2_Primeros_Pasos

June 18, 2025

Creado por:

Isabel Maniega

1 -0- Importamos nuestras dependencias

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

[2]:
```
# !pip install pandas
```

[3]:
```
# pip install matplotlib
```

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

2 ———— Introducción a Python ————————-

2.1 -1- Variables en Python

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

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

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

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

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

[7]:
```
10
```

2.2 -2- Print en Python

2.2.1 -2.1- print con Jupyter –> No es necesario poner print

[8]:
```
var1 = 1000
```

[9]:
```
var1
```

[9]: 1000

2.2.2 -2.2- print con VSC –> Es necesario poner print

[10]:
```
print(var1)
```

```
1000
```

2.2.3 -2.3- Formas de imprimir

[11]:
```
x = 2
y = 3
z = y + x
z
```

[11]: 5

[12]:
```
print("LA SUMA DE: ", x, "y", y, "es igual a:", z)
```

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

[13]:
```
print("La suma de: " + str(x) + " + " + str(y) + " es igual a: " + str(z))
```

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

[14]:
```
print(f"La suma de: {x} y {y} es igual a: {z}")
```

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

[15]:
```
print("La suma de: %s y %s es igual a: %s" %(x,y,z))
```

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

[16]:
```
print("La suma de: x y y es igual a: z")
```

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

Signo Scape (”) + newline (n)

[17]:
```
print("Hola \nMundo")
```

```
Hola
Mundo
```

[18]:
```
print("Hola")
print("\\")
print("Mundo")
```

```
Hola
\
Mundo
```

[19]:
```
print("Hola")
print("\")
print("Mundo")
```

```
  Cell In[19], line 2
    print("\")
          ^
SyntaxError: unterminated string literal (detected at line 2)
```

[20]:
```
print(f"La suma de: {x} y {y}", end=" ")
print(f"es igual a: {z}")
```

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

[21]:
```
print(f"La suma de: {x} y {y} ", f" es igual a: {z}", sep="-")
```

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

[22]:
```
print("Hola", "Mundo", "PCEP", sep='-')
```

```
Hola-Mundo-PCEP
```

[23]:
```
print("Hola Mundo", "PCEP", sep='-')
```

```
Hola Mundo-PCEP
```

[24]:
```
print(1, 2, 3, sep='*')
```

```
1*2*3
```

2.3 -3- String Python

[25]:
```
info = "Hola Mundo"
info
```

[25]:
```
'Hola Mundo'
```

[26]:
```
info = 'Hola Mundo'
info
```

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

[27]:
```
# Dos comillas dobles: invalid syntax
info = "Hola "Mundo""
info
```

```
  Cell In[27], line 2
    info = "Hola "Mundo""
                  ^
SyntaxError: invalid syntax
```

[28]:
```
# Comillas dobles y comillas simples:
info = "Hola 'Mundo'"
info
```

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

[29]:
```
info = "Hola \"Mundo\""
info
```

[29]:
```
'Hola "Mundo"'
```

2.3.1 -3.1- Concatenación

[30]:
```
info = "Hola " + "Mundo"
info
```

[30]:
```
'Hola Mundo'
```

[31]:
```
info = "Hola" + " " + "Mundo"
info
```

[31]:
```
'Hola Mundo'
```

[32]:
```
print(type(info))
info = 10
print(type(info))
```

```
<class 'str'>
<class 'int'>
```

2.3.2 -3.2- Métodos upper(), lower() y title()

[33]:
```
info = "hola mundo"
info = info.title()
info
```

[33]: 'Hola Mundo'

[34]:
```
info = "hola mundo".title()
info
```

[34]: 'Hola Mundo'

[35]:
```
info = "hola mundo"
info = info.upper()
info
```

[35]: 'HOLA MUNDO'

[36]:
```
info = info.lower()
info
```

[36]: 'hola mundo'

2.4 -4- Suma, restas, multiplicaciones y divisiones en Python

[37]:
```
# SUMA
# Pondremos la asignacion de una variable y después los datos que queremos␣
 ↪sumar con signo (+)
suma = 1 + 3
suma
```

[37]: 4

[38]:
```
x = 0

x = x + 1
x
```

[38]: 1

[39]:
```
# Abreviada:
x = 0
x += 1
x
```

[39]: 1

[40]:
```
# RESTAS
# Pondremos la asignacion de una variable y después los datos que queremos␣
 ↪restar con signo (-)
resta = 5 - 2
resta
```

[40]: 3

[41]:
```
# MULTIPLICACIÓN
# Pondremos la asignacion de una variable y después los datos que queremos␣
 ↪multiplicacion con signo (*)
multiplicacion = 3 * 5
multiplicacion
```

[41]: 15

[42]:
```
# DIVISIÓN
# Pondremos la asignacion de una variable y después los datos que queremos␣
 ↪división con signo (/)
division = 15 / 5
division
```

[42]: 3.0

[43]:
```
# Cociente de una división
# Pondremos la asignacion de una variable y después los datos que queremos␣
 ↪división con signo (//)
division = 15 // 5
division
```

[43]: 3

[44]:
```
# Resto de una división
# Pondremos la asignacion de una variable y después los datos que queremos␣
 ↪división con signo (%)
division = 17 % 5
division
```

[44]: 2

[45]:
```
# OPERACIÓN
# Pondremos la asignacion de una variable y después los datos que queremos␣
 ↪operar con signo
operar = 20 - 8 * 6 / 3 + 10
operar
```

[45]: 14.0

[46]:
```
# OPERACIÓN
# ASIGNACIÓN DE PRIORIDAD: paréntesis
operar = (20 - 8) * 6 / 3 + 10
operar
```

[46]: 34.0

2.4.1 -4.1- Exponente

[47]:
```
# Exponente
# Pondremos la asignacion de una variable y después el dato elevado (**) al␣
 ↪valor
elevado = 20 ** 3
elevado
```

[47]:
```
8000
```

2.4.2 -4.2- Decimales

[48]:
```
# En el caso de los decimales se pone (.), NUNCA (,)
number = 2,4
number
```

[48]:
```
(2, 4)
```

[49]:
```
type(number)
```

[49]:
```
tuple
```

[50]:
```
# En el caso de los decimales se pone (.), NUNCA (,)
number = 2.4
number
```

[50]:
```
2.4
```

[51]:
```
type(number)
```

[51]:
```
float
```

[52]:
```
# Si sólo ponemos punto lo interpreta como 0.5
number = .5
number
```

[52]:
```
0.5
```

[53]:
```
# Redondear: round(numero, numero de decimales)
number = round(0.3555, 2)
number
```

[53]:
```
0.36
```

2.4.3 -4.3- Tipos de datos

[54]:
```
number = 0.2365
type(number)
```

[54]:
```
float
```

[55]:
```
number = 25
type(number)
```

[55]:
```
int
```

[56]:
```
text = "Hola Mundo"
type(text)
```

[56]:
```
str
```

2.4.4 -4.4- Max, Min, Absoluto, suma

[57]:
```
# Valor absoluto
absoluto = abs(-6)
absoluto
```

[57]:
```
6
```

[58]:
```
# Maximo de una serie de números
maximo = max(6, -3, 8.56, -40, 25)
maximo
```

[58]:
```
25
```

[59]:
```
# Mínimo de una serie de números
minimo = min(6, -3, 8.56, -40, 25)
minimo
```

[59]:
```
-40
```

[60]:
```
suma_lista = sum([2, 2, 6])
suma_lista
```

[60]:
```
10
```

3 -5- Estructura de datos Básicos

3.0.1 -5.1- Tuplas

[61]:
```
# Tuplas o arrays
A = (10, 20, 30, 40) # 0,1,2,3
A
```

[61]: (10, 20, 30, 40)

[62]:
```
A[0]
```

[62]: 10

[63]:
```
A[1]
```

[63]: 20

[64]:
```
# Imprimir todos juntos
A[0], A[1], A[2], A[3]
```

[64]: (10, 20, 30, 40)

[65]:
```
A
```

[65]: (10, 20, 30, 40)

[66]:
```
# última posición
A[-1]
```

[66]: 40

¿Apendizar en tuplas?

[67]:
```
# 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)
Cell In[67], line 4
      1 # Listado2.append(numero)
      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

[68]:
```
# Listado[0] = 200
A[0] = 200
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[68], line 2
      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 no permiten modificar elementos

3.0.2 -5.2- Arrays

[69]:
```
import numpy as np
```

[70]:
```
B = np.array([10, 20, 30, 40])
B
```

[70]:
```
array([10, 20, 30, 40])
```

[71]:
```
B[0]
```

[71]:
```
np.int64(10)
```

[72]:
```
B
```

[72]:
```
array([10, 20, 30, 40])
```

[73]:
```
B[-1]
```

[73]:
```
np.int64(40)
```

Transformar a lista apartir de un np.array()

NombreArray.tolist()

[74]:
```
Listado_B = B.tolist()
Listado_B
```

[74]:
```
[10, 20, 30, 40]
```

lista a array (nuevamente)

[75]:
```
array_listado_B = np.array(Listado_B)
array_listado_B
```

[75]:
```
array([10, 20, 30, 40])
```

¿Es posible apendizar elementos a un np.array?

[76]:
```
B.append(50) #--> No Funciona
B
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[76], line 1
----> 1 B.append(50) #--> No Funciona
      2 B

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

[77]:
```
B
```

[77]:
```
array([10, 20, 30, 40])
```

[78]:
```
c = np.append(B, 50)
c
```

[78]:
```
array([10, 20, 30, 40, 50])
```

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

¿Es posible modificar elementos a un np.array?

[79]:
```
B[0] = 100
B
```

[79]:
```
array([100,  20,  30,  40])
```

[80]:
```
c[0], c[-1]
```

[80]:
```
(np.int64(10), np.int64(50))
```

Si es posible modificar datos

3.0.3 -5.3- Listas

[81]:
```
C = [10, 20, 30, 40]
C
```

[81]: [10, 20, 30, 40]

[82]:
```
type(C)
```

[82]: list

[83]:
```
test = (10, 20, 30, 40)
list_test = list(test)
list_test
```

[83]: [10, 20, 30, 40]

[84]:
```
B = list((10, 20, 30, 40))
B
```

[84]: [10, 20, 30, 40]

[85]:
```
type(B)
```

[85]: list

[86]:
```
B[2]
```

[86]: 30

¿Es posible apendizar elementos a una lista?

[87]:
```
B.append(100)
B
```

[87]: [10, 20, 30, 40, 100]

¿Es posible modificar elementos a una lista?

[88]:
```
B[0] = 500
B
```

[88]: [500, 20, 30, 40, 100]

Conclusión: * Es posible apendizar * Es posible modificar

3.0.4 -5.4- Mínimos y Máximos en estructura de datos

[89]:
```
listado = [300, 100, 700, 400]
listado
```

[89]: [300, 100, 700, 400]

[90]:
```
min(listado)
```

[90]: 100

[91]:
```
max(listado)
```

[91]: 700

3.0.5 -5.5- Recomendaciones

[92]:
```
L = [120, 230, 340, 400, 450, 500, 550, 600, 650, 700, 750, 800]
L
```

[92]: [120, 230, 340, 400, 450, 500, 550, 600, 650, 700, 750, 800]

[93]:
```
# Pero mejor de esta forma...
```

[94]:
```
L = [
     120, 230, 340,
     400, 450, 500,
     550, 600, 650,
     700, 750, 800
]
# mejor
L
```

[94]: [120, 230, 340, 400, 450, 500, 550, 600, 650, 700, 750, 800]

[95]:
```
# Matrices con Numpy
a = np.array([
     [120, 230, 340],
     [400, 450, 500],
     [550, 600, 650],
     [700, 750, 800]])
a
```

[95]:
```
array([[120, 230, 340],
       [400, 450, 500],
       [550, 600, 650],
       [700, 750, 800]])
```

3.0.6 -5.6- Dataframes

[96]:
```
# Usamos comillas dobles para los nombres de los estudiantes (E)
E = ["Andres", "Marcos", "Eva", "María"]
E
```

[96]: ['Andres', 'Marcos', 'Eva', 'María']

[97]:
```
# Notas de los exámenes (N), de 0 a 10, siendo 10 la nota más alta
N = [9, 7, 8, 6]
N
```

[97]:
```
[9, 7, 8, 6]
```

[98]:
```
# Edades de cada uno de los alumnos (M)
M = [21, 23, 25, 27]
M
```

[98]:
```
[21, 23, 25, 27]
```

[99]:
```
# pip install pandas
```

[100]:
```
import pandas as pd
```

[101]:
```
# Crear el dataframe con pandas
df = pd.DataFrame(E, columns=["Estudiante"])
df
```

[101]:

	Estudiante
0	Andres
1	Marcos
2	Eva
3	María

[102]:
```
df['Notas'] = N
df
```

[102]:

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

[103]:
```
df['Edad'] = M
df
```

[103]:

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

[104]:
```
df.head(2)
```

[104]:

	Estudiante	Notas	Edad
0	Andres	9	21

```
1     Marcos      7    23
```

[105]:
```
df.tail(2)
```

[105]:
```
  Estudiante  Notas  Edad
2        Eva      8    25
3      María      6    27
```

[106]:
```
df.Notas
```

[106]:
```
0    9
1    7
2    8
3    6
Name: Notas, dtype: int64
```

[107]:
```
df['Notas']
```

[107]:
```
0    9
1    7
2    8
3    6
Name: Notas, dtype: int64
```

[108]:
```
df[['Notas']]
```

[108]:
```
   Notas
0      9
1      7
2      8
3      6
```

4 Enteros: números octales y hexadecimales

Existen dos convenciones adicionales en Python que no son conocidas en el mundo de las matemáticas. El primero nos permite utilizar un número en su representación octal.

Si un número entero esta precedido por un código 0O o 0o (cero-o), el número será tratado como un valor octal. Esto significa que el número debe contener dígitos en el rango del [0..7] únicamente.

0o123 es un número octal con un valor (decimal) igual a 83.

[109]:
```
print(0o123)
```

```
83
```

La segunda convención nos permite utilizar números en hexadecimal. Dichos números deben ser precedidos por el prefijo 0x o 0X (cero-x).

0x123 es un número hexadecimal con un valor (decimal) igual a 291.

[110]:
```
print(0x123)
```

```
291
```

Creado por:

Isabel Maniega