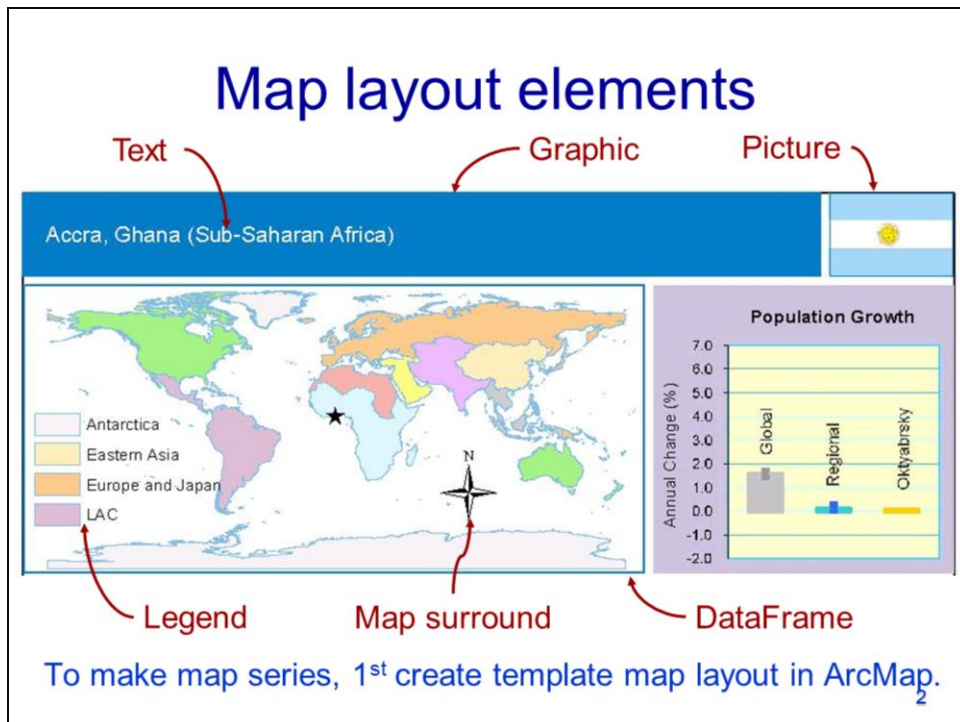


Working with the Map Layout

Automating map production

This video will discuss how to work with the map layout for the purpose of automating map production.

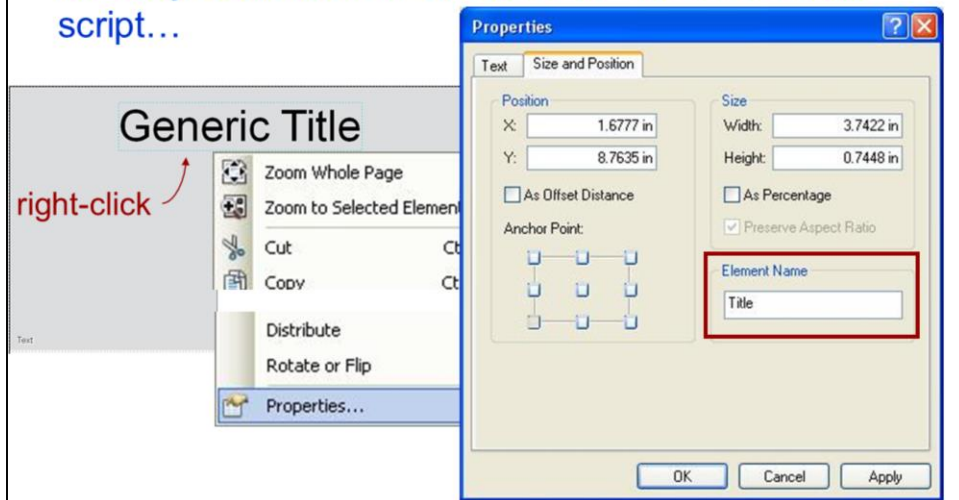


The map layout is where maps are created in ArcMap. Arcpy's mapping module provides access to the map layout to allow for automating map production. Arcpy can only modify existing map elements and so must use an existing map layout as a template.

There are 6 types of map elements that arcpy can access. These elements include 1) text, 2) graphics, 3) pictures, 4) legend, 5) map surround, and 6) data frame. Each of these map elements will be discussed in detail later in this video.

Naming map elements

- In the Map Layout template, assign names to allow for easy identification of elements from within the script...



Map elements should be assigned clear names in ArcMap so that they can be easily identified in the script.

The **element name** can be accessed, in ArcMap, through the element's properties page.

Getting a list of map elements

- Get list of map elements...

```
arcpy.mapping.ListLayoutElements(mxd, type, wildcard)
```

element type (optional)

map document object

element name
(* for wildcard)

```
TEXT_ELEMENT  
DATAFRAME_ELEMENT  
GRAPHIC_ELEMENT  
LEGEND_ELEMENT  
MAPSURROUND_ELEMENT  
PICTURE_ELEMENT  
NONE
```

element types

- Example: get list of text element(s) named "Title"...

```
arcpy.mapping.ListLayoutElements(mxd, "TEXT_ELEMENT", "Title")
```

[<TextElement object...>]

4

The mapping module's **ListLayoutElements** method can be used to get a list of the map elements in the map document. The element type and wildcard can be used to restrict the elements that are returned.

The available element types are listed here.

This example statement will get a list containing text elements that have the word "Title" in their element name (note that in the previous slide, we learned how to set this element name from within ArcMap).

Getting a map element

- `ListLayoutElements` creates a list.

```
>>> arcpy.mapping.ListLayoutElements(mxd)
[<MapSurroundElement object...>, <TextElement object...>,
 <DataFrame object...>, <DataFrame object...>]
```

- Use list indexing to retrieve individual element(s)....
element = elementLst[0]

- Use for loop to iterate through elements in list...

```
for element in elementLst:
    print element.name
```

5

The **ListLayoutElements** method creates a list of map element objects.

The list works like a Python list when retrieving items.

The for loop can be used with the element list.

Element properties

- Element name (read/edit)...

```
>>> element.name = "Title"
```

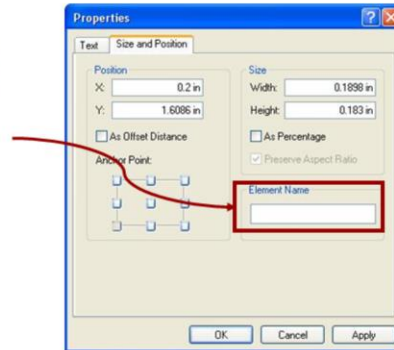
```
>>> element.name
```

```
u'Title'
```

- Element type (read)...

```
>>> element.type
```

```
u'TEXT_ELEMENT'
```



6

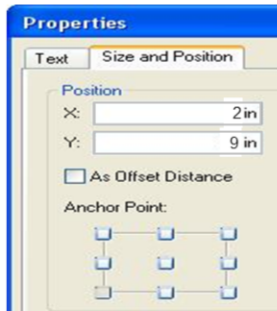
The next few slides will discuss some general properties that apply to all map element types.

The **name** property can be read or modified..

The **type** property can be read.

Element position (read/edit)

```
elem = elementLst[0]
```



```
>>> elem.elementPositionX
```

```
3
```

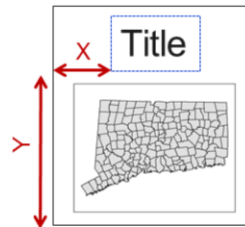
```
>>> elem.elementPositionY
```

```
10
```

```
>>> elem.elementPositionX = 2
```

```
>>> elem.elementPositionY = 9
```

- coordinates are of anchor point
- measured from lower left of page – this can be changed from within ArcMap.



7

The position of the element can be read and modified.

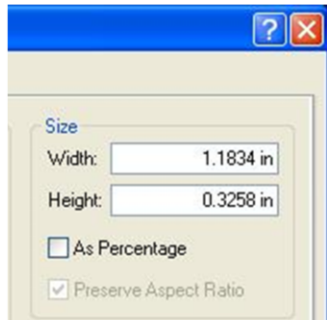
The **elementPositionX** and **elementPositionY** properties read the coordinates of the anchor point.

The same properties can also set the coordinates of the anchor point.

The anchor point position can only be specified in ArcMap. In this example, the anchor point corresponds to the lower left corner of the element.

The position coordinates are referenced to the lower left corner of the map page. The coordinate units are in the units of the map layout (e.g. inches, cm, or points).

Element size (read/edit)

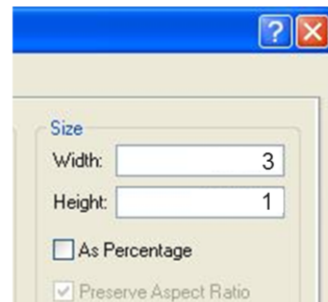


```
>>> elem.elementWidth  
1.1831
```

```
>>> elem.elementHeight  
0.3258
```

```
>>> elem.elementWidth = 3
```

```
>>> elem.elementHeight = 1
```



8

The **elementWidth** and **elementHeight** properties allow the element sizes to be read or modified. The element size are specified in the map layout units.

The data frame object

- Provides access to map coordinate system, scale, and map extent.
- To get a list of data frames...

```
dataFrames = arcpy.mapping.ListLayoutElements(mxd,  
                                              "DATAFRAME_ELEMENT")
```

```
>>> dF = dataFrames[0]
```

Read/modify scale...

```
>>> dF.scale = 24000  
>>> dF.scale  
24000
```

Read/modify spatial ref...

```
>>> dF.spatialReference = spatRef  
>>> dF.spatialReference  
<SpatialReference object...>
```

9

The data frame object provides access to the map coordinate system, scale, and map extent.

A list of the data frames in a map document can be obtained using the **ListLayoutElements** method and specifying the "DATAFRAME_ELEMENT" type.

The **scale** property of the data frame allows the map scale to be read or modified.

The **spatialReference** property allows the data frames coordinate system to be read or modified.

Data frame extent

- Data frame extent (read/edit)...

```
dF.extent = newExtent
```

```
>>> dF.extent
```

```
<Extent object ...>
```



- Center map on extent without changing scale...

```
dF.panToExtent(newExtent)
```



- Zoom to all selected features...

```
dF.zoomToSelectedFeatures()
```



The **extent** property allows the data frame extent to be read or modified. The property returns an extent object when read and requires an extent object when modified.

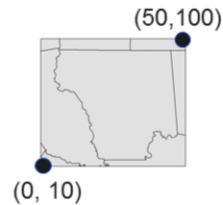
The **panToExtent** method will pan the data frame view to the coordinates specified by an extent object. Panning centers the view on the extent object but does not change the scale.

The **zoomToSelectedFeatures** method zooms the view to any selected features in the data frame's layers.

Extent object

- Stores min and max coordinates of a feature, feature class, or dataframe...

```
>>> extent.Xmin    >>> extent.Ymin
0                  10
>>> extent.Xmax    >>> extent.Ymax
50                 100
```



- To create an extent object...

```
newExtent = arcpy.Extent(Xmin, Ymin, Xmax, Ymax)
```

- Extent object can be retrieved from a geometry object, the describe tool, a layer object, a dataframe object...

11

The extent object stores the minimum and maximum X and Y coordinates of a feature, feature class, or a data frame. The coordinates are accessed from the extent object's properties.

An extent object can be created using arcpy's **Extent** method. The coordinates for the extent are specified as integer or decimal numbers.

The extent object can be retrieved using a number of methods.

Example: setting dataframe extent

```
mxd = arcpy.mapping.MapDocument(mxdFile)
dF = arcpy.mapping.ListDataFrames(mxd)[0]
lyr = arcpy.mapping.ListLayers(mxd, "Town")[0]
lyrExt = lyr.getExtent()
dF.extent = lyrExt
```

12

This slide will show an example of script that sets the extent of a data frame.

The first statement gets the map document object.

The ListDataFrames method gets a list of the data frames in the map document.

The ListLayers method gets the layer named "Town" from the map document.

The layer object's **getExtent** method gets the extent of the entire layer.

The data frame's extent property is set to the extent object that was obtained from the layer object.

Text elements

- Get list of text element(s)...

```
txtElems = arcpy.mapping.ListLayoutElements(  
    mxd, "TEXT_ELEMENT", "Map Title")
```

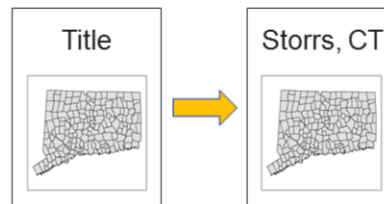
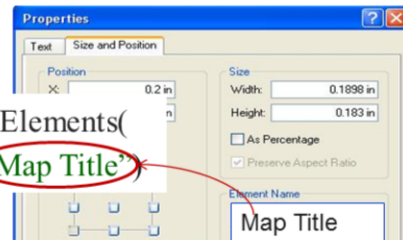
- Get a text element...

```
titleElem = txtElems[0]
```

- Read/modify text...

```
>>> titleElem.text  
u'Title'
```

```
>>> titleElem.text = "Storrs, CT"
```



13

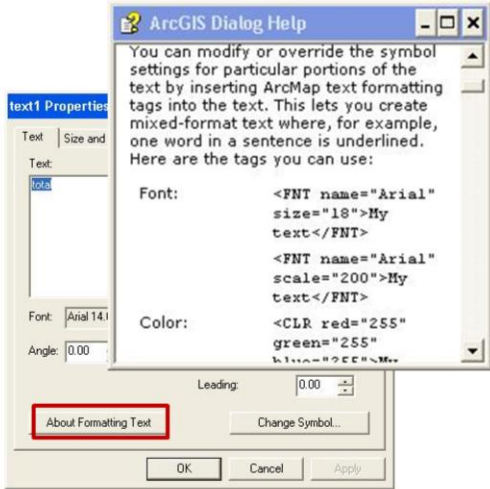
Text elements include titles and text boxes in the map layout.

A list of text elements can be retrieved using the **ListLayoutElements** method and specifying the "TEXT_ELEMENT" type.

The element must be retrieved from the list.

The **text** property can be read or modified.

Text elements: formatting fonts



The image shows a screenshot of the ArcGIS software interface. On the left, a 'text1 Properties' dialog box is open, showing a 'Text' field with the word 'note' and a font set to 'Arial 14.1'. A red box highlights the 'About Formatting Text' button. On the right, an 'ArcGIS Dialog Help' window is open, displaying text about formatting tags. The help text includes examples for font, color, and scale tags.

- ArcMap formatting tags can be applied to text.
- Refer to About Formatting Text help page.
- Formatting can be applied to all or part of a text string.
- To use in Python, set text property equal to tag.

The font, color, and style in a text element can be specified by using formatting tags in the text string.

Refer to the “About Formatting Text” documentation page for information.

Formatting can be applied to part or all of a text string.

Applying font format

1. Copy/paste tag from help page into script.

```
<CLR red= "255" green= "255" blue= "255">My text</CLR>
```

2. Change any quotes (if present) to single quotes; replace My text with desired text...

```
<CLR red=02550green=02550blue=0255010</CLR>
```

3. Enclose entire expression in double quotes and assign to element's text property...

```
txtElem.text = "<CLR red='255' green='255' blue='255'>10</CLR>"
```

4. For font color tags, modify color values (0-255)

```
txtElem.text = "<CLR red='255' green='0' blue='0'>10</CLR>"
```

15

This slide will show an example of how to format the font color in a text element.

The appropriate tag can be copied directly from the help page. In this example, the tag will set the font color.

Any double quotes in the tag should be changed to single quotes.

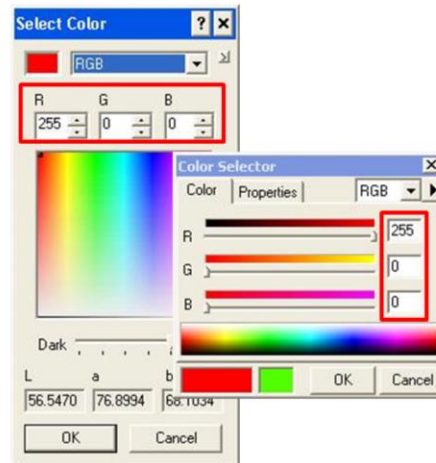
The "My text" should be replaced with the desired text.

The entire tag expression should be enclosed in double quotes and set equal to the element's text property.

In this case, the numbers corresponding to red, green, and blue can be modified to create the desired font color.

Font colors

- Look up RGB (red, green, blue) values for desired color.
- Modify values in format tag...



- Red text... **Mansfield**

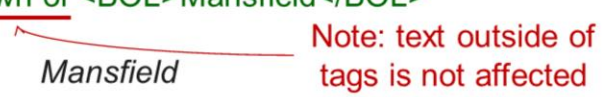
```
txtElem.text = "<CLR red='255' green='0' blue='0'>Mansfield</CLR>"
```

16

You can use the color palate to help determine the red, green, blue color combination needed to create the desired font color. The color palate can be accessed, in ArcMap, through the layer symbology page. Select the desired color from the palate and record the RGB numbers.

Update the tag line with the desired color combination. This example creates a red font.

Font format tags

- **Bold text...** Town of **Mansfield**
txtElem.text = “Town of <BOL>Mansfield</BOL>”

- **Italics text...** *Mansfield*
txtElem.text = “<ITA>Mansfield</ITA>”
- **Underlined text...** Mansfield
txtElem.text = “<UND>Mansfield</UND>”
- **Apply multiple formats (bold and underlined)...**
txtElem.text = “Town of <BOL><UND>Mansfield</UND></BOL>”
Town of **Mansfield**

17

This slide will show examples of formatting the font styles in a text element.

The tag shown here will make the “Mansfield” text bold. The “Town of” text is outside the tag and so will not be bold.

The tag here will make the text italics.

This tag will make the text underlined

Multiple tags can be applied by putting a tag within a tag. This example will make “Mansfield” bold and underlined.

Managing text element position

- Text elements can change position when text is modified...

Total sites: **total number of sites**

Total sites: **10**

- To anchor element to original position...

- get x position before modifying text

`x = elem.elementPositionX`

Total sites: **total numt**
x

- modify text...

`elem.text = 10`

Total sites: **10**
x

- reset to original x position after modifying text...

`elem.elementPositionX = x`

Total sites: **10**
x

18

Text elements can move when the text is changed.

To keep the element in its original position:

- 1) Get the element's X position before changing the text.
- 2) Then modify the text.
- 3) Finally, reset the element's X position back to the original location.

Legend elements

- List of legend elements...

```
legends = arcpy.mapping.ListLayoutElements(mxd,  
      "LEGEND_ELEMENT")
```

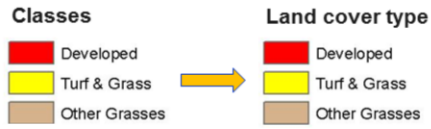
- Get a legend element...

```
legend = legends[0]
```

- Read/modify title...

```
>>> legend.title  
'Classes'
```

```
>>> legend.title = "Land cover type"
```



19

The legend element can be retrieved with the **ListLayoutElements** method.

The **title** property allows the legend title to be read or modified.

Legend elements continued

- Read name of associated dataframe...

```
>>> legend.parentDataFrameName  
u'Frame 1'
```

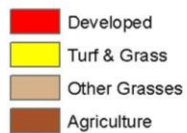


- Change column count...

```
legend.adjustColumnCount(2)
```

column count

Land cover type



Land cover types



20

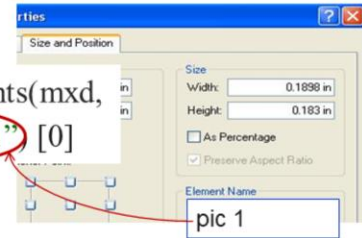
The legend object's **parentDataFrameName** returns the name of the data frame that is associated with the legend.

The **adjustColumnCount** allows the number of columns in the legend to be changed.

Picture elements

- ...Images inserted into a map layout
- Get a picture element...

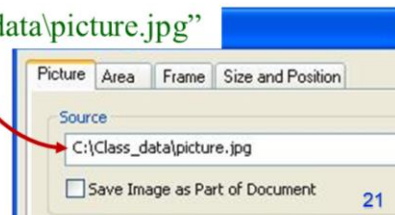
```
pic1 = arcpy.mapping.ListLayoutElements(mxd, in  
"PICTURE_ELEMENT", "pic 1" [0]
```



- Read/modify source image...

```
picture.sourceImage = r"C:\Class_data\picture.jpg"
```

```
>>> picture.sourceImage  
u'C:\Class_data\picture.jpg'
```



Picture elements in the map document can be retrieved using the **ListLayoutElements** method.

The image associated with the picture element can be read or modified through the element's **sourceImage** property.

Map surround elements

- Elements associated with a data frame...
 - north arrow, scale bar, scale text
- Arcpy limited to modifying size and position.
- Get a map surround element

```
northArrow = arcpy.mapping.ListLayoutElements(mxd,  
        "MAPSURROUND_ELEMENT", "northArrow") [0]
```
- Read associated data frame...

```
>>> northArrow.parentDataFrameName  
u'Frame 1'
```



22

Map surround elements include the north arrow, scale bar, and scale text.

For these elements, arcpy can only modify the size or positions.

The **ListLayoutElements** method can be used to get a list of these elements.

The **parentDataFrameName** property will return the name of the data frame associated with the element.

Exporting a map (to pdf)

- Data frame or map layout can be exported to several file formats.
- To export as pdf...

```
arcpy.mapping.ExportToPDF (map object, r"C:\maps\outputMap.pdf",  
    "PAGE_LAYOUT", resolution = 300, image_quality = "BEST")
```

Diagram illustrating the parameters of the `arcpy.mapping.ExportToPDF` method:

- `map object` (red text) points to the first parameter (mxd).
- `output map` (red text) points to the second parameter (r"C:\maps\outputMap.pdf").
- `data frame object or "PAGE_LAYOUT"` (red text) points to the third parameter ("PAGE_LAYOUT").
- `resolution` (red text) points to the fourth parameter (resolution = 300).
- `image quality` (red text) points to the fifth parameter (image_quality = "BEST").

- See ArcGIS help for more info on parameters

23

The data frame or the map layout can be exported as a .pdf, .jpg, or other format

The mapping module's **ExportToPDF** method will create a map in a .pdf format. The "PAGE LAYOUT" option will export the **map layout view**. A data frame object may also be specified to export the **Data View**.

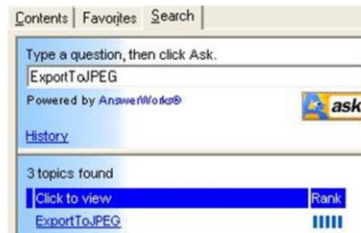
The resolution can be specified in dots/inch and the image quality can be specified.

See ArcGIS's documentation for further information on parameters for the ExportToPDF method.

Further info an exporting maps

- See ArcGIS Help for information on exporting to other formats...
 - Search by function...

-ExportToAI -ExportToJPEG
-ExportToBMP -ExportToPDF
-ExprotToEMF -ExportToPNG
-ExportToEPS -ExportToSVG
-ExportToGIF -ExportToTIFF



24

Arcpy has methods to export maps in a variety of different formats – search for these tools in the ArcGIS documentation for further info.