Nombre de la práctica	Numpy			No.	1
Asignatura:	Simulación	Carrera:	Ingeniería en Sistemas Computacionales	Duración de la práctica (Hrs)	

NOMBRE DEL ALUMNO: Vanesa Hernández Martínez
GRUPO: 3501

II. Lugar de realización de la práctica (laboratorio, taller, aula u otro):
Actividades en aula de clases y en equipo personal

III. Material empleado:
- Laptop
- Anaconda

Numpy

NumPy (Numerical Python) es una biblioteca fundamental para la computación científica en Python. Está diseñada para manejar grandes volúmenes de datos numéricos de manera eficiente. Entre sus valiosas funcionalidades, encontramos las siguientes:

Arreglos Multidimensionales (Arrays)

ndarray: Es el tipo principal de datos en NumPy, que permite crear arreglos (matrices) de N dimensiones. Estos son más eficientes que las listas de Python en términos de rendimiento y uso de memoria.Soporta operaciones matemáticas vectorizadas, lo que permite realizar cálculos sin la necesidad de bucles explícitos.

Operaciones Matemáticas

NumPy proporciona funciones rápidas para operaciones matemáticas como suma, resta, multiplicación, división, logaritmos, trigonometría, etc. Las operaciones en los arreglos de NumPy son más rápidas porque se realizan a nivel de bajo nivel (C/C++).

Funciones Estadísticas

NumPy tiene varias funciones para cálculos estadísticos como promedio (mean), desviación estándar (std), suma (sum), mínimo (min), máximo (max), etc.

Manipulación de Arreglos

NumPy permite fácilmente cambiar la forma de los arreglos (reshape), apilarlos, dividirlos, y transponer matrices.

Generación de Números Aleatorios

Proporciona herramientas para generar números aleatorios, lo cual es útil en simulaciones y pruebas estadísticas.

Introducción a Numpy

Numpy es una libreria 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 l arealizacion de las tareas relacionadas con álgebra lineal o números aleatorios.

Imports

```
import numpy as np
```

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 indice o clave. Un array se almacena de modo que la posición de cada elemento se pueda calcular apartir de su tupla de indice, mediante una fórmula matemática. El tipo mas simple de array es un array lineal tabien llamado array unidimensional.

En Numpy:

- Cada dimensión se denomina **axis**.
- El número de dimensiones se denomina **rank**.
- La lista de dimensiones con su correspondiente longuitud se denomina **shape**.
- El número total de elementos (multiplicación de la longuitud 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.],
```

a es un array:

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

```
a.shape
```

```
(2, 4)
```

```
a.ndim
```

```
2
```

```
a.size
```

```
8
```

Creación de arrays

```
# Array cuyos valores son todos 0.
```

```
np.zeros((2, 3, 4))
```

```
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.]]])
```

```
# Array cuyos valores son todos 1.
```

```
np.ones((2, 3, 4))
```

```
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.]]])
```

```
# Array cuyos valores son todos el valor indicado como segundo parámetro de la función.
```

```
np.full((2, 3, 4), 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]]])
```

```
# 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))
```

```
array([[[ 1.29131186e-316,  0.00000000e+000,  3.77172086e-317,
          7.23123644e-298,  6.80073566e-310,  3.77170505e-317,
```

```
        -4.28080863e-096,  6.80073700e-310,  6.80073566e-310], [
       6.38080118e-140,  6.80073566e-310,  6.80073565e-310,
         3.81849787e+301,  6.80073566e-310,  3.77170505e-317,
        -1.03628049e-287,  6.80073566e-310,  6.80073804e-310], [
       7.56244875e+056,  6.80073566e-310,  3.77172086e-317,
        -5.56741067e-194,  6.80073568e-310,  6.80073566e-310,
        -1.20231946e+047,  6.80073566e-310,  3.77170505e-317]],

      [[-1.05471104e-286,  6.80073566e-310,  6.80073565e-310, 6.08198946e-
         135,  6.80073566e-310,  1.50654528e-316,
        -4.99722324e+286,  6.80073566e-310,  6.80073568e-310], [
       7.48765886e-252,  6.80073566e-310,  6.80073566e-310,
        -2.09574284e+061,  6.80073566e-310,  3.77170505e-317,
        -1.45904605e-048,  6.80073566e-310,  3.77170505e-317], [-
       6.67149675e-082,  6.80073566e-310,  6.80073566e-310,
        -7.96297795e+078,  6.80073568e-310,  1.53190548e-316,
         1.97350947e+257,  6.80073566e-310,  1.54040736e-316]])
```

 # Inicializando el array utilizando un array de Python

```
b = np.array([[1, 2,3], [4, 5, 6]])
b
```

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

```
b.shape (2,
```

```
3)
```

 # Crear un array utilizando una funcion basada en rangos # (minimo, maximo, número de elemnts del array) print(np.linspace(0, 6, 10))

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

 # Inicializar el array con valores aleatorios.

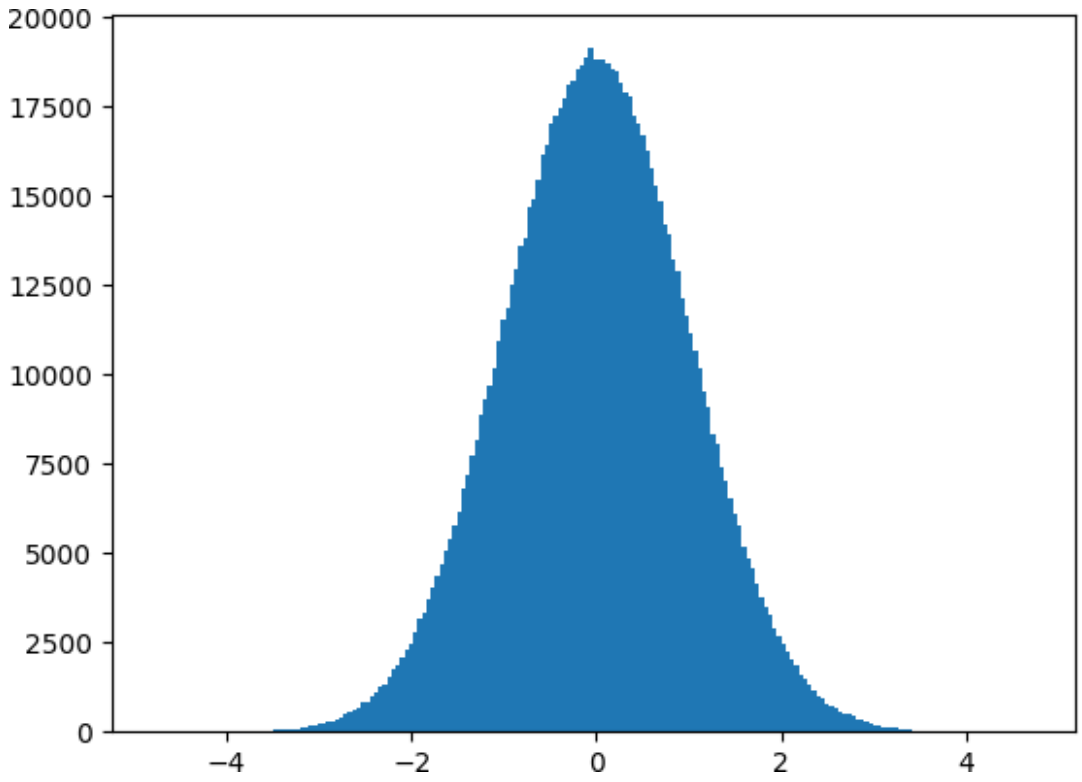
```
np.random.rand(3, 3, 4)
```

```
array([[[0.8426538 , 0.63755323, 0.51598983, 0.20923109],
        [0.89424774, 0.20489325, 0.90608485, 0.42176238],
        [0.56077529, 0.95576148, 0.78320915, 0.87110942]],

       [[0.3556085 , 0.68551554, 0.61837179, 0.54159098],
        [0.69324717, 0.12820094, 0.61481465, 0.79865897],
        [0.44368247, 0.14055149, 0.79498261, 0.90699857]],

       [[0.79403823, 0.81083356, 0.45631048, 0.98065189],
        [0.52556711, 0.002982  , 0.7156606 , 0.02572611],
        [0.40145605, 0.38100935, 0.49776159, 0.6625365 ]]])
```

```
# Iniciar array con valores aleatorios conforme a una distribucion normal .

np.random.randn(2,4)
```

```
array([[-2.60783208, -0.47897186,  1.54360315,  0.6337982 ],
       [-0.5421408 ,  0.99664992,  0.02803407, -0.1404264 ]])
```

```
%matplotlib inline

import matplotlib.pyplot as plt

c = np.random.randn(1000000) plt.hist(c,
bins=200) plt.show()
```




```
# Inicializar un array, utilizando una función personalizada.

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

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

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

Acceso a los elementos de un array

Array Unidimensional

```
# 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]
```

```
# Accediendo al quinto elemento del array.
```

```
array_uni[4] np.int64(9)
```

```
# Acceder al tercer y cuarto elemento del array
```

```
array_uni[2:4]
```

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

Array multidimensional

```
# Crear un array multidimensinal.
```

```
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]]
```

```
# Acceder al cuarto elemnto del array.
```

```
array_multi[0,3] np.int64(4)
```

```
# Acceder a la segunda fila del array
```

```
array_multi[1, :] array([5, 6,
```

```
7, 8])
```

```
# Accediendo al primer elemento de las dos primeras filas del array
```

```
array_multi[0:2, 2]
```

```
array([3, 7])
```

Modificacion de un array

```
# 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]
```

```
# 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]]
```

```
# El ejemplo anterior devuelve un array que apunta a los mismos datos. # 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]]
```

```
# Modificacion del nuevo array devuelto

array2[1, 3] = 30 print("Array2:\n", array2)
```

```
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]]
```

```
print("Array1:\n", array1)
```

```
Array1:
 [ 4  5  6  7]
 [ 8  9 30 11]
 [12 13 14 15]
 [24 25 26 27]]
```

Operaciones Aritméticas con Arrays

```
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]
```

```
# Suma
```

```
print(array1  +  array2) [ 2  5  8
```

```
11 14 17 20 23]
```

```
# Resta
```

```
print(array1 - array2) [2 3 4 5 6
```

```
7 8 9]
```

```
# Multiplicacion
```

```
# Nota: no es una multiplicacion de matrices
```

Broadcasting

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

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

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


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

```
# Suma de ambos arrays

array1 + array2 array([3, 4, 5, 6,

7])

# Multiplicación

array1 * array2

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

Funciones estadisticas sobre Arrays

```
# Creación de un array unidimensional array1 = np.arange(1,
20, 2) print("Array:\n", array1)
```

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

```
# Media de los elementos del array

array1.mean()

np.float64(10.0)

# Suma de los elementos del array

array1.sum()
```

Funciones universales proporcionadas por numpy: **unfunc**.

```
# Cuadrado de los elementos del array
```

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

# Raiz cuadrada de los elementos del array.

np.sqrt(array1)
array([1.        , 1.73205081, 2.23606798, 2.64575131, 3.        ,
       3.31662479, 3.60555128, 3.87298335, 4.12310563, 4.35889894])



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,
# Exponencial de los elementos del array.

np.exp(array1)

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

       2.41549528e+07, 1.78482301e+08])
```

Resumen de lo visto en la práctica:

Numpy

- Es una biblioteca para la computación en Python.
- Proporciona arrays N-dimensionales y funciones matemáticas avanzadas.
- Facilita tareas de álgebra lineal y generación de números aleatorios.
- Permite integrar código C/C++ y Fortran.

Arrays

Un array es una colección de elementos organizados por índices.

- **Dimensiones (axis)**: Cada dimensión es un "axis".
- **Rank**: Número de dimensiones del array.
- **Shape**: Lista de dimensiones con sus longitudes.
- **Size**: Número total de elementos (multiplicación de longitudes).

Creación de Arrays

- **np.zeros()**: Crea un array de ceros.
- **np.ones()**: Crea un array de unos.
- **np.full()**: Crea un array con un valor específico.
- **np.empty()**: Crea un array con valores aleatorios en memoria.
- **np.array()**: Inicializa el array a partir de una lista de Python.
- **np.linspace()**: Crea un array con valores distribuidos entre un rango.
- **np.random.rand()**: Genera arrays de valores aleatorios.

Acceso y Modificación de Arrays

- Los elementos del array se acceden por su índice.
- Se pueden cambiar las dimensiones y longitudes de un array con **reshape** o **shape**.
- Cambiar la forma no modifica los datos subyacentes.

Operaciones Aritméticas

- Las operaciones se aplican elemento por elemento.
- **Broadcasting**: Permite realizar operaciones entre arrays de diferentes formas ajustando sus dimensiones automáticamente.

Funciones Estadísticas

- NumPy proporciona funciones estadísticas como suma, promedio, y desviación estándar.
- Usa **ufuncs (universal functions)** para realizar operaciones matemáticas eficientes enarrays.

Conclusión

NumPy es una herramienta muy útil para trabajar con datos y hacer cálculos en Python. Facilita el manejo de grandes conjuntos de números a través de sus arrays, que son más rápidos y eficientes que las listas comunes. Con NumPy, es fácil hacer operaciones matemáticas, como sumar, multiplicar o encontrar promedios, todo de forma rápida y eficiente.

Además, permite crear arrays de muchas dimensiones, cambiarlos y realizar operaciones con arrays de diferentes tamaños gracias al **broadcasting**. En resumen, NumPy es una biblioteca clave para cualquiera que necesite hacer cálculos numéricos o trabajar con grandes cantidades de datos en Python, de manera sencilla y eficiente.