

Nombre de la práctica	Numpy			No.	1
Asignatura:	Simulación	Carrera:	Ingeniería en Sistemas Computacionales	Duración de la práctica (Hrs)	8

Nombre del alumno: **Sandra Alcántara Cruz**

Grupo: **3502**


**I. Competencia(s) específica(s):**

Comprende y analiza los conceptos de numpy

**II. Lugar de realización de la práctica (laboratorio, taller, aula u otro):**

AULA DE CLASES

**III. Material empleado:**

 Equipo de computo

 Python

**IV. Desarrollo de la práctica:**

**1. Introducción a Numpy**

[Numpy](#) es una biblioteca fundamental para la computación científica con python

- proporciona arrays N-dimensionales
- implementa funciones sofisticadas
- proporciona herramientas para integrar c/c++ y fortran
- proporciona mecanismos para facilitar la realización de tareas relacionadas con algebra lineal o números aleatorios

## Imports

```
[1]: 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 índice o clave. un array se almacena de

modo que la posición de cada elemento se pueda calcular a través de su tupla de índice mediante una fórmula matemática. el tipo más simple 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 que es la multiplicación de la longitud de las dimensiones se denomina **size**.

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

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

**a** es un 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 (2, 4) • con un **size** igual a 8

```
[3]: a.shape
```

```
[3]: (2, 4)
```

```
[4]: a.ndim
```

```
[4]: 2
```

```
[5]: a.size
```

```
[5]: 8
```

## Creacion de arrays

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

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

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

[7]: array([[8, 8, 8, 8],
          [8, 8, 8, 8],
          [8, 8, 8, 8]])

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

[8]: array([[1.52073070e-316, 0.00000000e+000, 2.74734810e-080,
          0.00000000e+000, 0.00000000e+000, 5.53682373e-317,
          1.56069909e-307, 4.09919231e-315, 6.72205493e-302],
          [2.10966031e-321, 5.31699581e-313, 2.52104764e-315,
          0.00000000e+000, 1.55755170e-307, 3.01275006e-282,
          3.12034120e-282, 3.09745422e-282, 4.56038381e-234],
          [3.61712994e-162, 0.00000000e+000, 0.00000000e+000,
          1.62446789e-214, 2.88531655e-128, 0.00000000e+000,
          1.38265635e-316, 3.71667844e-310, 4.18215543e-315]],
          [[3.01967325e-282, 4.56038407e-234, 4.42751685e-315,
          0.00000000e+000, 1.07034138e-296, 2.21168743e-316,
          2.88563622e-229, 6.89715642e-321, 8.25869242e-296],
          [4.94283807e-210, 8.90097491e-307, 2.21149653e-316,
          5.81554167e-309, 3.01238549e-310, 2.49208272e-306,
          2.21151036e-316, 1.00638549e-248, 6.89715642e-321],
          [8.25869242e-296, 4.22764033e-307, 7.01375591e-320,
          8.25869242e-296, 4.22764033e-307, 8.78580304e-268,
          1.10829416e-301, 1.39255974e-309, 5.56570092e-307]])

[9]: # Inicializacion del array utilizando un array de python
b = np.array ([[1, 2, 3], [4, 5, 6]])
b
```

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

[10]: b.shape

[10]: (2, 3)

[11]: # Creacion del array utilizando una funcion basada en rangos.
# minimo, maximo, número de elementos del array
# El print lo ve el usuario lo demas no
print(np.linspace(0, 6, 10))

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

```
[13]: # Inicializar un array con valores aleatorios.
      np.random.rand(2, 3, 4)

[13]: array([[0.71607228, 0.87425047, 0.95260894, 0.83226009],
            [0.68262354, 0.96772621, 0.84563394, 0.37991872],
            [0.26808655, 0.20612946, 0.80844394, 0.3830463 ]],

            [[0.6879514 , 0.58582906, 0.85252771, 0.31412342],
            [0.99941919, 0.04216291, 0.57899495, 0.22679014],
            [0.76348879, 0.44392534, 0.5232829 , 0.17930511]])

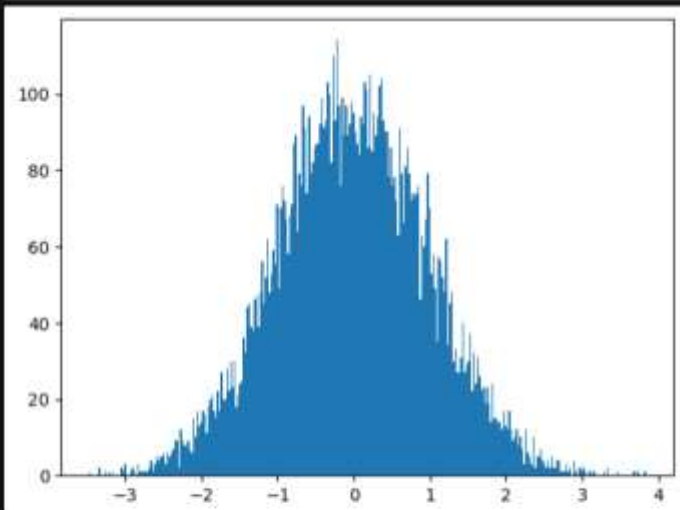
[14]: # Inicializar un array con valores aleatorios conforme a una distribución normal.
      np.random.randn(2, 4)

[14]: array([[ -0.25747888, -1.36257826,  2.5627755 , -0.39129991],
            [-0.65027261, -0.23263327,  1.13068336,  0.87213369]])

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

      c = np.random.randn(10000)

      plt.hist(c, bins=300)
      plt.show()
```



```
[20]: # Inicializar un array utilizando una función personalizada.
      def func(x, y):
          return x + 2 * y

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

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

## Acceso a los elemntos de un array

### Array unidimensional

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

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

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

[22]: np.int64(9)

[23]: # Accediendo al tercer y cuarto elemento del Array
# Los dos puntos dice que va a recorrer todo el arreglo
array_uni[2:4]

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

[29]: # Acceder a los elementos 0, 3, 5 del array.
array_uni[0::3]

[29]: array([1, 7])
```

### Array multidimensional

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

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

[30]: # Accediendo al cuarto elemento del Array
array_multi[0, 3]

[30]: np.int64(4)

[32]: # Accediendo a la segunda fila del array
array_multi[1, :]

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

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

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

## Modificación de un array

```
[34]: # Creación de un array unidimensional inicializado con el range de elementos 0-27
# range es un ciclo for
array1 = np.arange(28)
print("Shape: ", array1.shape)
print("Array_multi:", array1)

Shape: (28,)
Array_multi: [ 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]

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

Shape: (7, 4)
Array_multi:
[[ 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]]
```



```
[38]: # El ejemplo anterior devuelve un nuevo array que apunta a los mismos datos
# Nota: las modificaciones en un array, modifican al otro array
array2 = array1.reshape(4, 7)
print("Shape: ", array2.shape)
print("Array multi:\n", array2)
```

```
Shape: (4, 7)
Array multi:
[[ 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]]
```

```
[39]: # Modificación del Array Devuelto
array2[0, 3] = 20

print("Array 2:\n", array2)
```

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

```
[40]: print("Array 1:\n", array1)
```

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

```
[42]: # Devuelve el array a una sola dimensión
# Nota: el nombre array apunta a los mismos datos.
# Ravel regresa los datos como anteriormente estaba
print("Array 1:\n", array1.ravel())
```

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

## Operaciones aritméticas con arrays

```
[43]: # Creación de dos arrays unidimensionales.
array1 = np.arange(2, 10, 2)
array2 = np.arange(0)
print("Array 1:\n", array1)
print("Array 2:\n", array2)
```

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

```
[44]: # Suma
print(array1 + array2)

[ 2  5  8 11 14 17 20 23]
```

```
[45]: # Resta
print(array1 - array2)

[2 3 4 5 6 7 8 9]
```

```
[46]: # Multiplicación
# Nota: no es una multiplicación de matrices
print(array1 * array2)

[ 0  4 12 24 40 60 84 112]
```

## Broadcasting

Si se aplican operaciones aritméticas sobre arrays que no tiene la misma forma (shape) Numpy aplica una propiedad que se denomina Broadcasting.

```
[50]: # Creación de dos arrays unidimensionales
array1 = np.arange(5)
array2 = np.array([3])
print("Shape: ", array1.shape)
print("Array1:\n", array1)
print("Shape: ", array2.shape)
print("Array2:\n", array2)
```

```
Shape: (5,)
```





```

Shape: (5,)
Array1:
[0 1 2 3 4]
Shape: (1,)
Array2: [3]

[49]: # Suma de ambos arrays
array1 + array2

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

[51]: # Creación de dos arrays multidimensionales y unidimensional
array1 = np.arange(6)
array1.shape = (2, 3)
array2 = np.arange(6, 18, 4)
print("Shape: ", array1.shape)
print("Array1:\n", array1)
print("Shape: ", array2.shape)
print("Array2:", array2)

Shape: (2, 3)
Array1:
[[0 1 2]
 [3 4 5]]
Shape: (3,)
Array2: [ 6 18 14]

```

```

[52]: # Suma de ambos arrays
array1 + array2

[52]: array([[ 6, 11, 16],
 [ 9, 14, 19]])

```

## Funciones estadísticas sobre arrays

```

[53]: # Creación de un array unidimensional
array1 = np.arange(1, 20, 2)
print("Array1: ", array1)

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

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

[54]: np.float64(10.0)

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

[56]: np.int64(100)

```

## Funciones universales eficientes proporcionadas por Numpy: ufunc.

```

[57]: # Cuadrados de los elementos del array
np.square(array1)

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

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

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

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

[60]: array([2.71828183e+00, 2.00855369e+01, 1.48413159e+02, 1.09663316e+03,
 8.16388393e+03, 5.98741417e+04, 4.42413392e+05, 3.26981737e+06,
 2.41549528e+07, 1.70482301e+08])

[61]: # Logaritmo natural de los elementos del array
np.log(array1)

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

```

## Conclusión:

NumPy es una biblioteca clave en Python para trabajar con grandes volúmenes de datos numéricos de manera eficiente. Su principal fortaleza radica en los arreglos n-dimensionales, conocidos como arrays, que permiten realizar operaciones matemáticas complejas con una sintaxis simple y un rendimiento superior al de las listas nativas de Python. Además, optimiza el uso de la memoria y permite realizar operaciones vectorizadas, lo que mejora la velocidad de procesamiento.

En conclusión, NumPy es una herramienta poderosa que simplifica y acelera el cálculo numérico, siendo fundamental en proyectos de análisis de datos, aprendizaje automático y simulaciones científicas.