python ArcGIS

Template automation

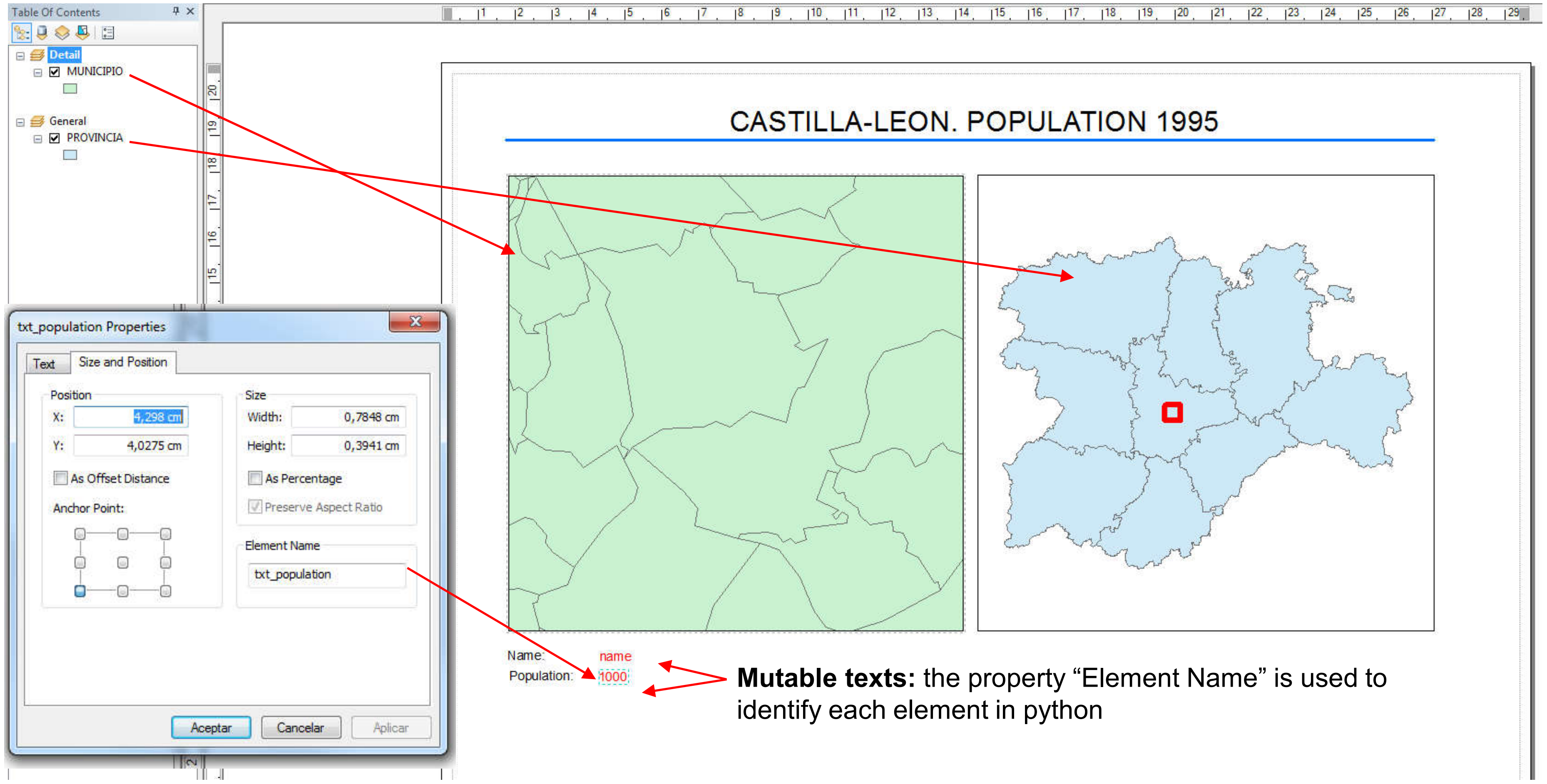
Access to template graphics

# A template (layout) has different types of graphic elements, like texts, images, lines, data frames, etc., with several properties that can be accessed using python.

# We can make map automation easier if we design a template very similar to the final result that we want to obtain (including symbology). That way, we will have "fixed" elements and "mutable" elements. The former never will change and the lasts can be changed using programming in runtime.

# Next example shows an mxd project with two data frames (general and detailed view). There are several fixed elements (two data frames, a title, a line and two texts). The only mutable elements are two texts that display some information linked to the layers.

Access to template graphics



txt_population Properties

Text | Size and Position

Position
X: 4,298 cm
Y: 4,0275 cm
☐ As Offset Distance
Anchor Point:

Size
Width: 0,7848 cm
Height: 0,3941 cm
☐ As Percentage
☑ Preserve Aspect Ratio

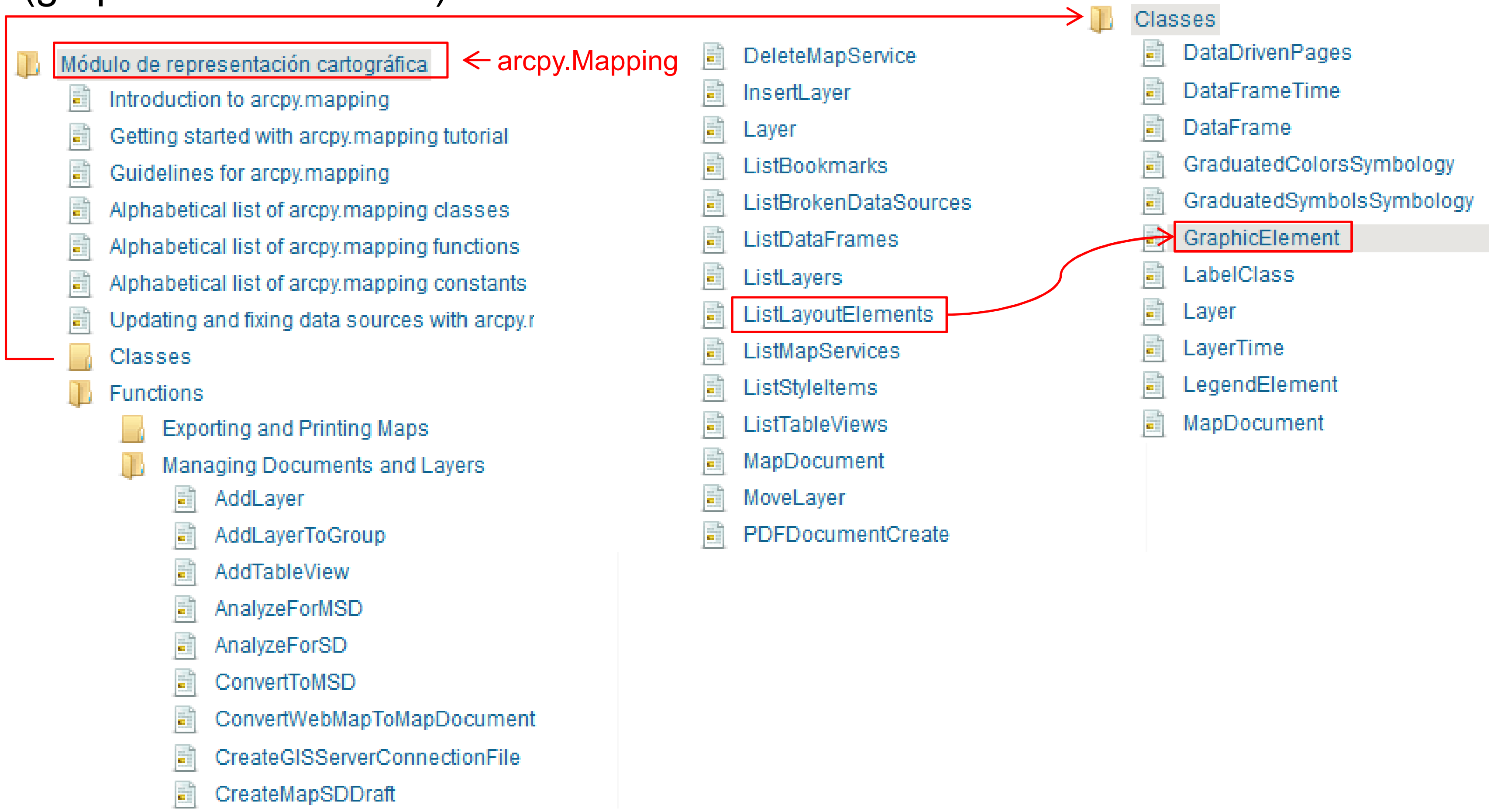
Element Name
txt_population

Aceptar | Cancelar | Aplicar

Mutable texts: the property “Element Name” is used to identify each element in python

Access to template graphics

# The module arcpy.mapping provides the function ListLayoutelements. This function allows access to every single property of each graphic element (graphicElement class).



Módulo de representación cartográfica ← arcpy.Mapping
- Introduction to arcpy.mapping
- Getting started with arcpy.mapping tutorial
- Guidelines for arcpy.mapping
- Alphabetical list of arcpy.mapping classes
- Alphabetical list of arcpy.mapping functions
- Alphabetical list of arcpy.mapping constants
- Updating and fixing data sources with arcpy.r
- Classes
- Functions
  - Exporting and Printing Maps
  - Managing Documents and Layers
    - AddLayer
    - AddLayerToGroup
    - AddTableView
    - AnalyzeForMSD
    - AnalyzeForSD
    - ConvertToMSD
    - ConvertWebMapToMapDocument
    - CreateGISServerConnectionFile
    - CreateMapSDDraft
    - DeleteMapService
    - InsertLayer
    - Layer
    - ListBookmarks
    - ListBrokenDataSources
    - ListDataFrames
    - ListLayers
    - ListLayoutElements
    - ListMapServices
    - ListStyleItems
    - ListTableViews
    - MapDocument
    - MoveLayer
    - PDFDocumentCreate

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

Access to template graphics

# The module arcpy.mapping provides the function ListLayoutelements. This function allows access to every single property of each graphic element (graphicElement class).

http://resources.arcgis.com/en/help/main/10.1/index.html#/GraphicElement/00s300000040000000/

http://resources.arcgis.com/en/help/main/10.1/index.html#/na/00s30000003w000000/

```python
#importing modules
import arcpy
import arcpy.mapping as map

mxd = map.MapDocument(r'E:\asignaturas\layout.mxd')
#listing all graphics elements within a layout
for graphic in map.ListLayoutElements(mxd):
    #get the graphic element type
    print(graphic.type)
```

9_access_template_elements.py

Access to template graphics

# We can apply any type of filter using the parameter "element type" of this class. In this example we only access to text elements.

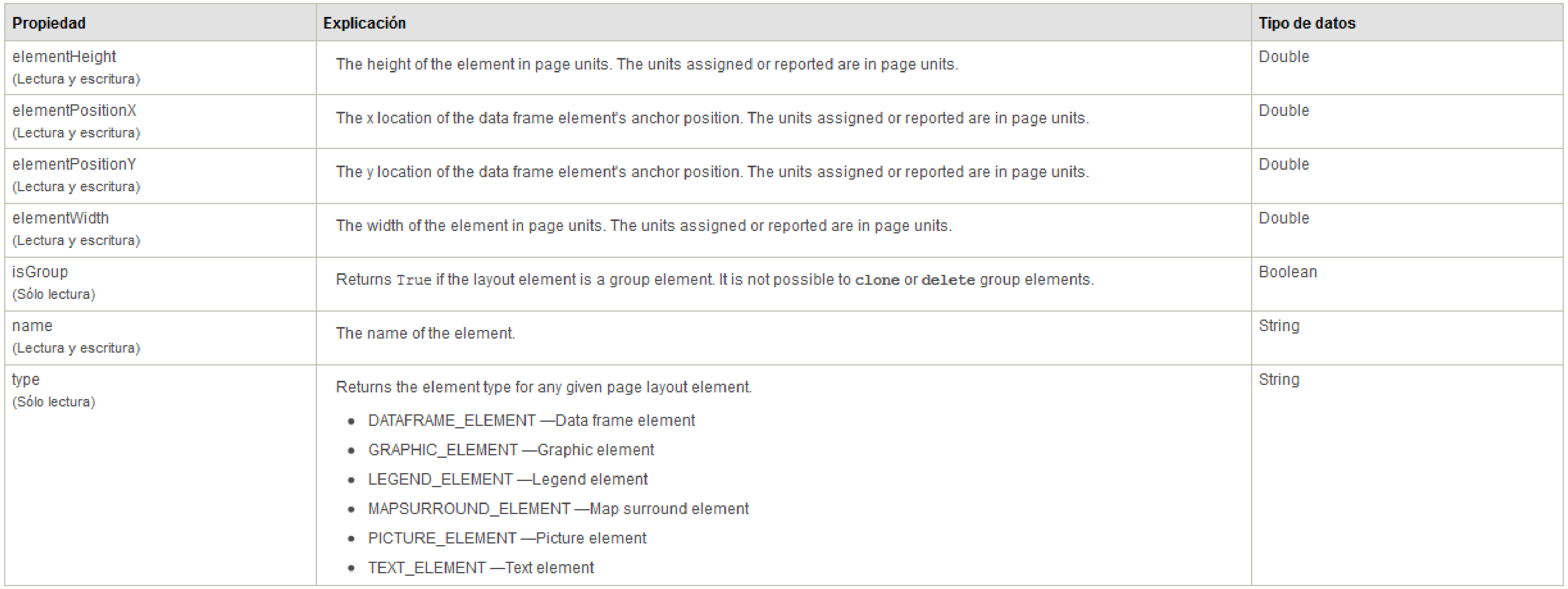
Arcpy.mapping. ListLayoutElements (map_document, {element_type}, {wildcard})

```
#importing modules
import arcpy
import arcpy.mapping as map

mxd = map.MapDocument(r'I:\asignaturas\layout.mxd')
#filtering all graphic elements by type
for graphic in map.ListLayoutElements(mxd,'TEXT_ELEMENT'):
    #get a graphic object using its name
    if graphic.name == 'text_name':
        print(grafico.text)
    if graphic.name == 'text_population':
        print (graphic.text)
```

10_access_template_texts.py

Access to template graphics

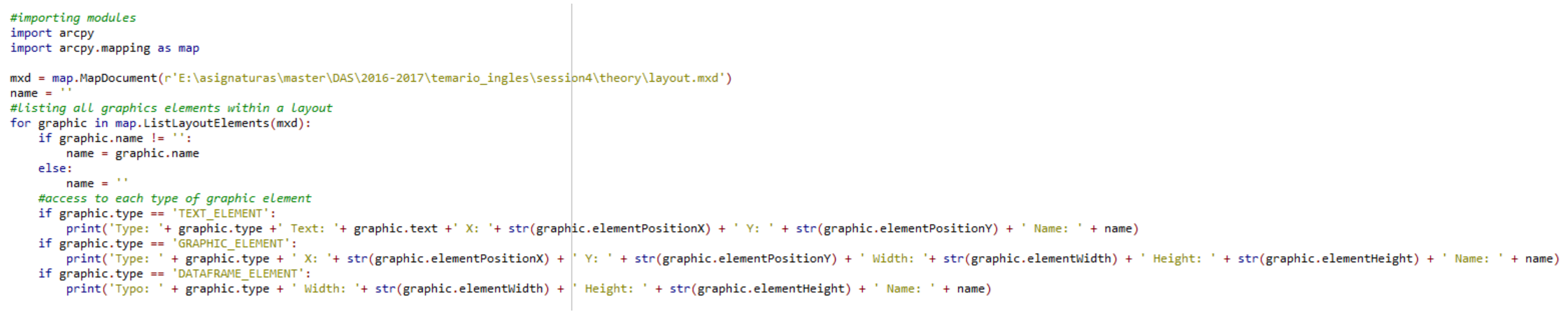
# Each type of graphic element has several properties, some of them common and some of them specific.



Propiedad	Explicación	Tipo de datos
elementHeight (Lectura y escritura)	The height of the element in page units. The units assigned or reported are in page units.	Double
elementPositionX (Lectura y escritura)	The x location of the data frame element's anchor position. The units assigned or reported are in page units.	Double
elementPositionY (Lectura y escritura)	The y location of the data frame element's anchor position. The units assigned or reported are in page units.	Double
elementWidth (Lectura y escritura)	The width of the element in page units. The units assigned or reported are in page units.	Double
isGroup (Sólo lectura)	Returns `True` if the layout element is a group element. It is not possible to **`clone`** or **`delete`** group elements.	Boolean
name (Lectura y escritura)	The name of the element.	String
type (Sólo lectura)	Returns the element type for any given page layout element. • DATAFRAME_ELEMENT —Data frame element • GRAPHIC_ELEMENT —Graphic element • LEGEND_ELEMENT —Legend element • MAPSURROUND_ELEMENT —Map surround element • PICTURE_ELEMENT —Picture element • TEXT_ELEMENT —Text element	String

Access to template graphics

# Each type of graphic element has several properties, some of them common and some of them specific.



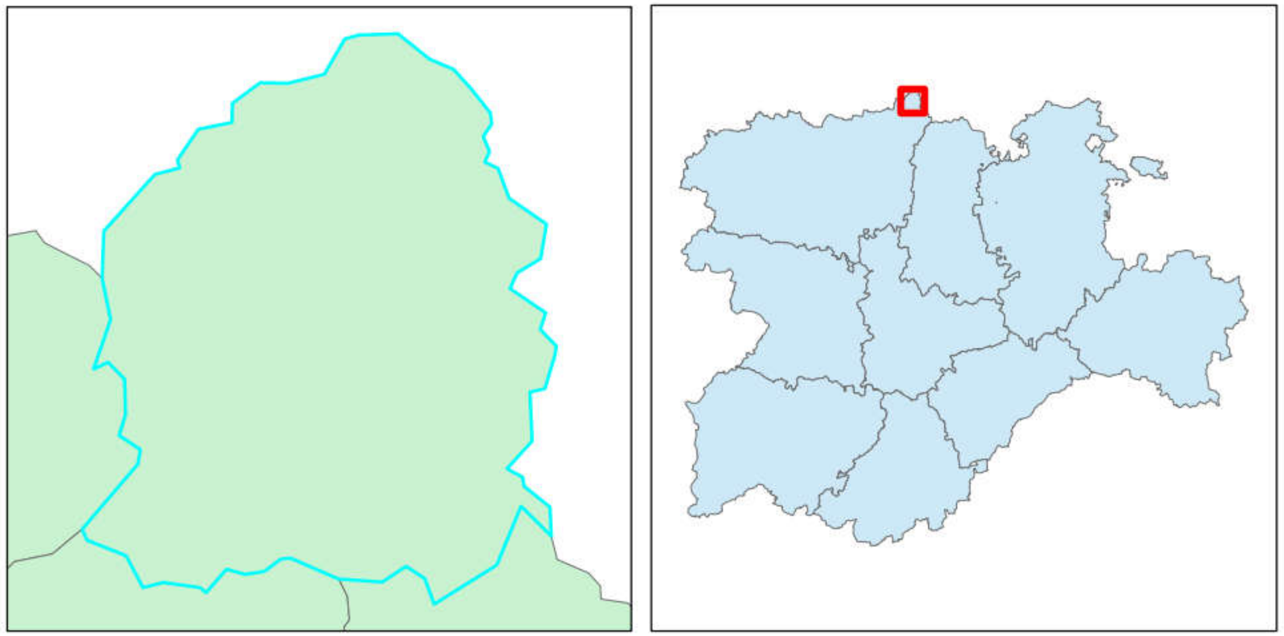
```python
#importing modules
import arcpy
import arcpy.mapping as map

mxd = map.MapDocument(r'E:\asignaturas\master\DAS\2016-2017\temario_ingles\session4\theory\layout.mxd')
name = ''
#listing all graphics elements within a layout
for graphic in map.ListLayoutElements(mxd):
    if graphic.name != '':
        name = graphic.name
    else:
        name = ''
    #access to each type of graphic element
    if graphic.type == 'TEXT_ELEMENT':
        print('Type: '+ graphic.type +' Text: '+ graphic.text +' X: '+ str(graphic.elementPositionX) + ' Y: ' + str(graphic.elementPositionY) + ' Name: ' + name)
    if graphic.type == 'GRAPHIC_ELEMENT':
        print('Type: ' + graphic.type + ' X: '+ str(graphic.elementPositionX) + ' Y: ' + str(graphic.elementPositionY) + ' Width: '+ str(graphic.elementWidth) + ' Height: ' + str(graphic.elementHeight) + ' Name: ' + name)
    if graphic.type == 'DATAFRAME_ELEMENT':
        print('Typo: ' + graphic.type + ' Width: '+ str(graphic.elementWidth) + ' Height: ' + str(graphic.elementHeight) + ' Name: ' + name)
```

11_access_graphic_elements_properties.py

Map automation

# In this case, we want to select an specific municipality using its name. Once selected, we will perform an automatic zoom to the selection. At the same time,each mutable text will be modified according to the information within the database. Finally, we will export the layout to a pdf file.

CASTILLA-LEON. POPULATION 1995



Name:	Posada de Valdeon
Population:	537

Map automation

```python
#importing modules
import arcpy
import arcpy.mapping as map

#mxd folder
mxd = map.MapDocument(r'E:\asignaturas\master\DAS\2016-2017\temario_ingles\session4\theory\layout.mxd')
#data frame access
df = map.ListDataFrames(mxd)[0]
#layer access (municipalities)
layer = map.ListLayers(mxd,'',df)[0]
#attribute query
query = '"NOMBRE" = \'Posada de Valdeon\'' #BE CAREFUL WITH THE SINTAX!!!!
arcpy.SelectLayerByAttribute_management(layer,'NEW_SELECTION',query)
#zoom to selected elements
df.zoomToSelectedFeatures()
#feature iteration (cursor)
cursor = arcpy.SearchCursor(layer,query)
name_mun = ''
population =0
for row in cursor:
    name_mun = row.getValue('NOMBRE')
    population = row.getValue('POB95')
#template processing
for graphic in map.ListLayoutElements(mxd):
    if graphic.name == 'txt_name':
        graphic.text = name_mun
    if graphic.name == 'txt_population':
        graphic.text = population
#export result to pdf file
map.ExportToPDF(mxd, r'E:\asignaturas\master\DAS\2016-2017\temario_ingles\session4\theory\map.pdf')
```

Modify both the code and the template to create a pdf file for every province where appear all field value for each record.

12_template_processing.py

Additional libraries

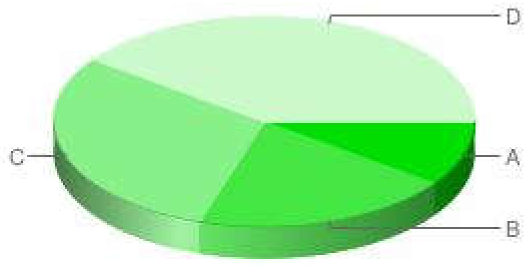
# The library "chartwarpper" allows to handle the Google Chart API from python.

# Link: https://code.google.com/p/google-chartwrapper/

# command line install: pip install GChartWrapper

# If we have the library PIL (pillow) previously installed, we will able to save the final chart to an image. That way, wew have the possibility to add this charts to our maps automatically.

```
from GChartWrapper import *

g = Pie3D([1,2,3,4])
g.label('A','B','C','D')
g.color('00dd00')
image = g.image
image().save('chart.jpg','JPEG')
```

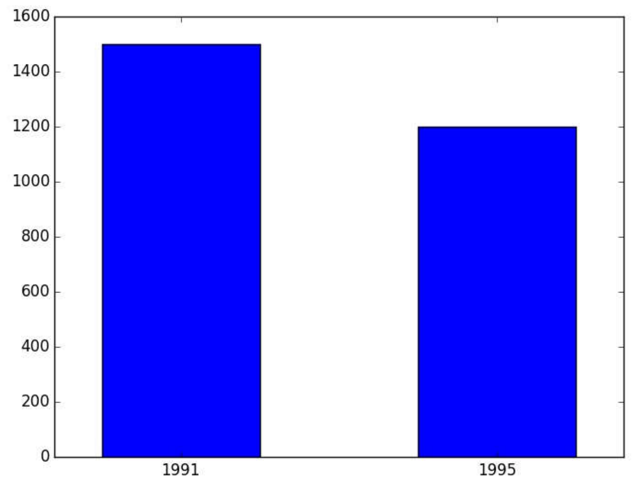


Additional libraries

# Matplotlib allows to make both simple and complex charts using python.

# Link: http://matplotlib.org/

```
import matplotlib.pyplot as plt
datos = [1500,1200]
etiquetas = [1,2]
# Example data
plt.bar(etiquetas,datos,width=0.5,align='center')
plt.xticks(etiquetas,['1991','1995'])
plt.savefig('grafica.jpg')
```


1600
1400
1200
1000
800
600
400
200
0
1991
1995