

Nombre de la práctica	Numpy			No.	1
Asignatura:	Simulación	Carrera:	INGENIERÍA EN SISTEMAS COMPUTACIONALES	Duración de la práctica (Hrs)	

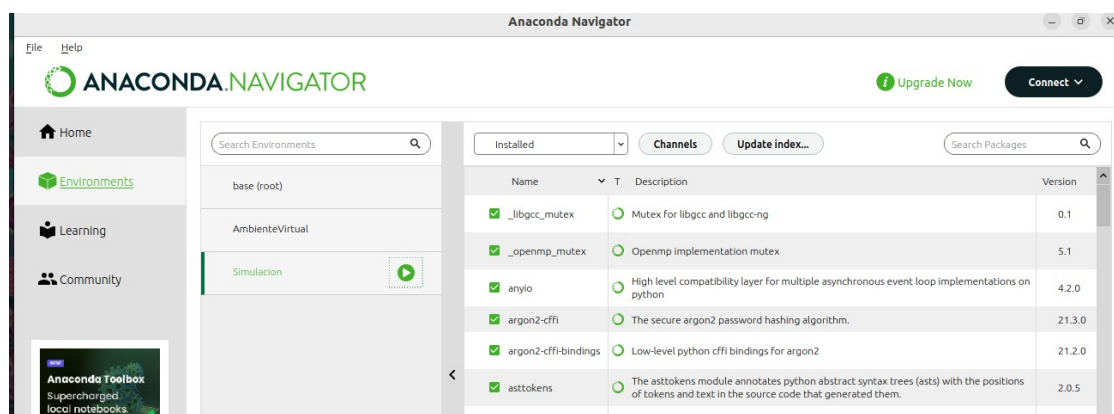
NOMBRE DEL ALUMNO: Fabiola Castañeda Mondragón
GRUPO: 3501

1.- Abrir Anaconda, desde la terminal con el comando **anaconda-navigator**:

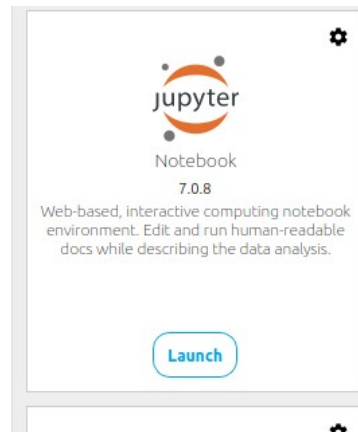
```
fabiola2004@pc8: ~
(base) fabiola2004@pc8:~$ anaconda-navigator
2024-09-11 16:24:18,437 - WARNING linux_scaling.get_primary_monitor_name:61
Can't detect primary monitor.

Warning: Ignoring XDG_SESSION_TYPE=wayland on Gnome. Use QT_QPA_PLATFORM=wayland
to run on Wayland anyway.
libGL error: MESA-LOADER: failed to open iris: /usr/lib/dri/iris_dri.so: no se p
uede abrir el archivo del objeto compartido: No existe el archivo o el directori
o (search paths /usr/lib/x86_64-linux-gnu/dri:\${ORIGIN}/dri:/usr/lib/dri, suff
ix _dri)
libGL error: failed to load driver: iris
libGL error: MESA-LOADER: failed to open swrast: /usr/lib/dri/swrast_dri.so: no
se puede abrir el archivo del objeto compartido: No existe el archivo o el direc
torio (search paths /usr/lib/x86_64-linux-gnu/dri:\${ORIGIN}/dri:/usr/lib/dri,
suffix _dri)
libGL error: failed to load driver: swrast
```

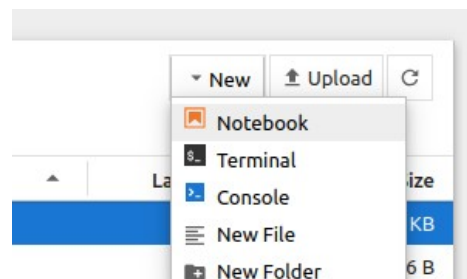
2.- Correr nuestro ambiente virtual:



3.- Abrimos Jupyter:



4.- Creamos un nuevo proyecto:



5.-Escribimos una breve Introducción a NumPy. Es una biblioteca de Python para computación científica. Ofrece:

- Arrays N-dimensionales.
- Funciones matemáticas avanzadas.
- Integración con C/C++ y Fortran.
- Herramientas para álgebra lineal y números aleatorios.

▼ Introducción a Numpy

Numpy es una librería para la computación con python.

- Proporciona Arrays N-Dimensionales.
- Implementa funciones matemáticas sofisticadas
- proporciona herramientas para integrar C/C++ y Fortran.
- Proporciona mecanismos para facilitar la realización de las tareas relacionadas con álgebra lineal o números aleatorios

6.- La línea `import numpy as np` importa la biblioteca **NumPy** y le asigna el alias **np** para simplificar su uso en el código.

Imports

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

7.- Un **array** es una estructura de datos con múltiples dimensiones. En **NumPy**:

- **Axis**: Cada dimensión.
- **Rank**: Número de dimensiones.
- **Shape**: Tamaño de cada dimensión.
- **Size**: Total de elementos.

El ejemplo muestra un array de ceros con forma (2, 4), rank 2, y tamaño 8.

Arrays

Un **array** es una estructura de datos que consiste en una colección de elementos (valores o variables), cada uno identificado por al menos un índice o clave. Un array se almacena de modo que la posición de cada elemento se pueda calcular a partir de su tupla de índices, mediante una fórmula matemática. El tipo más simple de array es un array lineal también llamado array unidimensional

En Numpy:

- Cada dimensión se denomina **axis**
- El número de dimensiones se denomina **rank**
- La lista de dimensiones con su correspondiente longitud se denomina **shape**
- El número total de elementos (multiplicación de la longitud de las dimensiones) a esto se denomina **size**

```
] # Array cuyos valores son todos 0.  
a= np.zeros((2, 4))  
a
```

```
] array([[0., 0., 0., 0.],  
        [0., 0., 0., 0.]])
```

a es una array:

- con dos **axis**, el primero de longitud 2 y el segundo de longitud 4.
- Con un **rank** igual a 2
- Con un **shape** igual a (2,4)
- Con un **size** igual a 8

```
] a.shape
```

```
] (2, 4)
```

```
] a.ndim
```

```
] 2
```

```
] a.size
```

```
] 8
```

8.-El código crea un array tridimensional de forma (2, 3, 4) donde todos los valores son 0, usando `np.zeros()`.

Creacion de Arrays

```
[6]: # Array cuyos valores son todos 0
np.zeros((2,3,4))

[6]: array([[[0., 0., 0., 0.],
           [0., 0., 0., 0.],
           [0., 0., 0., 0.]],

          [[0., 0., 0., 0.],
           [0., 0., 0., 0.],
           [0., 0., 0., 0.]])

[7]: # Array cuyos valores son todos 1
np.ones((2,3,4))

[7]: array([[[1., 1., 1., 1.],
           [1., 1., 1., 1.],
           [1., 1., 1., 1.]],

          [[1., 1., 1., 1.],
           [1., 1., 1., 1.],
           [1., 1., 1., 1.]])

[8]: # Array cuyos valores son todos el valor indicado como segundo parametro de la funcion
np.full((2,3,4),8)

[8]: array([[[8, 8, 8, 8],
           [8, 8, 8, 8],
           [8, 8, 8, 8]],

          [[8, 8, 8, 8],
           [8, 8, 8, 8],
           [8, 8, 8, 8]])

[9]: #El resultado de np.empty no es predecible
#Se inicializa con los valores del array con lo que haya en memoria en ese momento
np.empty((2, 3, 9))

[9]: array([[[ 1.94399274e-316,  0.00000000e+000,  6.85254118e-310,
               8.35670205e+049,  6.85254117e-310,  6.85254118e-310,
               4.61259571e+070,  6.85254620e-310,  6.85254118e-310],
            [ 3.08952169e-067,  6.85254117e-310,  6.85254118e-310,
               1.55884763e-302,  6.85254620e-310,  6.85254118e-310,
               4.36862299e+021,  6.85254117e-310,  6.85254118e-310],
            [ 9.60341413e-229,  6.85254135e-310,  6.85254118e-310,
               -3.92539080e-088,  6.85254117e-310,  6.85254118e-310,
               -3.87695570e+037,  6.85254135e-310,  6.85254118e-310]],

           [[ 1.81222822e-006,  6.85254117e-310,  6.85254118e-310,
               2.87096774e-100,  6.85254135e-310,  6.85254118e-310,
               -8.29772860e+300,  6.85254117e-310,  6.85254118e-310],
            [ 9.69380057e+053,  6.85254135e-310,  6.85254118e-310,
               -6.41125734e-215,  6.85254118e-310,  6.85254118e-310,
               1.53373348e-146,  6.85254118e-310,  6.85254118e-310],
            [ 4.00193173e-322,  2.26028709e-316,  6.85254806e-310,
               6.85254809e-310,  6.85254545e-310,  6.85254808e-310,
               6.85254809e-310,  6.85254119e-310,  6.85254117e-310]])

[10]: #Inicializar el array utilizando un array de Python
b = np.array([ [ 1,2,3], [ 4,5,6]])
b

[10]: array([[1, 2, 3],
           [4, 5, 6]])

[11]: b.shape

[11]: (2, 3)

[13]: #Crear un array utilizando una funcion basada en rangos
#/minimo, maximo, número elementos del array)
print(np.linspace(0,6,10))

[0.      0.66666667  1.33333333  2.      2.66666667  3.33333333
 4.      4.66666667  5.33333333  6.]

[14]: #Inicializar el array con valores aleatorios
np.random.rand(2,3,4)

[14]: array([[[2.40804605e-01, 2.92859601e-02, 8.79872983e-01, 6.01951262e-01],
           [5.67796290e-01, 5.40387906e-02, 7.25702918e-01, 6.07700992e-01],
           [1.46795525e-01, 1.46915799e-02, 7.05987127e-01, 4.06395274e-01]],

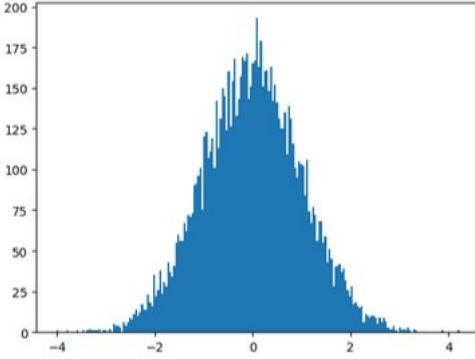
          [[2.73635274e-01, 5.52905867e-01, 3.08555844e-04, 5.79058321e-01],
           [5.64660051e-01, 8.11555217e-01, 6.08563506e-02, 5.45983919e-01],
           [5.32847879e-01, 6.15724386e-01, 9.62794589e-01, 3.33333167e-01]])

[15]: #Iniciar array con valores aleatorios conforme a una distribución normal.
np.random.randn(2,4)
```

9.- El código genera números aleatorios y muestra su distribución en un histograma con 200 barras. La forma es una campana (distribución normal).

```
[18]: %matplotlib inline
import matplotlib.pyplot as plt

c = np.random.randn(10000)
plt.hist(c, bins=200)
plt.show()
```



```
[19]: #Inicializar un array, utilizando una función personalizada

def func(x,y):
    return x + 2 * y

np.fromfunction(func, (3,5))

[19]: array([[ 0.,  2.,  4.,  6.,  8.],
           [ 1.,  3.,  5.,  7.,  9.],
           [ 2.,  4.,  6.,  8., 10.]])
```

10.- El código crea un array unidimensional con np.array. Luego, imprime su forma (shape) y los elementos del array.

Acceso a los elementos de un array

Array Unidimensional

```
[21]: #Acceder a los elementos de un array.
array_uni = np.array([1,3,5,7,9,11])
print("Shape:", array_uni.shape)
print("Array_uni", array_uni)

Shape: (6,)
Array_uni [ 1  3  5  7  9 11]

[22]: #Accediendo al quinto elemento del array.
array_uni[4]

[22]: np.int64(9)

[23]: #Acceder al tercer y cuarto elemento del array
array_uni[2:4]

[23]: array([5, 7])
```

11.- El código crea un array bidimensional (matriz) con `np.array`. Luego, imprime su forma (`shape`) y el contenido del array.

Array Multidimensional.

```
[25]: #Crear un array multidimensional.
array_multi = np.array([ [1,2,3,4 ],[ 5,6,7,8]])
print("Shape:", array_multi.shape)
print("Array_multi:\n", array_multi)

Shape: (2, 4)
Array_multi:
[[1 2 3 4]
 [5 6 7 8]]

[26]: #Acceder al cuarto elemento del array
array_multi[0,3]

[26]: np.int64(4)

[27]: #Acceder a la segunda fila del array
array_multi[1, :]

[27]: array([5, 6, 7, 8])

#Accediendo al primer elemento de las dos primeras filas del array array_multi[0:2, 2]
```

12.- El código crea un array unidimensional con elementos del 0 al 27 usando `np.arange(28)`. Luego, imprime su forma (`shape`) y el contenido del array.

Modificación de un array

```
[29]: # Crear un arreglo unidimensional e inicializarlo con un rango
# de elementos 0-27
array1 = np.arange(28)
print("Shape:", array1.shape)
print("Array:\n", array1)

Shape: (28,)
Array:
[ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
 24 25 26 27]

[31]: #Cambiar las dimensiones del array y sus longitudes.
array1.shape = (7, 4)
print("Shape:", array1.shape)
print("Array:\n", array1)

Shape: (7, 4)
Array:
[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]
 [12 13 14 15]
 [16 17 18 19]
 [20 21 22 23]
 [24 25 26 27]]

[32]: # El ejemplo anterior devuelve un nuevo array que apunta a los mismos datos.
# NOTA: modificaciones en el array, modificaran el otro array
array2 = array1.reshape(4, 7)
print("Shape:", array2.shape)
print("Array:\n", array2)

Shape: (4, 7)
```

```
[32]: # El ejemplo anterior devuelve un nuevo array que apunta a los mismos datos.  
# NOTA: modificaciones en el array, modificaran el otro array  
array2 = array1.reshape(4, 7)  
print("Shape:", array2.shape)  
print("Array:\n", array2)
```

```
Shape: (4, 7)  
Array:  
[[ 0  1  2  3  4  5  6]  
 [ 7  8  9 10 11 12 13]  
 [14 15 16 17 18 19 20]  
 [21 22 23 24 25 26 27]]
```

```
[33]: #Modificación del nuevo array devuelto  
array2[1, 3] = 30  
print("Shape:", array2.shape)  
print("Array2:\n", array2)
```

```
Shape: (4, 7)  
Array2:  
[[ 0  1  2  3  4  5  6]  
 [ 7  8  9 30 11 12 13]  
 [14 15 16 17 18 19 20]  
 [21 22 23 24 25 26 27]]
```

```
[34]: print("Array1:\n", array1)
```

```
Array1:  
[[ 0  1  2  3]  
 [ 4  5  6  7]  
 [ 8  9 30 11]  
 [12 13 14 15]  
 [16 17 18 19]  
 [20 21 22 23]  
 [24 25 26 27]]
```

```
[35]: #Devolver el array a su estado original  
print("Array1: ",array1.ravel())
```

```
Array1: [ 0  1  2  3  4  5  6  7  8  9 30 11 12 13 14 15 16 17 18 19 20 21 22 23  
 24 25 26 27]
```

13.- El código crea dos arrays:

1. array1 contiene números pares del 2 al 16 (np.arange(2, 18, 2)).
2. array2 contiene números del 0 al 7 (np.arange(8)). Luego, imprime ambos arrays.

Operaciones Aritméticas con Arrays

```
1: array1 = np.arange(2, 18, 2)  
array2 = np.arange(8)  
print("Array 1:", array1)  
print("Array 2:", array2)
```

```
Array 1: [ 2  4  6  8 10 12 14 16]  
Array 2: [0 1 2 3 4 5 6 7]
```

```
1: #Suma  
print(array1 + array2)  
[ 2  5  8 11 14 17 20 23]
```

```
1: #Resta  
print(array1 - array2)  
[2 3 4 5 6 7 8 9]
```

```
1: #Multiplicación  
#Nota: No es una multiplicación de matrices.  
print(array1 * array2)  
[ 0  4 12 24 40 60 84 112]
```


14.- El código crea dos arrays y muestra su forma y contenido:

1. array1 es un array unidimensional con elementos del 0 al 4 (`np.arange(5)`).
Su forma es `(5,)`.
2. array2 es un array unidimensional con un solo elemento (`np.array([3])`).
Su forma es `(1,)`.

Imprime la forma y los elementos de ambos arrays.

Broadcasting

Si se aplican operaciones aritmeticas sobre arrays que no tienen la misma forma (shape), Numpy aplica una propiedad que se llama Broadcasting.

```
[43]: array1 = np.arange(5)
      array2 = np.array([ 3])
      print("Shape:", array1.shape)
      print("Array 1:", array1)
      print("\n")
      print("Shape:", array2.shape)
      print("Array 2:\n", array2)
```

Shape: (5,)
Array 1: [0 1 2 3 4]

Shape: (1,)
Array 2:
[3]

```
[44]: # Suma de ambos Arrays
      array1 + array2
```

```
[44]: array([3, 4, 5, 6, 7])
```

```
[45]: # Multiplicación
      array1 * array2
```

```
[45]: array([ 0,  3,  6,  9, 12])
```

15.- El código crea un array unidimensional con números impares del 1 al 19 (`np.arange(1, 20, 2)`) y luego imprime su contenido.

Funciones estadísticas sobre Arrays

```
[46]: #Creacion de un array multidimensional
      array1 = np.arange(1, 20, 2)
      print("Array1:\n", array1)
```

Array1:
[1 3 5 7 9 11 13 15 17 19]

```
[47]: #Media de los elementos del array
      array1.mean()
```

```
[47]: np.float64(10.0)
```

```
[48]: #Suma de los elementos del array
      array1.sum()
```

```
[48]: np.int64(100)
```


16.- El código calcula el cuadrado de cada elemento en array1 usando `np.square(array1)` y devuelve un nuevo array con los resultados.

Funciones universales proporcionadas por Numpy: ufunc

```
[49]: # Cuadrado de los elementos del array
      np.square(array1)

[49]: array([ 1,  9, 25, 49, 81, 121, 169, 225, 289, 361])

[50]: #Raiz cuadrada de los elementos del array.
      np.sqrt(array1)

[50]: array([1.         , 1.73205081, 2.23606798, 2.64575131, 3.         ,
        3.31662479, 3.60555128, 3.87298335, 4.12310563, 4.35889894])

[51]: # Exponencial de los elementos del array.
      np.exp(array1)

[51]: array([2.71828183e+00, 2.00855369e+01, 1.48413159e+02, 1.09663316e+03,
        8.10308393e+03, 5.98741417e+04, 4.42413392e+05, 3.26901737e+06,
        2.41549528e+07, 1.78482301e+08])

[53]: # log de los elementos del array
      np.log(array1)

[53]: array([0.         , 1.09861229, 1.60943791, 1.94591015, 2.19722458,
        2.39789527, 2.56494936, 2.7080502 , 2.83321334, 2.94443898])

[ ]:
```

Conclusión:

NumPy es una herramienta esencial para la computación científica y el análisis de datos en Python. Su capacidad para manejar arrays multidimensionales de manera eficiente, junto con una amplia gama de funciones matemáticas y herramientas avanzadas, lo convierte en una biblioteca fundamental para cualquier proyecto que requiera manipulación de datos y cálculos numéricos. La integración con lenguajes de bajo nivel y las capacidades para álgebra lineal y generación de números aleatorios amplían aún más su utilidad, haciendo de NumPy una pieza clave en el ecosistema de Python para la ciencia de datos y la ingeniería.