



Universidad **Ricardo Palma**

RECTORADO
PROGRAMA DE ESPECIALIZACIÓN EN CIENCIA DE DATOS

Formamos seres humanos para una cultura de paz

BIG DATA APLICADO

SESIÓN 03

Python

Expositores:

David Narváez

Eder Pineda

bigdataplicado@gmail.com





AGENDA - Python

1. Introducción al Python
2. Manejo del Editor Jupyter for Python
3. Tipos de objetos
4. Estructuras de control (if, else, while, for)
5. Manejo de funciones
6. Manejo de Librerías archivos, procesamiento y plotting
7. Machine Learning (Arboles de Decisión, Random Forest, XGBoosting)

1. INTRODUCCION AL Python

Características :

- Lenguaje interpretado.
- Lenguaje multipropósito.
- Lenguaje multiparadigma.

Guido Van Rossum



https://es.wikipedia.org/wiki/Guido_van_Rossum



1. INTRODUCCION AL Python

Características :

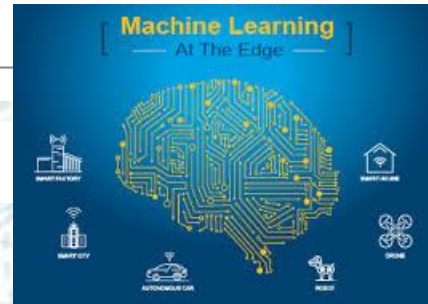
- Lenguaje interpretado.



1. INTRODUCCION AL Python

Características :

- Lenguaje multipropósito.



Desarrollo Web



Desarrollo de
Videojuegos



Desarrollo de GUIs
(interfaz gráfica de usuario)

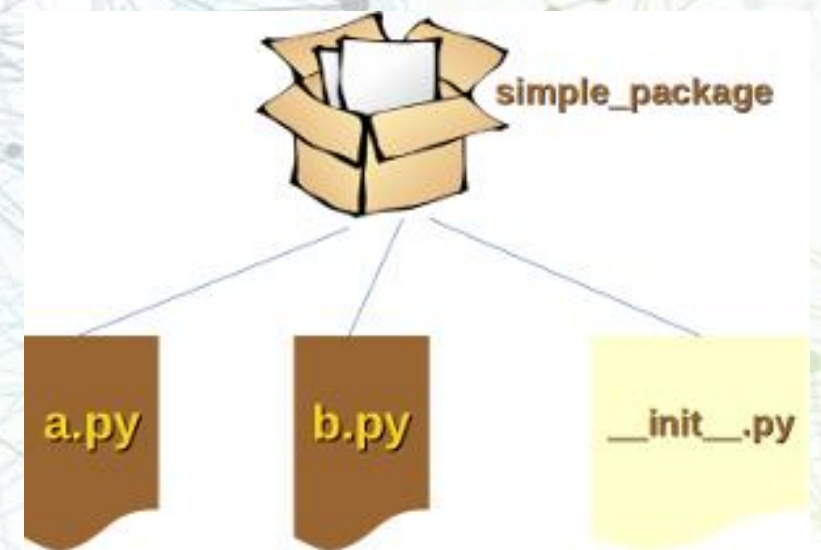
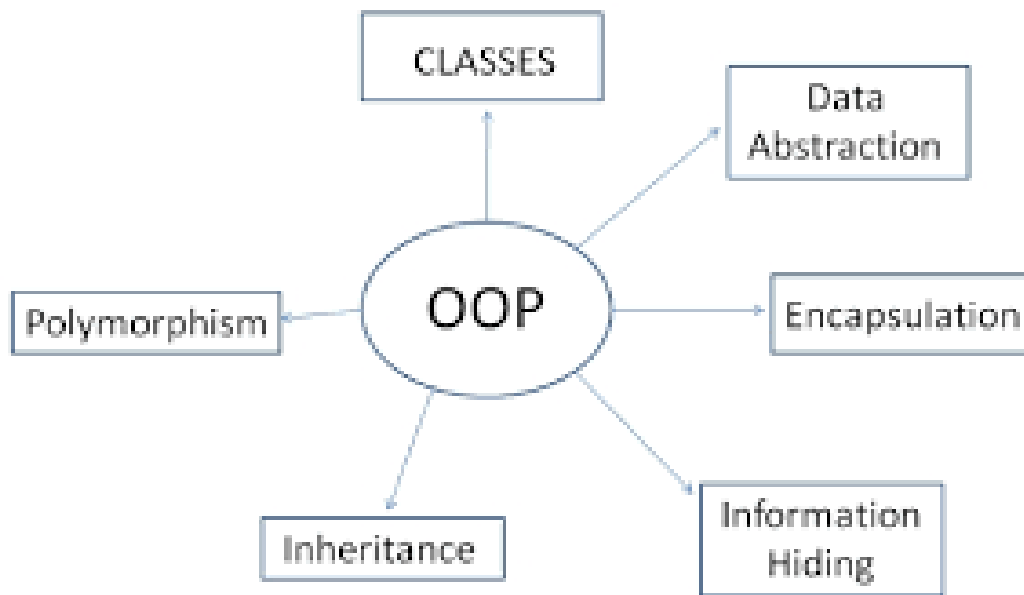


Desarrollo
de Software

1. INTRODUCCION AL Python

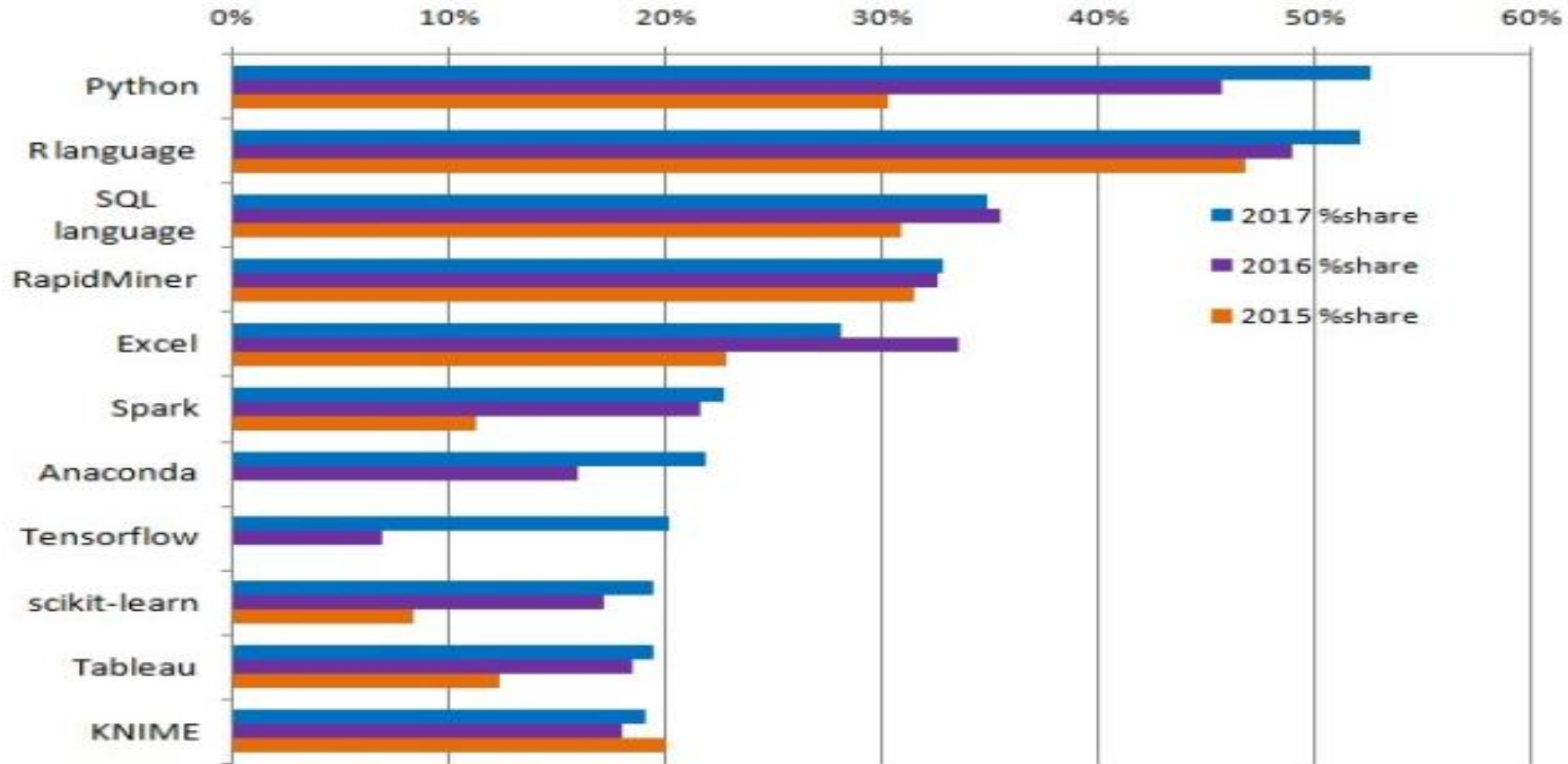
Características :

- Lenguaje multiparadigma.



1. INTRODUCCION AL Python

KDnuggets Analytics, Data Science, Machine Learning Software Poll, top tools share, 2015-2017



Recurso en la web :

- <https://www.python.org/>
- <https://www.continuum.io/downloads>
- <https://docs.python.org/3/tutorial/index.html>
- <http://www.python.org.ar/aprendiendo-python/>
- <http://www.safecreative.org/work/1207302042960-curso-python-para-principiantes>
- <http://docs.python.org.ar/tutorial/>
- <http://mundogeek.net/tutorial-python/>



2. MANEJO DEL EDITOR JUPYTER FOR PYTHON



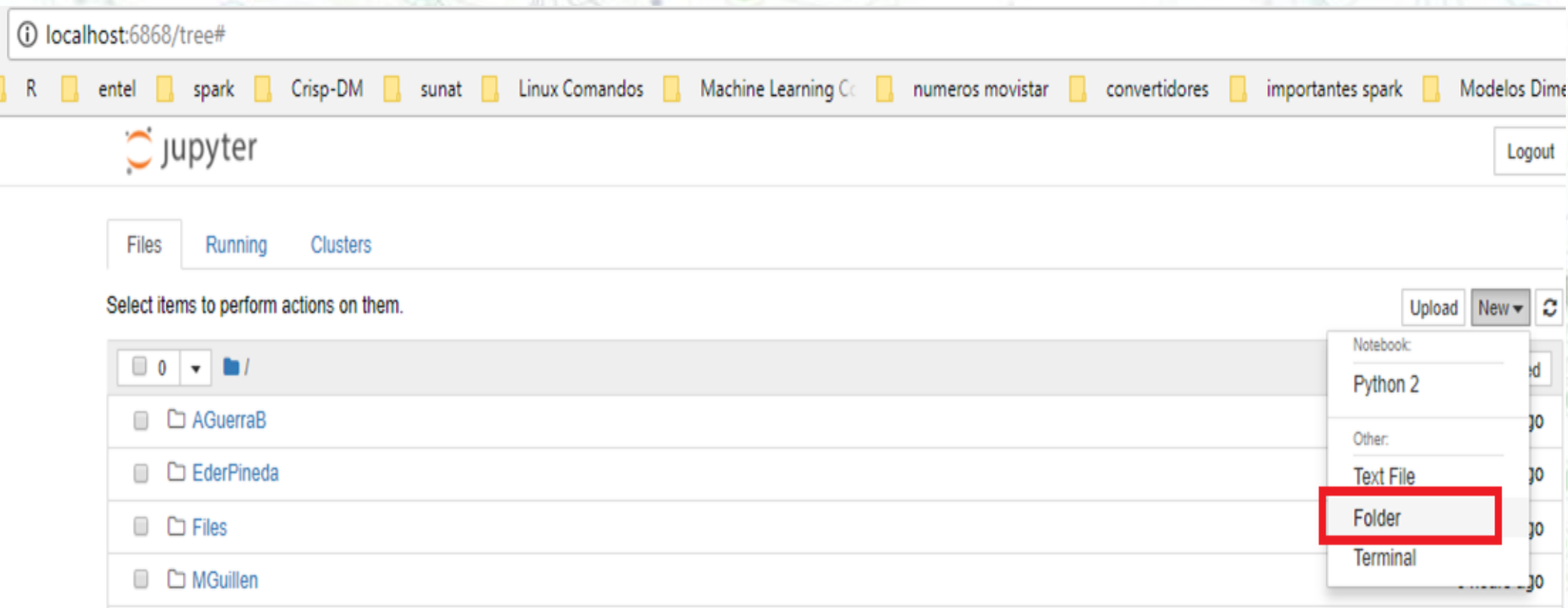
2. MANEJO DEL EDITOR JUPYTER FOR PYTHON

Jupyter es un **editor de texto** que contienen código fuente, ecuaciones, visualizaciones y texto explicativo.

Jupyter se puede utilizar en conjunto con Apache Spark para que podamos hacer análisis interactivo.

2. MANEJO DEL EDITOR JUPYTER FOR PYTHON

- Crear Carpeta



The screenshot displays the JupyterLab web interface in a browser window. The address bar shows 'localhost:6868/tree#'. A navigation bar at the top contains various icons and labels. Below this, the Jupyter logo is visible on the left and a 'Logout' button on the right. A tabbed interface shows 'Files', 'Running', and 'Clusters', with 'Files' being the active tab. A message 'Select items to perform actions on them.' is displayed above a file list. The file list shows a root directory with four subfolders: 'AGuerraB', 'EderPineda', 'Files', and 'MGuillen'. On the right side of the file list, there are buttons for 'Upload', 'New', and a refresh icon. The 'New' dropdown menu is open, showing options under 'Notebook:' (Python 2) and 'Other:'. The 'Folder' option under 'Other:' is highlighted with a red rectangle. The 'Terminal' option is also visible below 'Folder'.

2. MANEJO DEL EDITOR JUPYTER FOR PYTHON

Crear Carpeta

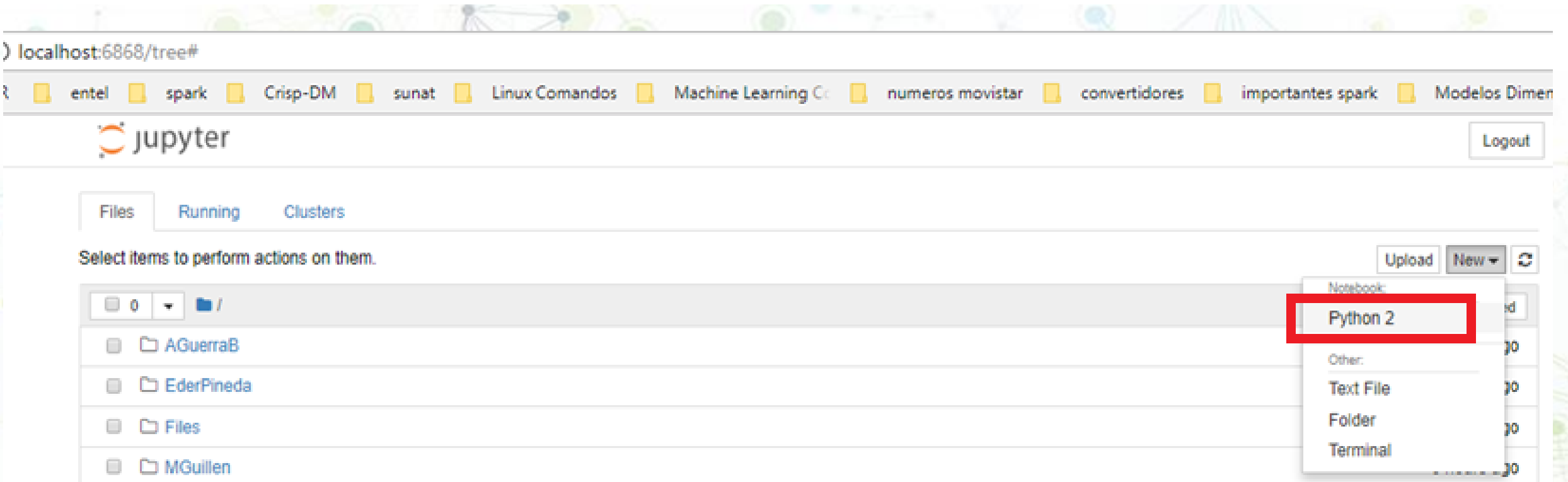


The screenshot displays the JupyterLab file browser interface. At the top, there are tabs for 'Files', 'Running', and 'Clusters'. Below these, there are buttons for 'Rename', 'Move', and a trash icon. On the right side, there are buttons for 'Upload', 'New', and a refresh icon. The main area shows a list of folders with columns for 'Name' and 'Last Modified'. The 'Untitled Folder' is highlighted with a red box.

	Name	Last Modified
<input type="checkbox"/>	AGuerraB	6 hours ago
<input type="checkbox"/>	EderPineda	an hour ago
<input type="checkbox"/>	Files	13 days ago
<input type="checkbox"/>	MGuillen	6 hours ago
<input type="checkbox"/>	Roenna	6 hours ago
<input checked="" type="checkbox"/>	Untitled Folder	a day ago

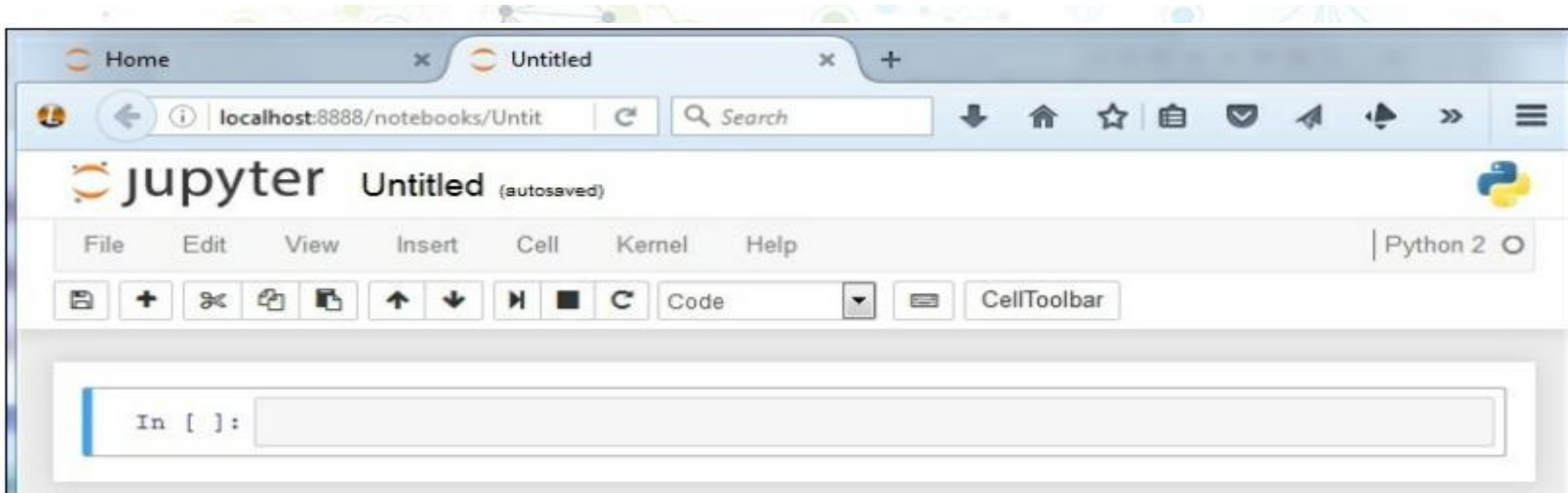
2. MANEJO DEL EDITOR JUPYTER FOR PYTHON

Crear Archivo



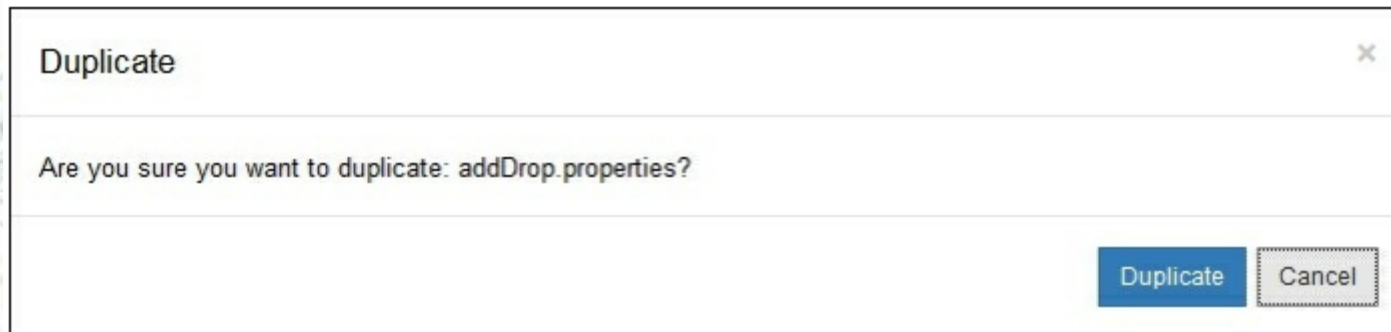
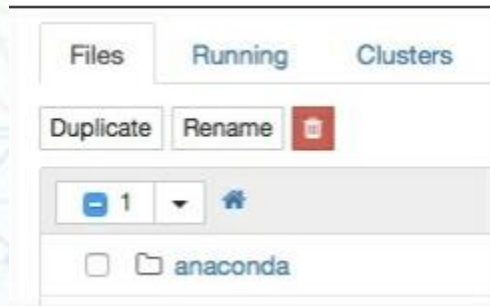
2. MANEJO DEL EDITOR JUPYTER FOR PYTHON

Crear Archivo



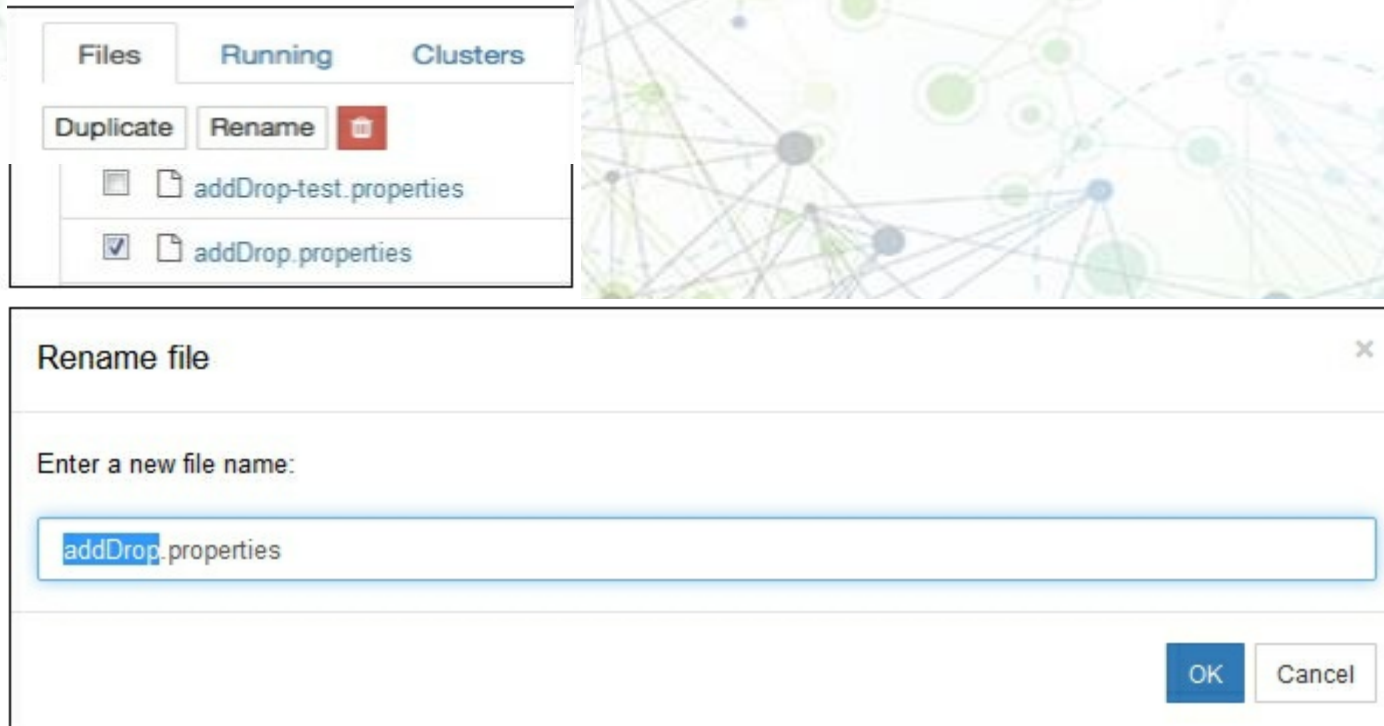
2. MANEJO DEL EDITOR JUPYTER FOR PYTHON

Duplicar archivos.



2. MANEJO DEL EDITOR JUPYTER FOR PYTHON

Renombrar Archivo



The screenshot displays the JupyterLab interface with the 'Files' tab selected. In the top-left pane, a file list shows 'addDrop-test.properties' and 'addDrop.properties', with the latter selected. A 'Rename' button is visible. A 'Rename file' dialog box is open in the foreground, prompting the user to 'Enter a new file name:'. The text 'addDrop.properties' is entered in the input field. The dialog includes 'OK' and 'Cancel' buttons at the bottom right.

Files Running Clusters

Duplicate Rename

☐ addDrop-test.properties

☒ addDrop.properties

Rename file

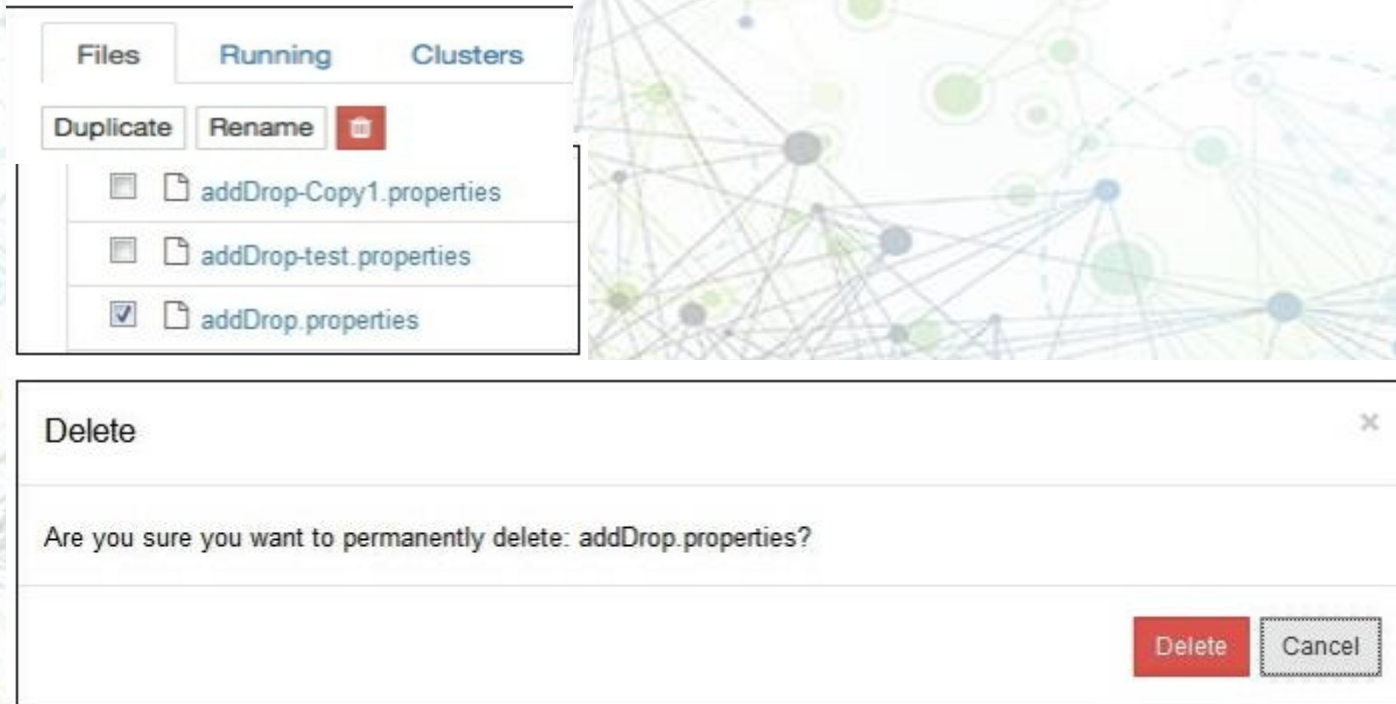
Enter a new file name:

addDrop.properties

OK Cancel

2. MANEJO DEL EDITOR JUPYTER FOR PYTHON

Borrar Archivo



2. MANEJO DEL EDITOR JUPYTER FOR PYTHON

Subir Archivo (Carpeta Local)



The screenshot shows the JupyterLab interface. At the top left is the Jupyter logo. To its right is a 'Logout' button. Below the logo are three tabs: 'Files', 'Running', and 'Clusters'. The 'Files' tab is active. Below the tabs is a text prompt: 'Select items to perform actions on them.' To the right of this prompt are three buttons: 'Upload' (highlighted with a red box), 'New', and a refresh icon. Below these buttons is a file list table. The table has two columns: 'Name' and 'Last Modified'. The first row is a folder named 'EderPineda'. Below it are several files, including 'Arbol de Decision.ipynb', 'Bayes.ipynb', 'Caso Practico.ipynb', 'clase.ipynb' (which is marked as 'Running'), 'Clustering.ipynb', and 'Ingesta Spark (DB)-Caso.ipynb'. The 'Last Modified' column shows the time since each file was last modified, such as 'seconds ago', '17 days ago', and '11 hours ago'.

jupyter Logout

Files Running Clusters

Select items to perform actions on them.

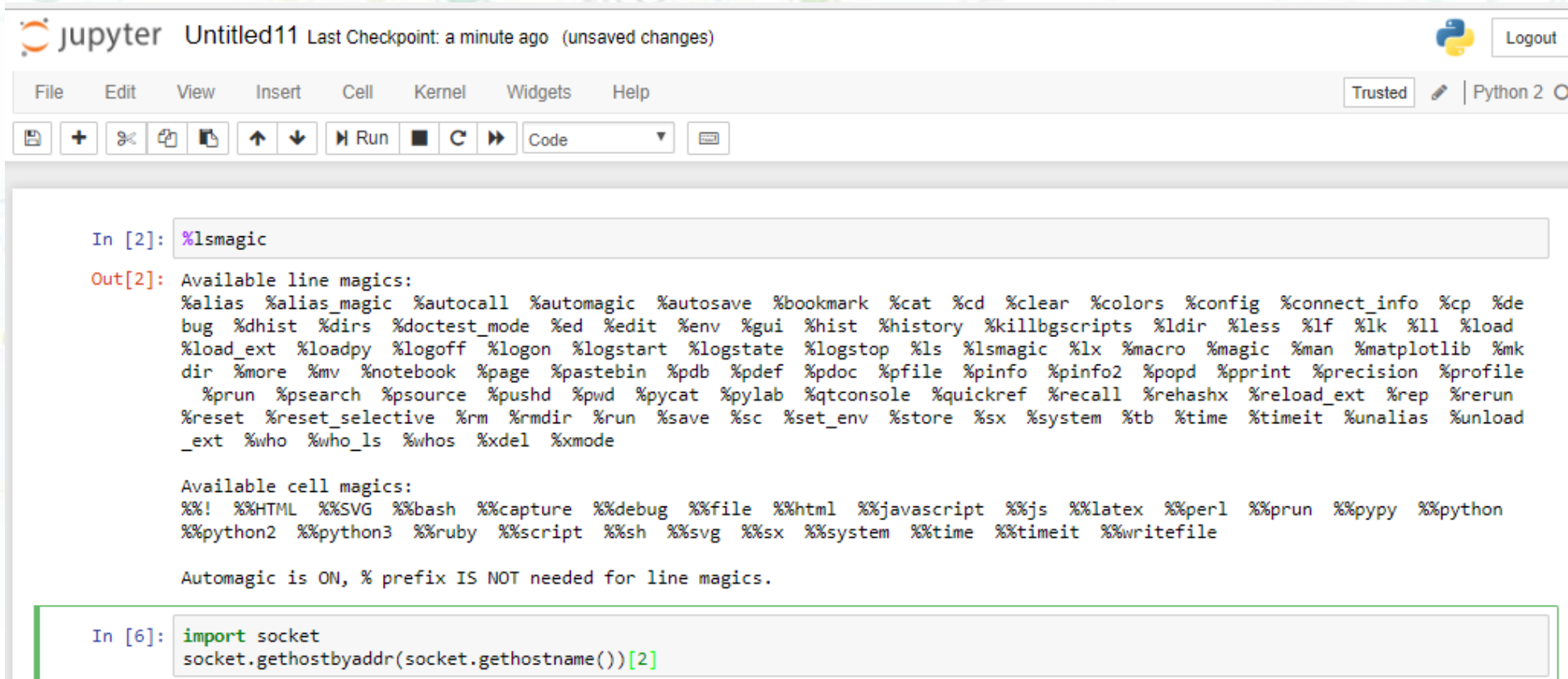
Upload New ↕

0 / EderPineda

	Name ↓	Last Modified
..		seconds ago
<input type="checkbox"/>	Arbol de Decision.ipynb	17 days ago
<input type="checkbox"/>	Bayes.ipynb	17 days ago
<input type="checkbox"/>	Caso Practico.ipynb	2 days ago
<input type="checkbox"/>	clase.ipynb	Running 11 hours ago
<input type="checkbox"/>	Clustering.ipynb	17 days ago
<input type="checkbox"/>	Ingesta Spark (DB)-Caso.ipynb	17 days ago

2. MANEJO DEL EDITOR JUPYTER FOR PYTHON

Comandos del editor de Jupyter



The screenshot displays the Jupyter Notebook interface. At the top, the header shows the Jupyter logo, the file name 'Untitled11', and the status 'Last Checkpoint: a minute ago (unsaved changes)'. On the right, there is a 'Logout' button and a Python 2 logo. Below the header is a menu bar with options: File, Edit, View, Insert, Cell, Kernel, Widgets, and Help. A toolbar contains icons for saving, opening, and running files, along with a 'Code' dropdown menu. The main area shows two input prompts. The first, 'In [2]:', is followed by the command '%lsmagic'. The output, 'Out[2]:', lists available line and cell magics. The line magics include %alias, %alias_magic, %autocall, %automagic, %autosave, %bookmark, %cat, %cd, %clear, %colors, %config, %connect_info, %cp, %debug, %dhist, %dirs, %doctest_mode, %ed, %edit, %env, %gui, %hist, %history, %killbgscripts, %ldir, %less, %lf, %lk, %ll, %load, %load_ext, %loadpy, %logoff, %logon, %logstart, %logstate, %logstop, %ls, %lsmagic, %lx, %macro, %magic, %man, %matplotlib, %mkdir, %more, %mv, %notebook, %page, %pastebin, %pdb, %pdef, %pdoc, %pfile, %pinfo, %pinfo2, %popd, %pprint, %precision, %profile, %prun, %psearch, %psource, %pushd, %pwd, %pycat, %pylab, %qtconsole, %quickref, %recall, %rehashx, %reload_ext, %rep, %rerun, %reset, %reset_selective, %rm, %rmdir, %run, %save, %sc, %set_env, %store, %sx, %system, %tb, %time, %timeit, %unalias, %unload, %ext, %who, %who_ls, %whos, %xdel, and %xmode. The cell magics include %!, %%HTML, %%SVG, %%bash, %%capture, %%debug, %%file, %%html, %%javascript, %%js, %%latex, %%perl, %%prun, %%pypy, %%python, %%python2, %%python3, %%ruby, %%script, %%sh, %%svg, %%sx, %%system, %%time, %%timeit, and %%writefile. A note at the bottom states 'Automagic is ON, % prefix IS NOT needed for line magics.' The second input prompt, 'In [6]:', is followed by the code 'import socket' and 'socket.gethostbyaddr(socket.gethostname())[2]'.

jupyter Untitled11 Last Checkpoint: a minute ago (unsaved changes) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 2

In [2]: %lsmagic

Out[2]: Available line magics:
%alias %alias_magic %autocall %automagic %autosave %bookmark %cat %cd %clear %colors %config %connect_info %cp %debug %dhist %dirs %doctest_mode %ed %edit %env %gui %hist %history %killbgscripts %ldir %less %lf %lk %ll %load %load_ext %loadpy %logoff %logon %logstart %logstate %logstop %ls %lsmagic %lx %macro %magic %man %matplotlib %mkdir %more %mv %notebook %page %pastebin %pdb %pdef %pdoc %pfile %pinfo %pinfo2 %popd %pprint %precision %profile %prun %psearch %psource %pushd %pwd %pycat %pylab %qtconsole %quickref %recall %rehashx %reload_ext %rep %rerun %reset %reset_selective %rm %rmdir %run %save %sc %set_env %store %sx %system %tb %time %timeit %unalias %unload %ext %who %who_ls %whos %xdel %xmode

Available cell magics:
%%! %%HTML %%SVG %%bash %%capture %%debug %%file %%html %%javascript %%js %%latex %%perl %%prun %%pypy %%python %%python2 %%python3 %%ruby %%script %%sh %%svg %%sx %%system %%time %%timeit %%writefile

Automagic is ON, % prefix IS NOT needed for line magics.

In [6]: import socket
socket.gethostbyaddr(socket.gethostname())[2]

2. MANEJO DEL EDITOR JUPYTER FOR PYTHON

Sintaxis

- Usa tabulación para mostrar la estructura de bloques.
- Tabula una vez para indicar comienzo de bloque.
- Des-Tabula para indicar el final del bloque.

Python

```
if x:
    if y:
        accionx+y
    else:
        accionx-y
```

C

```
if (x){
    if (y){
        accionx+y
    }
    else{
        accionx-y
    }
}
```



2. MANEJO DEL EDITOR JUPYTER FOR PYTHON

OPERADORES ARITMETICOS			
Operador	Descripción	Ejemplo	Resultado
+	Suma	<code>c = 3 + 5</code>	<code>c = 8</code>
-	Resta	<code>c = 4 - 2</code>	<code>c = 2</code>
-	Negación	<code>c = -7</code>	<code>c = -7</code>
*	Multiplicación	<code>c = 3 * 6</code>	<code>c = 18</code>
**	Potenciación	<code>c = 2 ** 3</code>	<code>c = 8</code>
/	División	<code>c = 7.5 / 2</code>	<code>c = 3.75</code>
//	División entera	<code>c = 7.5 // 2</code>	<code>c = 3.0</code>
%	Módulo	<code>c = 8 % 3</code>	<code>c = 2</code>

2. MANEJO DEL EDITOR JUPYTER FOR PYTHON

Operadores Relacionales

Operador	Descripción	Ejemplo
<code>==</code>	¿son iguales a y b?	<code>r = 5 == 3 # r es False</code>
<code>!=</code>	¿son distintos a y b?	<code>r = 5 != 3 # r es True</code>
<code><</code>	¿es a menor que b?	<code>r = 5 < 3 # r es False</code>
<code>></code>	¿es a mayor que b?	<code>r = 5 > 3 # r es True</code>
<code><=</code>	¿es a menor o igual que b?	<code>r = 5 <= 5 # r es True</code>
<code>>=</code>	¿es a mayor o igual que b?	<code>r = 5 >= 3 # r es True</code>

2. MANEJO DEL EDITOR JUPYTER FOR PYTHON

Comandos Importantes

Getwd:

- Obtiene el directorio actual

Code:

```
pwd
```

```
import os  
print os.getcwd()
```

setwd:

Asigna un directorio de trabajo

Code:

```
cd /ruta
```

```
Import os  
os.chdir('/home/epinedac/data')
```

2. MANEJO DEL EDITOR JUPYTER FOR PYTHON

Comandos Importantes

Asignación:

- Para almacenar alguna variable en memoria

Code:

=

del:

Para borrar un objeto de memoria

Code:

del variable

2. MANEJO DEL EDITOR JUPYTER FOR PYTHON

Comandos Importantes

Borrar Todo:

- Si deseamos dejar limpia todo nuestro entorno de trabajo

Code:

```
%reset -f
```

Paste:

Este comando sirve para concatenar cadenas

Code:

```
"eder"+"pineda"
```


2. MANEJO DEL EDITOR JUPYTER FOR PYTHON

Comandos Importantes

Class:

- Obtiene el tipo de objeto creado

Code:

```
type(variable)
```

Comentarios:

Para comentar nuestro código para hacerlo mas entendible, utilizamos el simbolo #

Code:

```
# Añadiendo comentario
```

2. MANEJO DEL EDITOR JUPYTER FOR PYTHON

Comandos Importantes

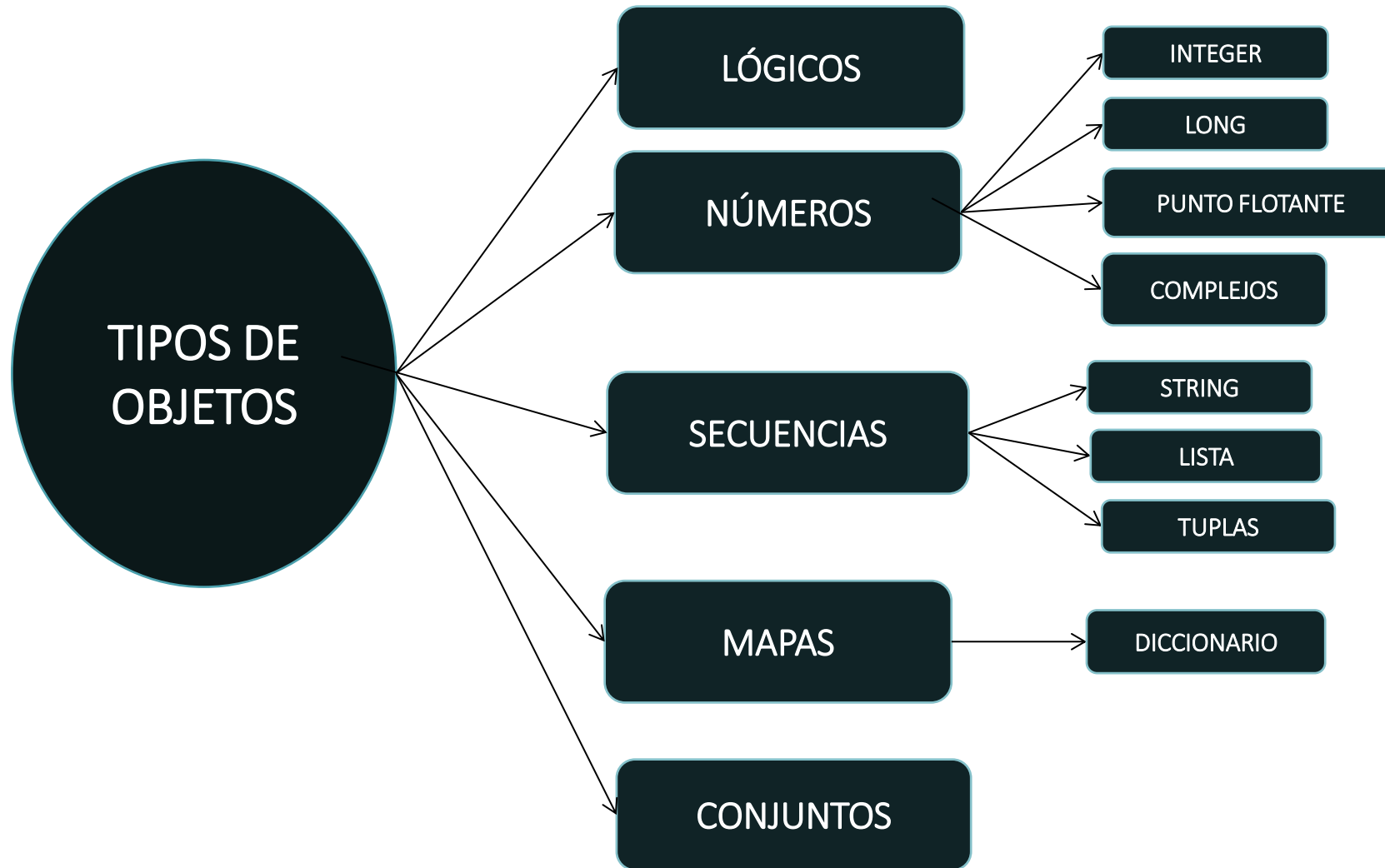
help:

- Brinda una descripción de ayuda de la función.

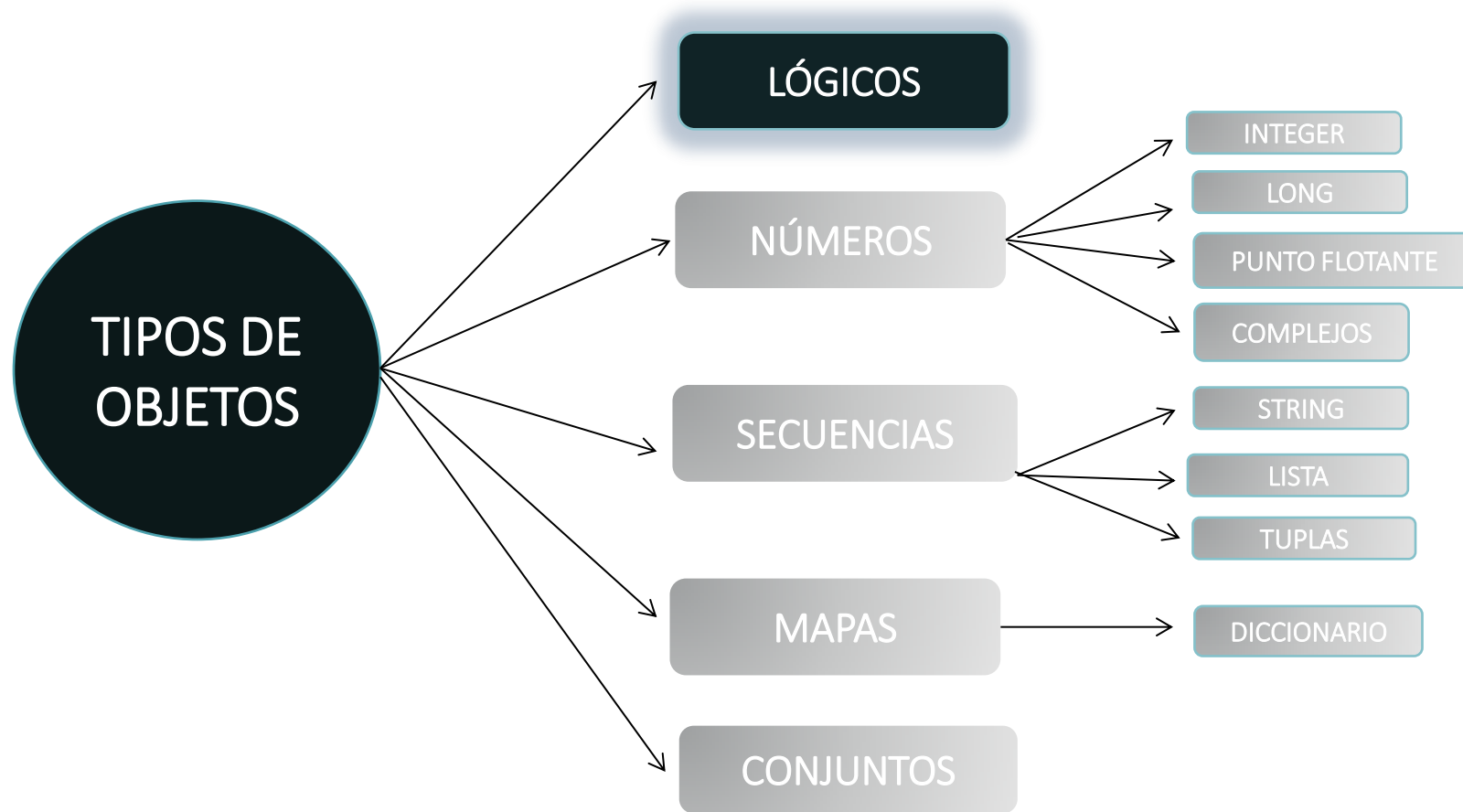
Code:

```
Help(funcion)
```

3. TIPOS DE OBJETOS



3. TIPOS DE OBJETOS



3. TIPOS DE OBJETOS

Logicos

- Este tipo de datos almacena valores **Boolean** (TRUE y FALSE), y pueden ser operados mediante operadores booleanos como se muestran a continuación:
- .

OR		TRUE	FALSE
TRUE		TRUE	TRUE
FALSE		TRUE	FALSE

XOR	^	TRUE	FALSE
TRUE		FALSE	TRUE
FALSE		TRUE	FALSE

AND	&	TRUE	FALSE
TRUE		TRUE	FALSE
FALSE		FALSE	FALSE

NOT	
TRUE	FALSE
FALSE	TRUE

3. TIPOS DE OBJETOS

Logicos

- Ejemplos:

Code:

```
In [36]: True
```

```
Out[36]: True
```

```
In [42]: not True
```

```
Out[42]: False
```

```
In [40]: True and True
```

```
Out[40]: True
```

```
In [41]: False or False
```

```
Out[41]: False
```

```
In [43]: True ^ False
```

```
Out[43]: True
```

```
In [44]: (True and False) or (not True)
```

```
Out[44]: False
```

3. TIPOS DE OBJETOS

Logicos

- Ejemplos:

Code:

```
In [45]: 2>3 # Operador mayor que
```

```
Out[45]: False
```

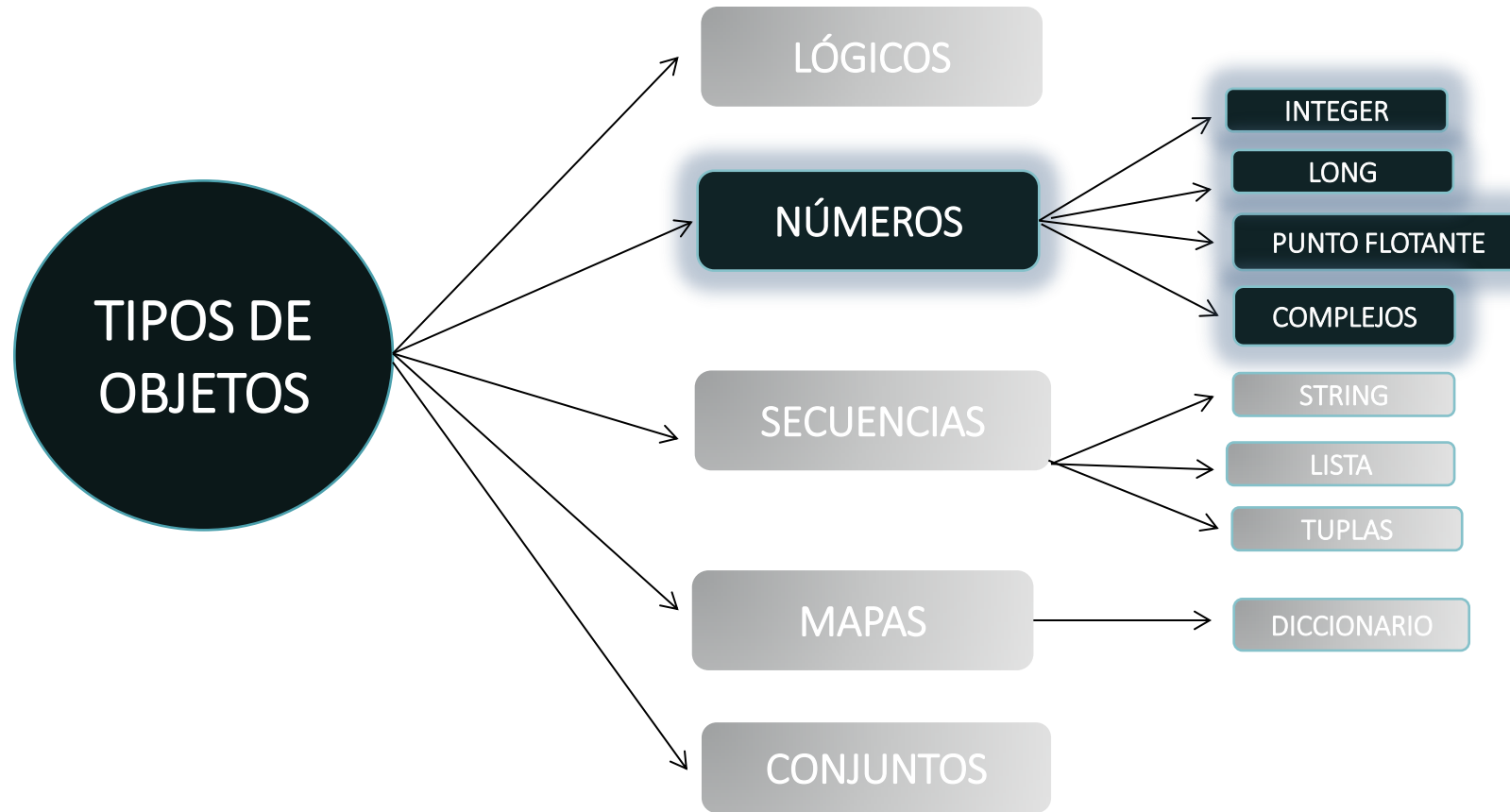
```
In [47]: 3==3 # Operador de igualdad
```

```
Out[47]: True
```

```
In [48]: (2+3)>(2-3) # Expresión compleja comparada logicamente
```

```
Out[48]: True
```


3. TIPOS DE OBJETOS



3. TIPOS DE OBJETOS

Numericos

Integer:

- Son aquellos números que no tienen decimales, tanto positivos como negativos (además del cero).

Code:

```
In [7]: 3
```

```
Out[7]: 3
```

```
In [8]: entero=3
```

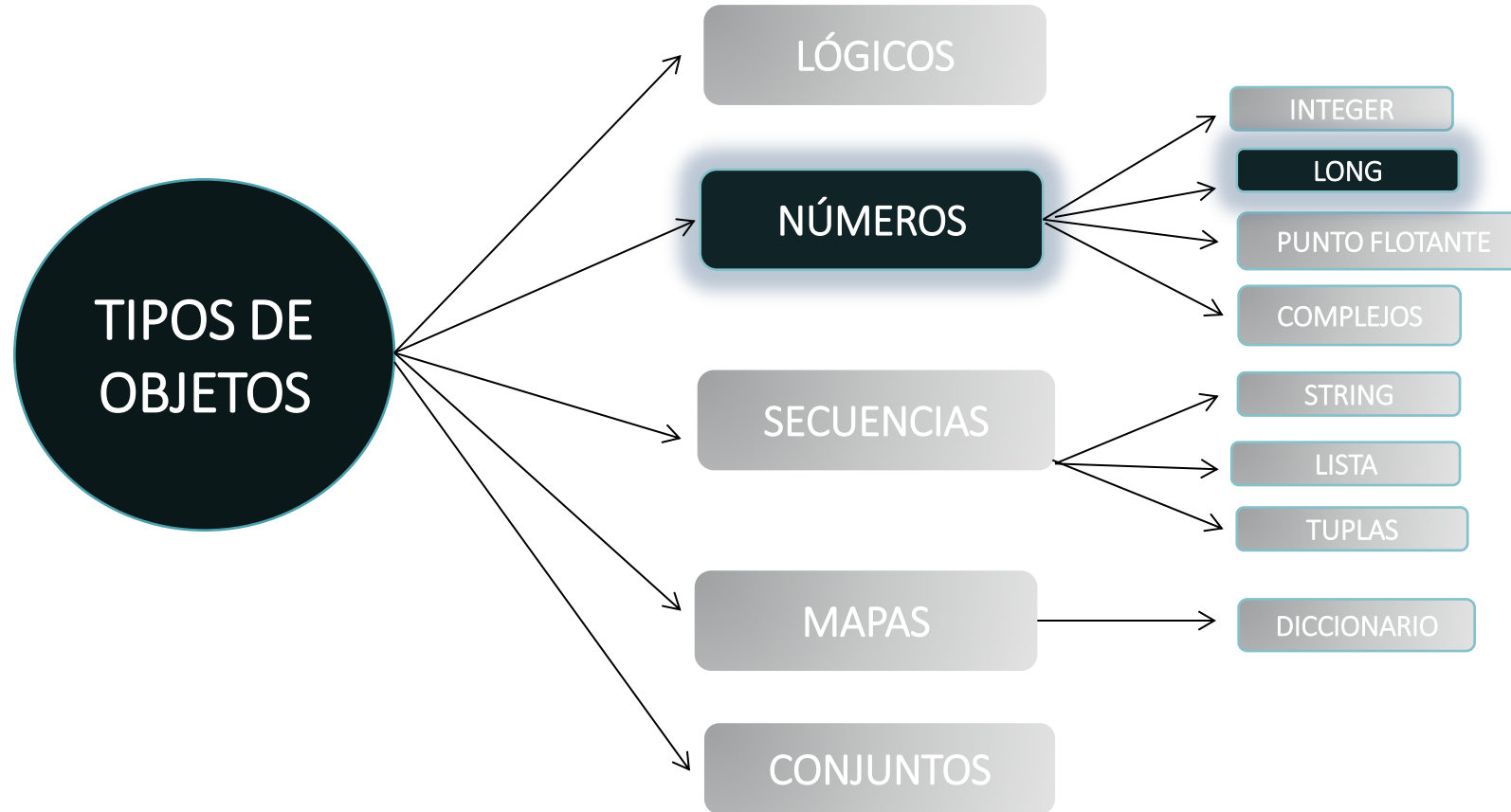
```
In [10]: print(entero+3)
```

```
6
```

```
In [11]: type(entero)
```

```
Out[11]: int
```

3. TIPOS DE OBJETOS



3. TIPOS DE OBJETOS

Numericos

Long:

- Similar al entero pero soporta números mas grandes, se aconseja no utilizar este tipo de variable para no malgastar memoria, este tipo de objeto se ha unido al tipo entero a partir del python version 3.

Code:

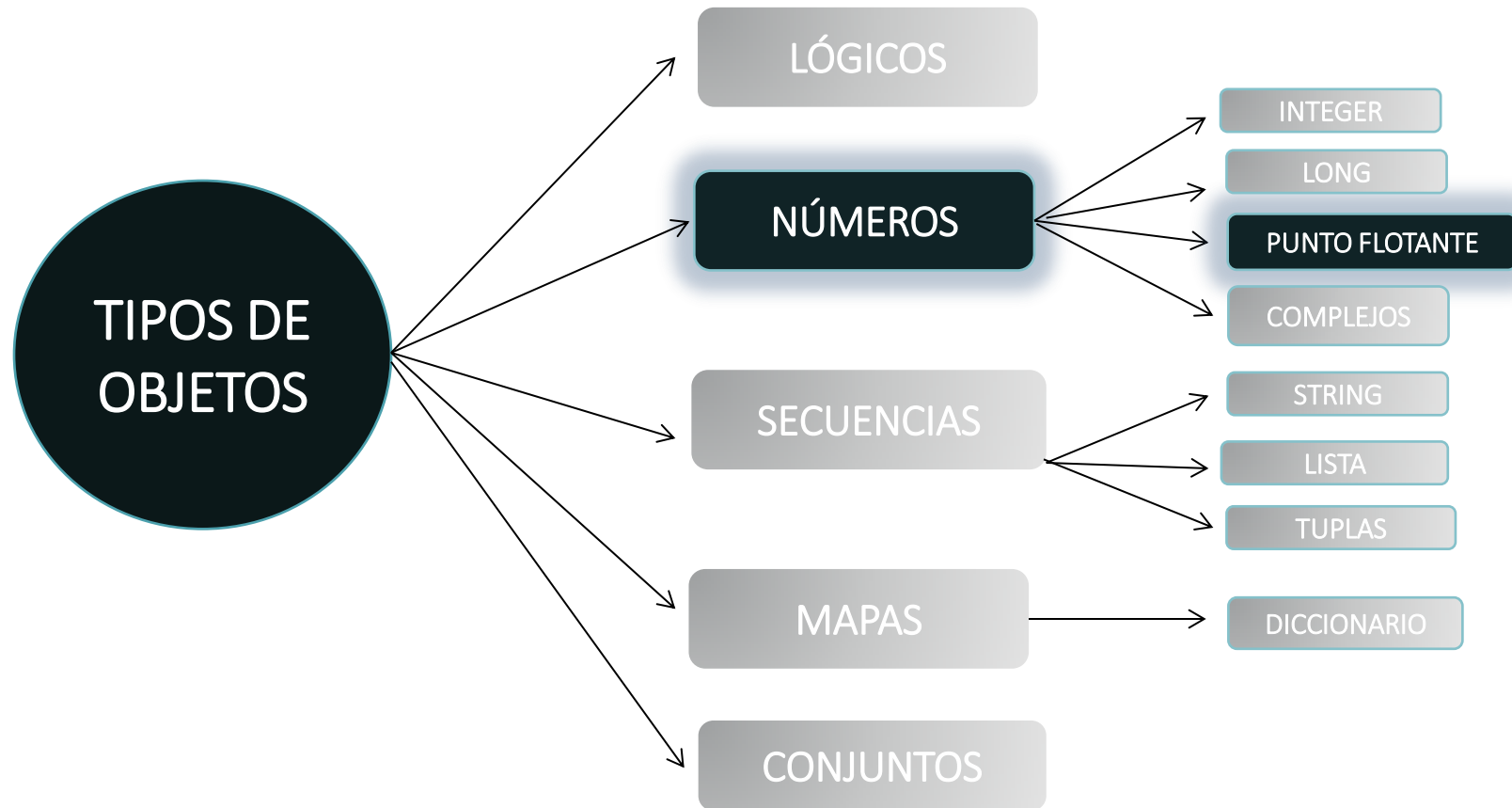
```
In [6]: 22222222222
```

```
Out[6]: 22222222222L
```

```
In [7]: type(22222222222L)
```

```
Out[7]: long
```

3. TIPOS DE OBJETOS



3. TIPOS DE OBJETOS

Numericos

Float:

- Son aquellos números que poseen decimales.

Code:

```
In [12]: 3.2
```

```
Out[12]: 3.2
```

```
In [13]: float=3.2
```

```
In [14]: print(float+3)
```

```
6.2
```

```
In [15]: type(float)
```

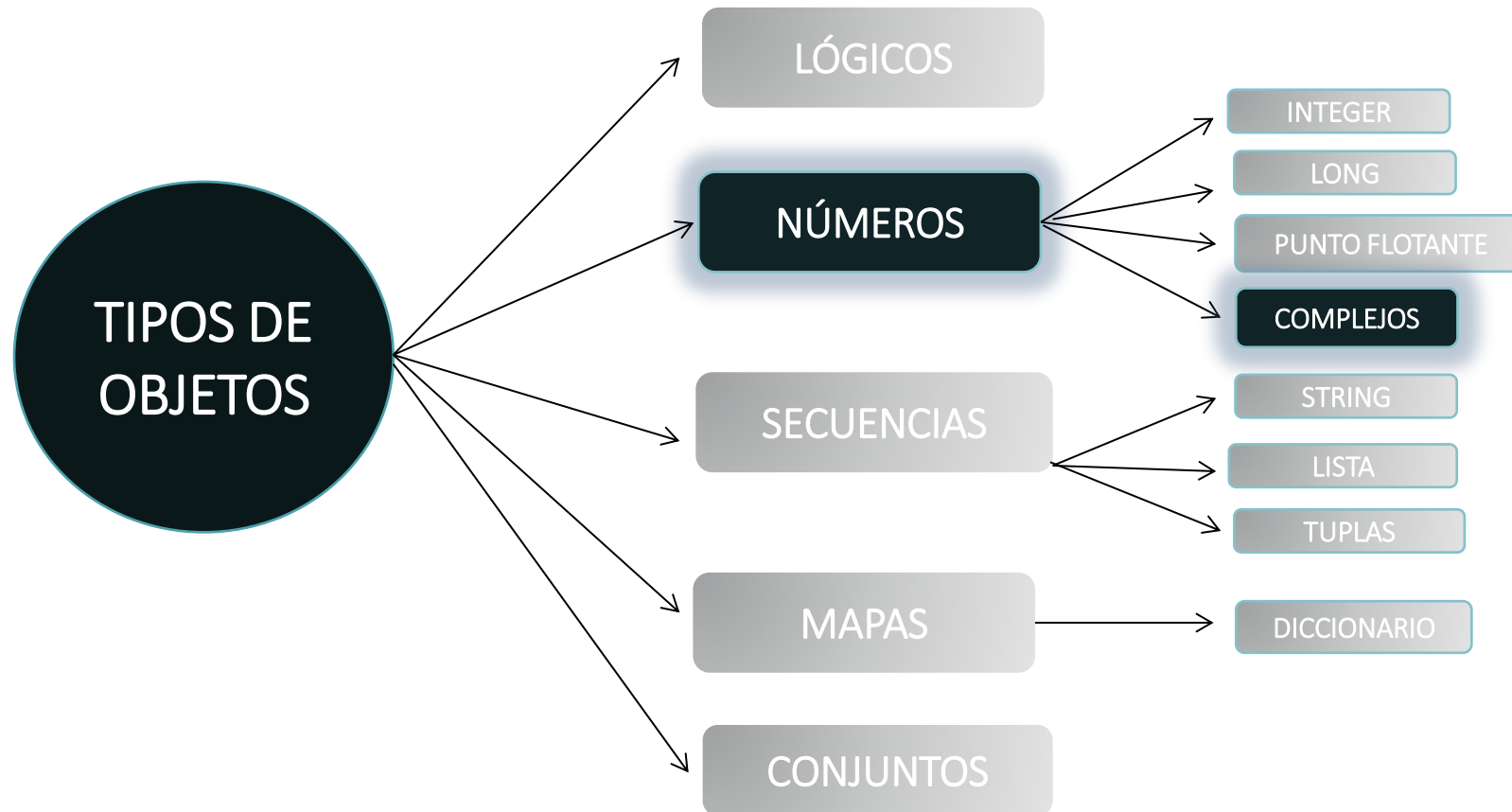
```
Out[15]: float
```

```
In [22]: # Adicionalmente se puede utilizar notación científica para indicar un exponente en base 10  
float_1=0.3e2
```

```
In [23]: type(float_1)
```

```
Out[23]: float
```

3. TIPOS DE OBJETOS



3. TIPOS DE OBJETOS

Numericos

Complex

- Este tipo de variable es usado en ciencia y estudios de ingeniería.

Code:

```
In [8]: 4+6j
```

```
Out[8]: (4+6j)
```

```
In [9]: type(4+6j)
```

```
Out[9]: complex
```

```
In [10]: complex(2,3)
```

```
Out[10]: (2+3j)
```

3. TIPOS DE OBJETOS



3. TIPOS DE OBJETOS



3. TIPOS DE OBJETOS

Sequences

String

- Un objeto del tipo String es meramente una secuencia de caracteres, los cuales pueden ser codificados en Mayusculas o minusculas, por tanto son case-sensitive.

Code:

```
In [19]: Nombre='Eder'  
Nombre
```

```
Out[19]: 'Eder'
```

```
In [20]: type(Nombre)
```

```
Out[20]: str
```

```
In [21]: 'Eder'+'Pineda'
```

```
Out[21]: 'EderPineda'
```

```
In [22]: nombre='Eder'  
nombre+' Pineda'
```

```
Out[22]: 'Eder Pineda'
```

3. TIPOS DE OBJETOS



3. TIPOS DE OBJETOS

Sequences

List []

Es similar a la tupla con la diferencia que permite modificar los datos una vez creado, y se define con [].

Code:

```
In [30]: mi_lista=['987456111',15,True,90.4,'Prepago']
```

```
In [31]: print(mi_lista)
['987456111', 15, True, 90.4, 'Prepago']
```

```
In [32]: mi_lista[1]
```

```
Out[32]: 15
```

```
In [33]: mi_lista[1:4]
```

```
Out[33]: [15, True, 90.4]
```

```
In [34]: mi_lista[-2]
```

```
Out[34]: 90.4
```

3. TIPOS DE OBJETOS

Sequences

List []

Code:

```
In [36]: mi_lista[1]=18  
mi_lista
```

```
Out[36]: ['987456111', 18, True, 90.4, 'Prepago']
```

```
In [37]: mi_lista.append('Nuevo Dato')  
mi_lista
```

```
Out[37]: ['987456111', 18, True, 90.4, 'Prepago', 'Nuevo Dato']
```

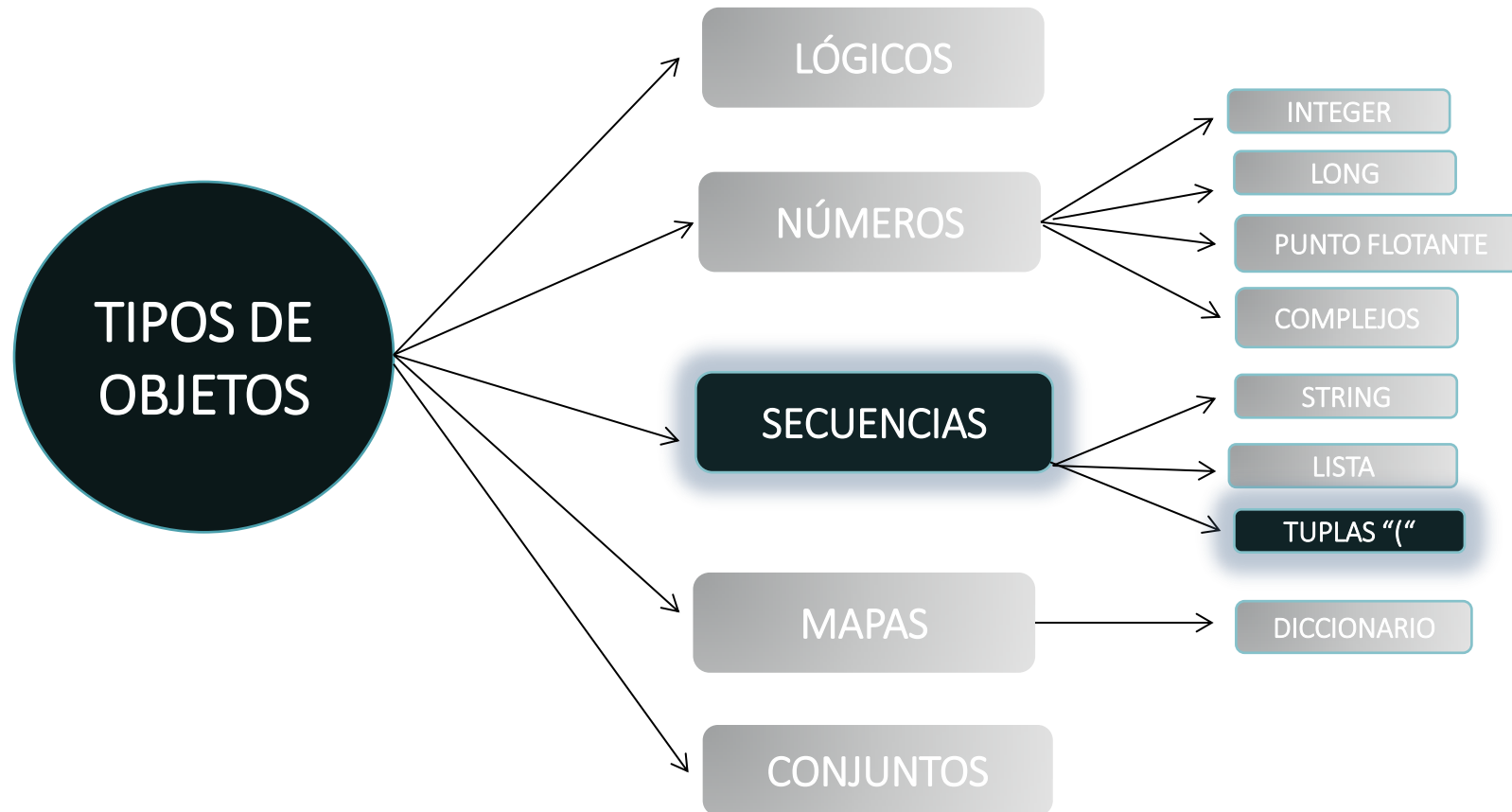
```
In [38]: mi_lista.extend([1,2,3,9,3,4,5])  
mi_lista
```

```
Out[38]: ['987456111', 18, True, 90.4, 'Prepago', 'Nuevo Dato', 1, 2, 3, 9, 3, 4, 5]
```

```
In [39]: del mi_lista[12]  
mi_lista
```

```
Out[39]: ['987456111', 18, True, 90.4, 'Prepago', 'Nuevo Dato', 1, 2, 3, 9, 3, 4]
```


3. TIPOS DE OBJETOS



3. TIPOS DE OBJETOS

Sequences

Tuplas ()

Es una variable que permite almacenar varios datos inmutables de tipos diferentes, y se define con ().

Code:

```
In [41]: mi_tupla=('987456111',15,True,90.4,'Prepago')
mi_tupla
```

```
Out[41]: ('987456111', 15, True, 90.4, 'Prepago')
```

```
In [43]: print(mi_tupla[1])
print(mi_tupla[1:3])
```

```
15
(15, True)
```

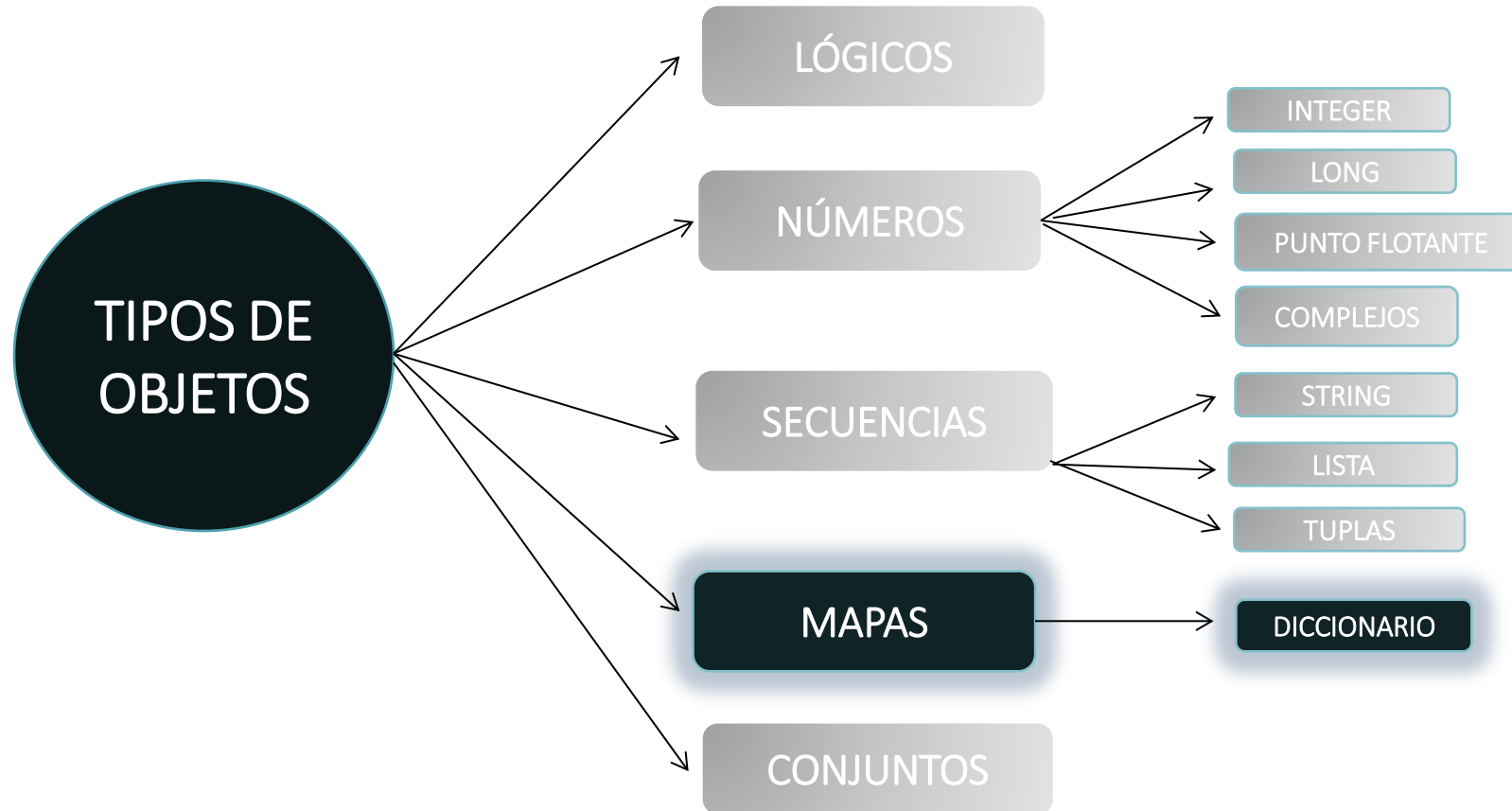
```
In [44]: print(mi_tupla[:3])

('987456111', 15, True)
```

```
In [47]: print(mi_tupla[-1])

Prepago
```

3. TIPOS DE OBJETOS



3. TIPOS DE OBJETOS

Mapping (Mapas)

Diccionario {}

Es un conjunto desordenado de parejas clave-valor.

Code:

```
In [53]: un_dic={'Star Wars':'George Lucas','Kill Bill':'Tarantino','A.I.': 'Steven Spielberg'}  
un_dic
```

```
Out[53]: {'A.I.': 'Steven Spielberg',  
          'Kill Bill': 'Tarantino',  
          'Star Wars': 'George Lucas'}
```

```
In [54]: un_dic['Star Wars']
```

```
Out[54]: 'George Lucas'
```

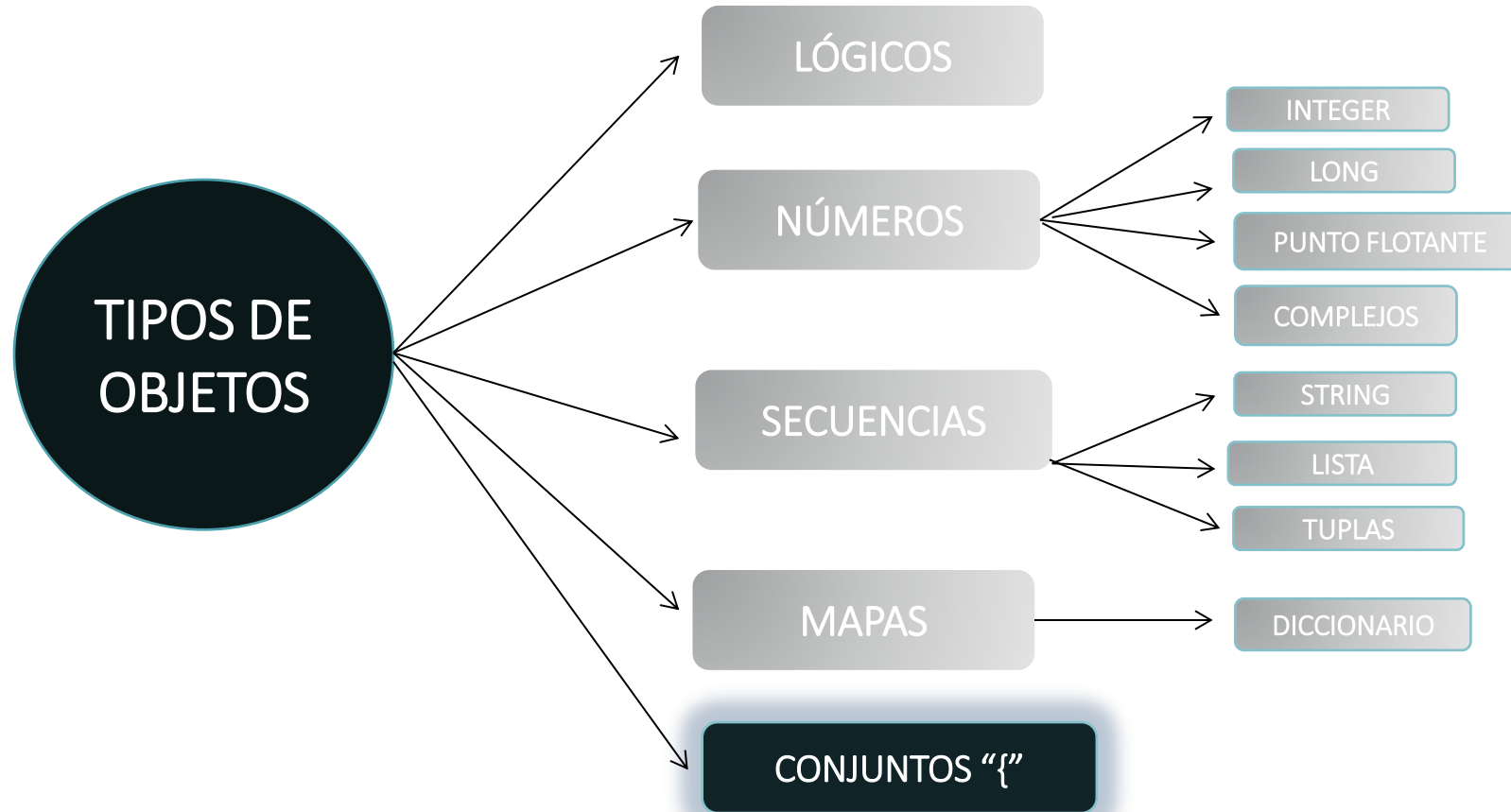
```
In [55]: un_dic['Star Wars']="Garet Edwards"  
print(un_dic)
```

```
{'Star Wars': 'Garet Edwards', 'A.I.': 'Steven Spielberg', 'Kill Bill': 'Tarantino'}
```

```
In [56]: del un_dic['Star Wars']  
un_dic
```

```
Out[56]: {'A.I.': 'Steven Spielberg', 'Kill Bill': 'Tarantino'}
```

3. TIPOS DE OBJETOS



3. TIPOS DE OBJETOS

Sets(Conjuntos)

Conjuntos

Es una no ordenada colección de objetos (a nivel de memoria).

Puede contener simultáneamente valores de cualquier tipo de datos.

Con dos conjuntos se pueden realizar las típicas operaciones de unión, intersección y diferencia de conjuntos.

No permite valores repetidos.

Esta definido por {}

Code:

```
In [1]: set(['h','e','l','l','o',1,2.0,3+4j])
```

```
Out[1]: {1, 2.0, 'e', 'h', 'l', 'o', (3+4j)}
```


3. TIPOS DE OBJETOS

Sets(Conjuntos)

Conjuntos

```
>>> un_conjunto={1,2,3,4,5}  
>>> un_conjunto
```

```
>>> un_conjunto.add(6)  
>>> un_conjunto
```

```
>>> un_conjunto.update({7,8})  
>>> un_conjunto
```

Code:

```
>>> un_conjunto.discard(8)  
>>> un_conjunto
```

```
>>> un_conjunto.remove(7)  
>>> un_conjunto
```



3. TIPOS DE OBJETOS

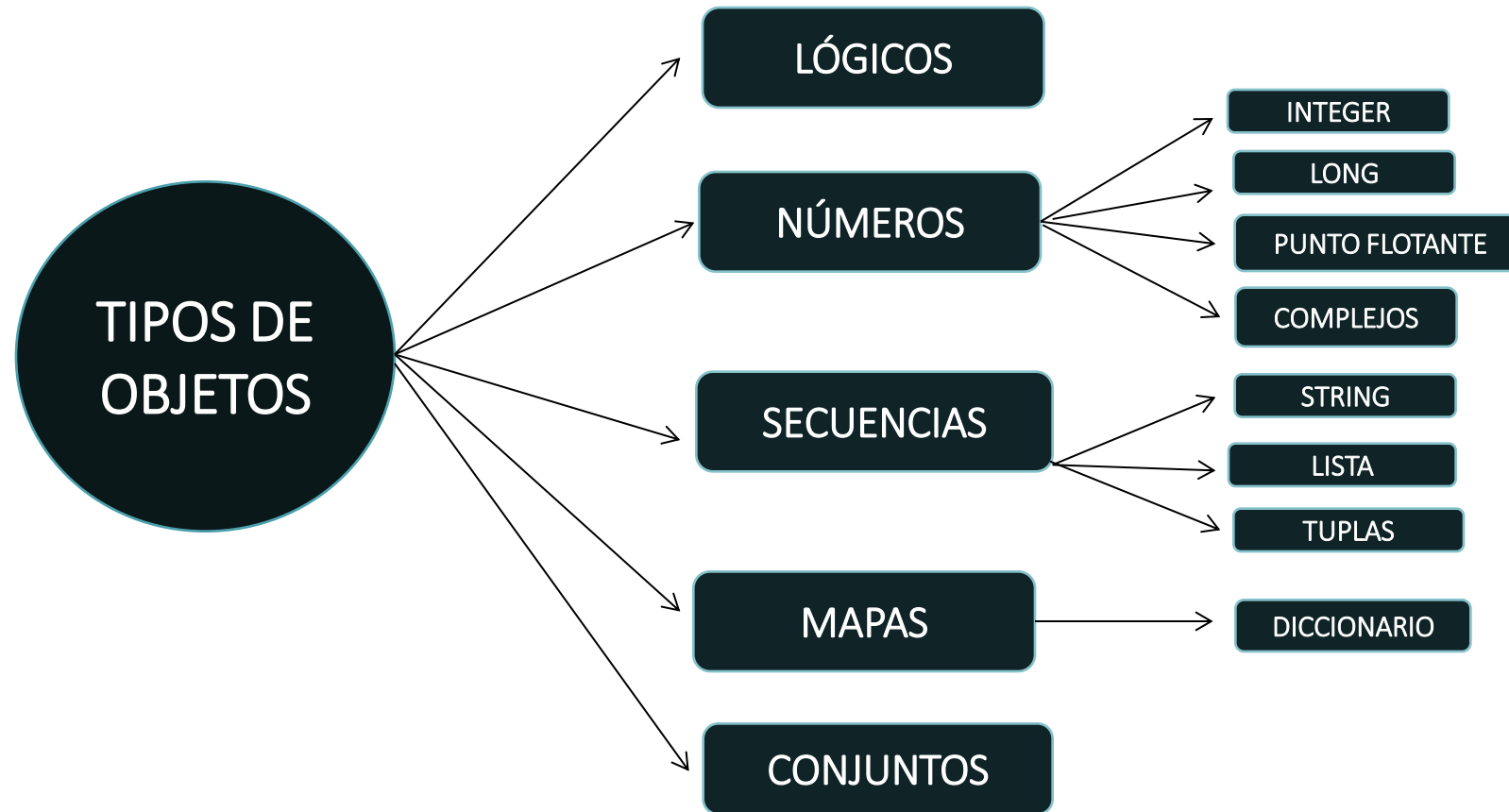
Sets(Conjuntos)

Conjuntos

Code:

```
>>> un_conjunto = {2, 4, 5, 9, 12, 21, 30, 51, 76, 127, 195}
>>> 30 in un_conjunto
True
>>> 31 in un_conjunto
False
>>> otro_conjunto = {1, 2, 3, 5, 6, 8, 9, 12, 15, 17, 18, 21}
>>> un_conjunto.union(otro_conjunto)
{1, 2, 195, 4, 5, 6, 8, 12, 76, 15, 17, 18, 3, 21, 30, 51, 9, 127}
>>> un_conjunto.intersection(otro_conjunto)
{9, 2, 12, 5, 21}
>>> un_conjunto.difference(otro_conjunto)
{195, 4, 76, 51, 30, 127}
```

3. TIPOS DE OBJETOS



4. ESTRUCTURAS DE CONTROL (IF, ELSE, WHILE, FOR)

IF:

- Permite que un programa ejecute unas instrucciones cuando se cumple una condición

`if` condición:

aquí van las órdenes que se ejecutan si la condición es cierta
y que pueden ocupar varias líneas

Code:

```
numAños=14  
if numAños >= 18:  
    print("Mayor de Edad")
```

4. ESTRUCTURAS DE CONTROL (IF, ELSE, WHILE, FOR)

IF-ELSE:

- Permite que un programa ejecute unas instrucciones cuando se cumple una condición y otras instrucciones cuando no se cumple esa condición

Code:

```
numAños=14
if numAños > 18:
    print("Mayor de Edad")
else:
    print("Menor de Edad")
```

if condición:

aquí van las órdenes que se ejecutan si la condición es cierta y que pueden ocupar varias líneas

else:

y aquí van las órdenes que se ejecutan si la condición es falsa y que también pueden ocupar varias líneas

4. ESTRUCTURAS DE CONTROL (IF, ELSE, WHILE, FOR)

IF-ELSEIF - ELSE:

- Permite encadenar varias condiciones, elif es una contracción de if-else

Code:

```
numAños=74
if (numAños < 18):
    print("Menor de Edad")
elif (numAños > 18 and numAños<65):
    print("Mayor de Edad")
else:
    print("Adulto Mayor")
```

```
if condición_1:
    bloque 1
elif condición_2:
    bloque 2
else:
    bloque 3
```

```
if condición_1:
    bloque 1
else:
    if condición_2:
        bloque 2
    else:
        bloque 3
```


4. ESTRUCTURAS DE CONTROL (IF, ELSE, WHILE, FOR)

WHILE:

- Permite repetir la ejecución de un grupo de instrucciones mientras se cumpla la condición.

`while` condición:
cuerpo del bucle

Code:

```
In [2]: _iteracion=1

while(_iteracion < 12):
    if(_iteracion==5):
        _iteracion=16
    print("Iteracion:",_iteracion)
    _iteracion+=1

('Iteracion:', 1)
('Iteracion:', 2)
('Iteracion:', 3)
('Iteracion:', 4)
('Iteracion:', 16)
```

4. ESTRUCTURAS DE CONTROL (IF, ELSE, WHILE, FOR)

FOR:

- Es un bucle que repite el bucle un número predeterminado de veces

`for` variable `in` elemento iterable (lista, cadena, range, etc.):
cuerpo del bucle

Code:

```
In [2]: # Creamos una lista que contenga tus datos

_listaPersona=['Eder','Pineda Claros',34,'M','Superior']
for item in _listaPersona:
    print("El dato actual es:",item)
```

```
('El dato actual es:', 'Eder')
('El dato actual es:', 'Pineda Claros')
('El dato actual es:', 34)
('El dato actual es:', 'M')
('El dato actual es:', 'Superior')
```

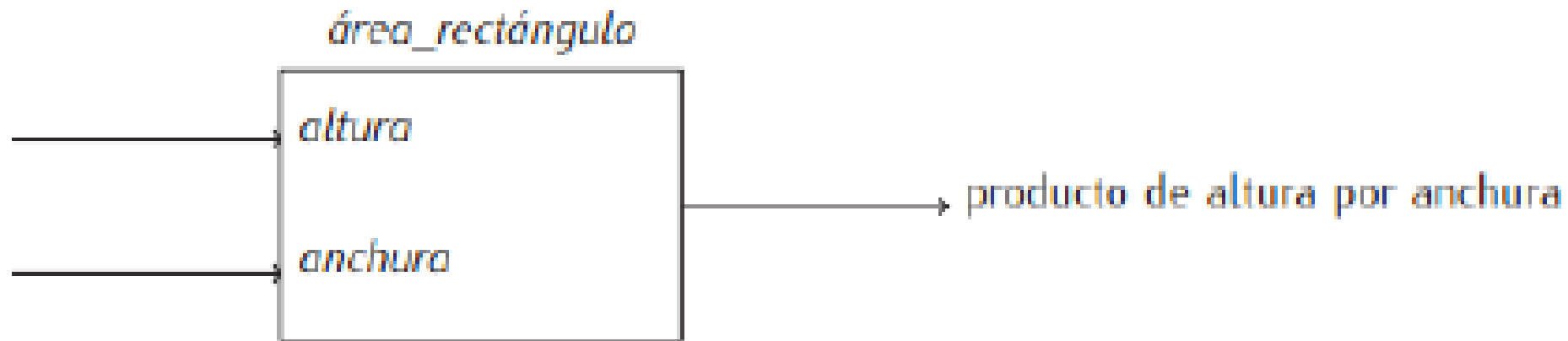
```
In [3]: _listaPersona[0]
```

```
Out[3]: 'Eder'
```

5. MANEJO DE FUNCIONES

FUNCIONES:

- Una función es una parte de un programa, donde un número de instrucciones de programación se agrupa como un bloque, y puede ser llamada cuando se desee.
- Habilita un enfoque modular para la programación de tareas.



5. MANEJO DE FUNCIONES

Función

Y la función se define de la siguiente manera:

```
def nombre_funcion(parametro_1, parametro_2, ...):  
    """Descripción de la función"""  
    # Comentarios de la sentencias a ejecutar  
    Sentencias  
    return parametros_retorno
```

Donde:

- Cabecera: comienza con el keyword def, y termina en :
- Descripción de la función: Un detalle que describe el proposito de la función (buena practica)
- Cuerpo: Sentencias que se encuentran en la parte inferior de la cabecera

FUNCIONES:

Tenemos algunos tipos de funciones, según su aplicación:

- Funciones sin parámetros sin retorno
- Funciones sin parámetros con retorno
- Funciones con parámetros sin retorno
- Funciones con parámetros con retorno

5. MANEJO DE FUNCIONES

FUNCIONES:

- Funciones sin parámetros sin retorno.

Code:

```
In [57]: def fnCursoBigData():  
        ...  
        La función fnCursoBigData da la bienvenida  
        al curso de Big Data Aplicado.  
        ...|  
        print("Bienvenidos al Curso de Big Data Aplicado")
```

```
In [59]: fnCursoBigData()  
  
Bienvenidos al Curso de Big Data Aplicado
```

```
In [63]: help(fnCursoBigData)  
  
Help on function fnCursoBigData in module __main__:  
  
fnCursoBigData()  
    La función fnCursoBigData da la bienvenida  
    al curso de Big Data Aplicado.
```

5. MANEJO DE FUNCIONES

FUNCIONES:

- Funciones sin parámetros con retorno.

Code:

```
In [3]: def _fedadminimavotacion():  
        '''La siguiente función devuelve la edad minima para poder realizar una votacion'''  
        return 18  
  
        print("La edad Minima de votacion es:",_fedadminimavotacion())  
  
('La edad Minima de votacion es:', 18)
```


5. MANEJO DE FUNCIONES

FUNCIONES:

- Funciones con parámetros sin retorno.

Code:

```
In [1]: def _esPar(numero):  
        '''funcion que indica si el numero ingresado es par o no'''  
        if numero%2==0:  
            print("El numero es par")  
        else:  
            print("El numero es impar")
```

```
In [3]: _esPar(7)  
  
El numero es impar
```

```
In [4]: help(_esPar)  
  
Help on function _esPar in module __main__:  
  
_esPar(numero)  
    funcion que indica si el numero ingresado es par o no
```

5. MANEJO DE FUNCIONES

FUNCIONES:

- Funciones con parámetros con retorno.

Code:

```
In [3]: def _areaRectangulo(altura,anchura):  
        '''funcion que devuelve el area del rectangulo  
        no se hace mencion de la escala de los parametros de entrada  
        ...  
        return altura*anchura  
  
        _resultado=_areaRectangulo(2,6)  
        print("el resultado es :",_resultado)  
  
('el resultado es :', 12)
```

```
In [4]: help(_areaRectangulo)  
  
Help on function _areaRectangulo in module __main__:  
  
_areaRectangulo(altura, anchura)  
    funcion que devuelve el area del rectangulo  
    no se hace mencion de la escala de los parametros de entrada
```

5. MANEJO DE FUNCIONES

FUNCIONES:

- Programación modular.

funciones_datos

```
1 def suma_datos(a,b):  
2     return (a+b)  
3 def resta_datos(a,b):  
4     return (a-b)  
5 def multip_datos(a,b):  
6     return (a*b)  
7
```

Importando librerías

```
: 1 import funciones_datos  
:  
:  
: 1 funciones_datos.suma_datos(2,3)
```

6. MANEJO DE LIBRERIAS , ARCHIVOS, PROCESAMIENTO Y PLOTTING

Librería (Modulos):

- Los modulos encapsulan funcionalidad, son colecciones de programas python usados para una especifica tarea, es imposible mencionar todos los modulos asi q a continuación se presentan los mas populares:

Field of Study	Name of Python Module
Scientific Computation	scipy, numpy, sympy
Statistics	pandas
Networking	networkx
Cryptography	pyOpenSSL
Game Development	PyGame
Graphic User Interface	pyQT
Machine Learning	scikit-learn, tensorflow
Image Processing	scikit-image
Plotting	Matplotlib
Database	SQLAlchemy
HTML and XML parsing	BeautifulSoup
Natural Language Processing	nltk
Testing	nose

6. MANEJO DE LIBRERIAS , ARCHIVOS, PROCESAMIENTO Y PLOTTING

Librería math:

- Es usada para realizar operaciones matemáticas, y veremos las distintas formas de importación.

Code:

```
In [7]: import math  
        math.sqrt(4)
```

```
Out[7]: 2.0
```

```
In [8]: from math import *  
        sqrt(4)
```

```
Out[8]: 2.0
```

```
In [13]: from math import pow  
         math.pow(2,4)
```

```
Out[13]: 16.0
```

6. MANEJO DE LIBRERIAS , ARCHIVOS, PROCESAMIENTO Y PLOTTING

Librería Numpy:

- El modulo Numpy contiene varias funciones para ser usadas en computación numerica, por lo tanto hace referencia a datos numericos Python.
- Uno de los objetos mas usados de numpy es el array, el cual es homoganeo (debe tener solo un tipo de dato).

Code:

```
In [38]: import numpy
print (numpy.version.version)
```

```
1.13.1
```

```
In [39]: # Creando un arreglo en función de una lista
_arrayNotas=[12,18,16,15]
_numpy_arrayNotas = numpy.array(_arrayNotas) # numpy.array(_arrayNotas, dtype=float)
print("Array contenido _numpy_arrayNotas:",_numpy_arrayNotas)
print("Tipo _numpy_arrayNotas:",type(_numpy_arrayNotas))
```

```
('Array contenido _numpy_arrayNotas:', array([12, 18, 16, 15]))
('Tipo _numpy_arrayNotas:', <type 'numpy.ndarray'>)
```

```
In [63]: print("Tipo de objeto:",_numpy_arrayNotas.dtype)
print("Forma del Array:",_numpy_arrayNotas.shape)
print("Tamaño del Array nXm:",_numpy_arrayNotas.size)
```

```
('Tipo de objeto:', dtype('int32'))
('Forma del Array:', (4L,))
('Tamaño del Array nXm:', 4)
```

6. MANEJO DE LIBRERIAS , ARCHIVOS, PROCESAMIENTO Y PLOTTING

Librería Numpy:

Code:

```
In [64]: nombres=('Eder','Melissa','Francisco','Amy')
          edades=(34,38,54,22)
          sexo=('M','F','M','F')
          _personas = numpy.array( [ nombres, edades, sexo ] ) # Definiendo arreglo
```

```
In [69]: print("Tipo de objeto:",_personas.dtype)
          print("Forma del Array (n, m):",_personas.shape)
          print("Tamaño del Array nXm:",_personas.size)

          ('Tipo de objeto:', dtype('S9'))
          ('Forma del Array (n, m):', (3L, 4L))
          ('Tamaño del Array nXm:', 12)
```

```
In [71]: _personas.reshape(1,12) # Transformando la forma del array
```

```
Out[71]: array([[ 'Eder', 'Melissa', 'Francisco', 'Amy', '34', '38', '54', '22', 'M',
                  'F', 'M', 'F']],
          dtype='|S9')
```


6. MANEJO DE LIBRERIAS , ARCHIVOS, PROCESAMIENTO Y PLOTTING

Librería Numpy:

Code:

```
In [86]: _ra=numpy.random.randint(0,20,[4,2]) # Un entero aleatorio hasta el numero 10
         _ra
```

```
Out[86]: array([[17,  6],
                [ 2, 17],
                [17,  1],
                [ 7, 16]])
```

```
In [88]: numpy.random.shuffle(_ra) # Un entero aleatorio hasta el numero 10
         _ra
```

```
Out[88]: array([[17,  1],
                [17,  6],
                [ 2, 17],
                [ 7, 16]])
```

```
In [89]: numpy.arange(1,10,0.5)
```

```
Out[89]: array([ 1. ,  1.5,  2. ,  2.5,  3. ,  3.5,  4. ,  4.5,  5. ,  5.5,  6. ,
                6.5,  7. ,  7.5,  8. ,  8.5,  9. ,  9.5])
```

6. MANEJO DE LIBRERIAS , ARCHIVOS, PROCESAMIENTO Y PLOTTING

Librería Pandas:

- Es usado para realizar análisis de datos. Contiene las siguientes estructuras:
 - a) Series
 - b) Dataframe

- `read_csv`
- `read_excel`
- `read_hdf`
- `read_sql`
- `read_json`
- `read_msgpack (experimental)`
- `read_html`
- `read_gbq (experimental)`
- `read_stata`
- `read_sas`
- `read_clipboard`
- `read_pickle`

6. MANEJO DE LIBRERIAS , ARCHIVOS, PROCESAMIENTO Y PLOTTING

Librería Pandas: tiene los siguientes métodos

Code:

df.describe() - Summary statistics for numerical columns

df.mean() - Returns the mean of all columns

df.corr() - Returns the correlation between columns in a DataFrame

df.count() - Returns the number of non-null values in each DataFrame column

df.max() - Returns the highest value in each column

df.min() - Returns the lowest value in each column

df.median() - Returns the median of each column

df.std() - Returns the standard deviation of each column

6. MANEJO DE LIBRERIAS , ARCHIVOS, PROCESAMIENTO Y PLOTTING

Librería Pandas:

Code:

```
In [53]: import pandas as pd
```

```
In [54]: dfCsv=pd.read_csv("C:\\Users\\epinedac\\salario.csv")
dfTxt=pd.read_csv("C:\\Users\\epinedac\\salario.txt",sep=',')
```

```
In [55]: print(dfCsv.head(2))
print("-"*100)
print(type(dfCsv))
print("-"*100)
print(dfCsv.shape)
print("-"*100)
print(dfCsv.columns)
print("-"*100)
print(dfCsv.dtypes)
print("-"*100)
salary_df = dfCsv['salary']
type(salary_df)
```

	rank	discipline	yrs.since.phd	yrs.service	sex	salary
0	Prof	B	19	18	Male	139750
1	Prof	B	20	16	Male	173200

```
<class 'pandas.core.frame.DataFrame'>
```

```
(397, 6)
```

6. MANEJO DE LIBRERIAS , ARCHIVOS, PROCESAMIENTO Y PLOTTING

Librería Pandas:

Merge method	SQL Join Name	Description
left	LEFT OUTER JOIN	Use keys from left frame only
right	RIGHT OUTER JOIN	Use keys from right frame only
outer	FULL OUTER JOIN	Use union of keys from both frames
inner	INNER JOIN	Use intersection of keys from both frames

Code:

```
In [44]: result = pd.merge(left, right, how='left', on=['key1', 'key2'])
```

left					right					Result						
	A	B	key1	key2		C	D	key1	key2		A	B	key1	key2	C	D
0	A0	B0	K0	K0	0	C0	D0	K0	K0	0	A0	B0	K0	K0	C0	D0
1	A1	B1	K0	K1	1	C1	D1	K1	K0	1	A1	B1	K0	K1	NaN	NaN
2	A2	B2	K1	K0	2	C2	D2	K1	K0	2	A2	B2	K1	K0	C1	D1
3	A3	B3	K2	K1	3	C3	D3	K2	K0	3	A2	B2	K1	K0	C2	D2
										4	A3	B3	K2	K1	NaN	NaN

6. MANEJO DE LIBRERIAS , ARCHIVOS, PROCESAMIENTO Y PLOTTING

Librería BeautifulSoup:

- Es usada para realizar web scraping, es decir extraer datos de paginas web. Se recomienda usarla en marketing de contenidos, ganar visibilidad en redes sociales, controlar la imagen y visibilidad de la misma en internet, etc.

Code:

```
In [6]: from bs4 import BeautifulSoup
        from urllib import FancyURLopener

        import sys
        from imp import reload
        reload(sys)
        sys.setdefaultencoding('utf-8')

        def get_soup(url):
            return BeautifulSoup(myopener.open(url))

        class MyOpener(FancyURLopener):
            version = 'Mozilla/5.0 (Windows; U; Windows NT 5.1; it; rv:1.8.1.11) Gecko/20071127 Firefox/2.0.0.11'
        myopener = MyOpener()
```

```
In [7]: mainUrl = 'http://www.personasperdidas.org.ar/missing/xxxxxxxxxxxxx'
        f = open('LibroNombres.csv', 'r')
        g = open('LibroNombresout.txt', 'w')
```

```
In [8]: for element in f:
        companyUrl = mainUrl.replace('xxxxxxxxxxxxx', element)
        url = str(companyUrl)+"<["+get_soup(companyUrl).get_text()
        g.write(str(url)+'\n')
```

6. MANEJO DE LIBRERIAS , ARCHIVOS, PROCESAMIENTO Y PLOTTING

Librería Pyodbc

- Es usada para realizar conexiones ODBC en windows, linux,

Code:

```
In [1]: import pandas as pd
import numpy as np
import pyodbc
```

```
In [2]: conn_str = (
    "DRIVER={PostgreSQL};"
    "DATABASE=dbprueba;"
    "UID=epinedac;"
    "PWD=epinedac;"
    "SERVER=mydatabasesprueba.ctqd53cbomwx.us-east-2.rds.amazonaws.com;"
    "PORT=5432;"
)
cnxn = pyodbc.connect(conn_str)
query = """SELECT *
          FROM db_tablitas.base_drogas
        """
df_baseDrogas = pd.read_sql(query, con=cnxn)
cnxn.close()
```

6. MANEJO DE LIBRERIAS , ARCHIVOS, PROCESAMIENTO Y PLOTTING

Librería Plotting:

El ploteo de datos es la parte mas importante del análisis descriptivo, nos da una visión de como están distribuidos nuestros datos.

Code:

```
In [88]: import matplotlib.pyplot as plt  
         from pandas.tools.plotting import scatter_matrix
```


6. MANEJO DE LIBRERIAS , ARCHIVOS, PROCESAMIENTO Y PLOTTING

Librería Plotting:
Grafica de Líneas

Code:

```
import matplotlib.pyplot as plt

# Realizar un grafico de lineas: year en el x-axis, pop en el y-axis
plt.plot(year,pop)

plt.show()
```

6. MANEJO DE LIBRERIAS , ARCHIVOS, PROCESAMIENTO Y PLOTTING

Librería Plotting:
Grafica de Líneas

Code:

```
import matplotlib.pyplot as plt  
plt.plot(gdp_cap,life_exp)  
  
# Mostrar el gráfico  
plt.show()
```

6. MANEJO DE LIBRERIAS , ARCHIVOS, PROCESAMIENTO Y PLOTTING

Librería Plotting:
Gráfico de Dispersión

Code:

```
import matplotlib.pyplot as plt  
plt.scatter(gdp_cap, life_exp)  
plt.xscale('log')
```

```
plt.show()
```

```
# Ahora realizaremos una prueba para ver si existe una relación entre la  
población y la expectativa de vida.
```

6. MANEJO DE LIBRERIAS , ARCHIVOS, PROCESAMIENTO Y PLOTTING

Librería Plotting:

Histograma

Code:

```
import matplotlib.pyplot as plt  
plt.hist(life_exp)  
plt.show()
```

Si se desea ajustar el gráfico se puede mostrar o reducir la cantidad de bins, para esto se ajustara los datos en la función plt.hist(lista,bin_value)

6. MANEJO DE LIBRERIAS , ARCHIVOS, PROCESAMIENTO Y PLOTTING

Librería Plotting:

Si deseamos añadir mas información a nuestro gráfico tales como valores en los ejes, títulos, color, etc

Code:

```
# Cednas
xlab = 'PBI per Capita [en USD]'
ylab = 'Espectativa de vida [en años]'
title = 'Desarrollo mundial desde 2007'

# Añadir los axis labels
plt.xlabel(xlab)
plt.ylabel(ylab)

# Añadir Título
plt.title(title)
```

7. MACHINE LEARNING

