Creado por:

Isabel Maniega

-0- Importamos nuestras dependencias

In [1]:
```
# pip install pandas
```

In [3]:
```
# !pip install pandas
```

In [4]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

----------- Introducción a Python ------------------

-1- Variables en Python

In [5]:
```
info = "Hola Mundo"
info
```

Out[5]:
```
'Hola Mundo'
```

In [6]:
```
info = 'Hola Mundo'
info
```

Out[6]:
```
'Hola Mundo'
```

In [7]:
```
info = 10
info
```

Out[7]:
```
10
```

-2- Print en Python

-2.1- print con Jupyter --> No es necesario poner print

In [10]:
```
var1 = 1000
```

In [11]:
```
var1
```

Out[11]:
```
1000
```

-2.2- print con VSC --> Es necesario poner print

In [12]:
```
print(var1)
```

```
1000
```

-2.3- Formas de imprimir

In [13]:
```
x = 2
y = 3
z = y + x
z
```

Out[13]:
```
5
```

In [14]:
```
print("LA SUMA DE: ", x, "y", y, "es igual a:", z)
```

```
LA SUMA DE:  2 y 3 es igual a: 5
```

In [17]:
```
print("La suma de: " + str(x) + " + " + str(y) + " es igual a: " + str(z))
```

```
La suma de: 2 + 3 es igual a: 5
```

In [18]:
```
print(f"La suma de: {x} y {y} es igual a: {z}")
```

```
La suma de: 2 y 3 es igual a: 5
```

In [21]:
```
print("La suma de: %s y %s es igual a: %s" %(x,y,z))
```

```
La suma de: 2 y 3 es igual a: 5
```

In [22]:
```
print("La suma de: x y y es igual a: z")
```

```
La suma de: x y y es igual a: z
```

-3- String Python

In [23]:
```
info = "Hola Mundo"
info
```

Out[23]:
```
'Hola Mundo'
```

In [24]:
```
info = 'Hola Mundo'
info
```

Out[24]:
```
'Hola Mundo'
```

In [25]:
```
# Dos comillas dobles: invalid syntax
info = "Hola "Mundo""
info
```

```
  File "<ipython-input-25-ae57fe8e50e9>", line 2
    info = "Hola "Mundo""
                  ^
SyntaxError: invalid syntax
```

In [26]:
```
# Comillas dobles y comillas simples:
info = "Hola 'Mundo'"
info
```

Out[26]:
```
"Hola 'Mundo'"
```

-3.1- Concatenación

In [27]:
```
info = "Hola " + "Mundo"
info
```

Out[27]:
```
'Hola Mundo'
```

In [28]:
```
info = "Hola" + " " + "Mundo"
info
```

Out[28]:
```
'Hola Mundo'
```

In [30]:
```
type(info)
info = 10
type(info)
```

Out[30]:
```
int
```

-3.2- Métodos upper(), lower() y title()

In [1]:
```
info = "hola mundo"
info = info.title()
info
```

Out[1]:
```
'Hola Mundo'
```

In [3]:
```
info = "hola mundo".title()
info
```

Out[3]:
```
'Hola Mundo'
```

In [5]:
```
info = "hola mundo"
info = info.upper()
info
```

Out[5]:
```
'HOLA MUNDO'
```

In [6]:
```
info = info.lower()
info
```

Out[6]:
```
'hola mundo'
```

-4- Suma, restas, multiplicaciones y divisiones en Python

In [7]:
```
# SUMA
# Pondremos la asignacion de una variable y después los datos que queremos sumar con signo (+)
suma = 1 + 3
suma
```

Out[7]:
```
4
```

In [8]:
```
x = 0

x = x + 1
x
```

Out[8]:
```
1
```

In [9]:
```
# Abreviada:
x = 0
x += 1
x
```

Out[9]:
```
1
```

In [14]:
```
# RESTAS
# Pondremos la asignacion de una variable y después los datos que queremos restar con signo (-)
resta = 5 - 2
resta
```

Out[14]:
```
3
```

In [15]:
```
# MULTIPLICACIÓN
# Pondremos la asignacion de una variable y después los datos que queremos multiplicacion con signo (*)
multiplicacion = 3 * 5
multiplicacion
```

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

In [16]:
```
# DIVISIÓN
# Pondremos la asignacion de una variable y después los datos que queremos división con signo (/)
division = 15 / 5
division
```

Out[16]:
```
3.0
```

In [17]:
```
# Cociente de una división
# Pondremos la asignacion de una variable y después los datos que queremos división con signo (//)
division = 15 // 5
division
```

Out[17]:
```
3
```

In [19]:
```
# Resto de una división
# Pondremos la asignacion de una variable y después los datos que queremos división con signo (%)
division = 17 % 5
division
```

Out[19]:
```
2
```

In [21]:
```
# OPERACIÓN
# Pondremos la asignacion de una variable y después los datos que queremos operar con signo
operar = 20 - 8 * 6 / 3 + 10
operar
```

Out[21]:
```
14.0
```

In [22]:
```
# OPERACIÓN
# ASIGNACIÓN DE PRIORIDAD: paréntesis
operar = (20 - 8) * 6 / 3 + 10
operar
```

Out[22]:
```
34.0
```

-4.1- Exponente

In [23]:
```
# Exponente
# Pondremos la asignacion de una variable y después el dato elevado (**) al valor
elevado = 20 ** 3
elevado
```

Out[23]:
```
8000
```

-4.2- Decimales

In [24]:
```
# En el caso de los decimales se pone (.), NUNCA (,)
number = 2,4
number
```

Out[24]:
```
(2, 4)
```

In [25]:
```
type(number)
```

Out[25]:
```
tuple
```

In [26]:
```
# En el caso de los decimales se pone (.), NUNCA (,)
number = 2.4
number
```

Out[26]:
```
2.4
```

In [27]:
```
type(number)
```

Out[27]:
```
float
```

In [28]:
```
# Si sólo ponemos punto lo interpreta como 0.5
number = .5
number
```

Out[28]:
```
0.5
```

In [29]:
```
# Redondear: round(numero, numero de decimales)
number = round(0.3555, 2)
number
```

Out[29]:
```
0.36
```

-4.3- Tipos de datos

In [30]:
```
number = 0.2365
type(number)
```

Out[30]:
```
float
```

In [31]:
```
number = 25
type(number)
```

Out[31]:
```
int
```

In [32]:
```
text = "Hola Mundo"
type(text)
```

Out[32]:
```
str
```

-4.4- Max, Min, Absoluto, suma

In [33]:
```
# Valor absoluto
absoluto = abs(-6)
absoluto
```

Out[33]:
```
6
```

In [34]:
```
# Maximo de una serie de números
maximo = max(6, -3, 8.56, -40, 25)
maximo
```

Out[34]:
```
25
```

In [35]:
```
# Minimo de una serie de números
minimo = min(6, -3, 8.56, -40, 25)
minimo
```

Out[35]:
```
-40
```

In [36]:
```
suma_lista = sum([2, 2, 6])
suma_lista
```

Out[36]:
```
10
```

-5- Estructura de datos Básicos

-5.1- Tuplas

In [1]:
```
# Tuplas o arrays
A = (10, 20, 30, 40) # 0,1,2,3
A
```

Out[1]:
```
(10, 20, 30, 40)
```

In [2]:
```
A[0]
```

Out[2]:
```
10
```

In [3]:
```
A[1]
```

Out[3]:
```
20
```

In [4]:
```
# Imprimir todos juntos
A[0], A[1], A[2], A[3]
```

Out[4]:
```
(10, 20, 30, 40)
```

In [5]:
```
A
```

Out[5]:
```
(10, 20, 30, 40)
```

In [6]:
```
# Última posición
A[-1]
```

Out[6]:
```
40
```

¿Apendizar en tuplas?

In [7]:
```
# Listado2.append(numero)
# Listado2 --> es el nombre de la lista la cual quiero apendizar
# .append(numero) --> Añadir un valor en la serie de numeros
A.append(50)
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-7-78bd40e9ad25> in <module>
      2 # Listado2 --> es el nombre de la lista la cual quiero apendizar
      3 # .append(numero) --> Añadir un valor en la serie de numeros
----> 4 A.append(50)

AttributeError: 'tuple' object has no attribute 'append'
```

No es posible!!!

Modificar valores en la tupla

In [8]:
```
# Listado[0] = 200
A[0] = 200
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-8-e08f3a4b0813> in <module>
      1 # Listado[0] = 200
----> 2 A[0] = 200

TypeError: 'tuple' object does not support item assignment
```

No nos permite!!!

Conclusión:

- Las tuplas no permiten apendizar elementos
- Las tuplas nos permiten modificar elementos

-5.2- Arrays

In [10]:
```
import numpy as np
```

In [11]:
```
B = np.array([10, 20, 30, 40])
B
```

Out[11]:
```
array([10, 20, 30, 40])
```

In [12]:
```
B[0]
```

Out[12]:
```
10
```

In [13]:
```
B[-1]
```

Out[13]:
```
40
```

In [14]:
```
B
```

Out[14]:
```
array([10, 20, 30, 40])
```

Transformar a lista apartir de un np.array()

NombreArray.tolist()

In [15]:
```
Listado_B = B.tolist()
Listado_B
```

Out[15]:
```
[10, 20, 30, 40]
```

lista a array (nuevamente)

In [16]:
```
array_listado_B = np.array(Listado_B)
array_listado_B
```

Out[16]:
```
array([10, 20, 30, 40])
```

¿Es posible apendizar elementos a un np.array?

In [17]:
```
B.append(50) #--> No Funciona
B
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-17-f12a63196591> in <module>
----> 1 B.append(50)
      2 B

AttributeError: 'numpy.ndarray' object has no attribute 'append'
```

In [18]:
```
B
```

Out[18]:
```
array([10, 20, 30, 40])
```

In [19]:
```
c = np.append(B, 50)
c
```

Out[19]:
```
array([10, 20, 30, 40, 50])
```

Si es posible pero usando la librería numpy (np.append)

¿Es posible modificar elementos a un np.array?

In [20]:
```
B[0] = 100
B
```

Out[20]:
```
array([100,  20,  30,  40])
```

In [21]:
```
c[0], c[-1]
```

Out[21]:
```
(10, 50)
```

Si es posible modificar datos

-5.3- Listas

In [22]:
```
C = [10, 20, 30, 40]
C
```

Out[22]:
```
[10, 20, 30, 40]
```

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

Out[23]:
```
list
```

In [24]:
```
B = list((10, 20, 30, 40))
B
```

Out[24]:
```
[10, 20, 30, 40]
```

In [25]:
```
type(B)
```

Out[25]:
```
list
```

In [26]:
```
B[2]
```

Out[26]:
```
30
```

¿Es posible apendizar elementos a una lista?

In [27]:
```
B.append(100)
B
```

Out[27]:
```
[10, 20, 30, 40, 100]
```

¿Es posible modificar elementos a una lista?

In [28]:
```
B[0] = 500
B
```

Out[28]:
```
[500, 20, 30, 40, 100]
```

Conclusión:

- Es posible apendizar
- Es posible modificar

-5.4- Mínimos y Máximos en estrtuctura de datos

In [29]:
```
listado = [300, 100, 700, 400]
listado
```

Out[29]:
```
[300, 100, 700, 400]
```

In [30]:
```
min(listado)
```

Out[30]:
```
100
```

In [31]:
```
max(listado)
```

Out[31]:
```
700
```

-5.5- Recomendaciones

In [32]:
```
L = [120, 230, 340, 400, 450, 500, 550, 600, 650, 700, 750, 800]
L
```

Out[32]:
```
[120, 230, 340, 400, 450, 500, 550, 600, 650, 700, 750, 800]
```

In [33]:
```
# Pero mejor de esta forma...
```

In [34]:
```
L = [
     120, 230, 340,
     400, 450, 500,
     550, 600, 650,
     700, 750, 800
]
# mejor
L
```

Out[34]:
```
[120, 230, 340, 400, 450, 500, 550, 600, 650, 700, 750, 800]
```

In [35]:
```
# Matrices con Numpy
a = np.array([
     [120, 230, 340],
     [400, 450, 500],
     [550, 600, 650],
     [700, 750, 800]])
a
```

Out[35]:
```
array([[120, 230, 340],
       [400, 450, 500],
       [550, 600, 650],
       [700, 750, 800]])
```

-5.6- Dataframes

In [36]:
```
# Usamos comillas dobles para los nombres de los estudiantes (E)
E = ["Andres", "Marcos", "Eva", "María"]
E
```

Out[36]:
```
['Andres', 'Marcos', 'Eva', 'María']
```

In [37]:
```
# Notas de los exámenes (N), de 0 a 10, siendo 10 la nota más alta
N = [9, 7, 8, 6]
N
```

Out[37]:
```
[9, 7, 8, 6]
```

In [38]:
```
# Edades de cada uno de los alumnos (M)
M = [21, 23, 25, 27]
M
```

Out[38]:
```
[21, 23, 25, 27]
```

In [40]:
```
# pip install pandas
```

In [41]:
```
import pandas as pd
```

In [42]:
```
# Crear el dataframe con pandas
df = pd.DataFrame(E, columns=["Estudiante"])
df
```

Out[42]:

	Estudiante
0	Andres
1	Marcos
2	Eva
3	María

In [43]:
```
df['Notas'] = N
df
```

Out[43]:

	Estudiante	Notas
0	Andres	9
1	Marcos	7
2	Eva	8
3	María	6

In [44]:
```
df['Edad'] = M
df
```

Out[44]:

	Estudiante	Notas	Edad
0	Andres	9	21
1	Marcos	7	23
2	Eva	8	25
3	María	6	27

In [45]:
```
df.head(2)
```

Out[45]:

	Estudiante	Notas	Edad
0	Andres	9	21
1	Marcos	7	23

In [46]:
```
df.tail(2)
```

Out[46]:

	Estudiante	Notas	Edad
2	Eva	8	25
3	María	6	27

In [47]:
```
df.Notas
```

Out[47]:
```
0    9
1    7
2    8
3    6
Name: Notas, dtype: int64
```

In [48]:
```
df['Notas']
```

Out[48]:
```
0    9
1    7
2    8
3    6
Name: Notas, dtype: int64
```

In [49]:
```
df[['Notas']]
```

Out[49]:

	Notas
0	9
1	7
2	8
3	6

Creado por:

Isabel Maniega