El futuro digital es de todos | MinTIC

UNIVERSIDAD EL BOSQUE | MisiónTIC 2022

Ciclo 1

# Semana 6

*Estructuras de Datos Lineales*

**Lectura 4** – *Implementaciones en Python*

*Convenio de asociación no. 777 de 2021 suscrito entre el Fondo Único de Tecnologías de la Información y las Comunicaciones y la Universidad el Bosque.*

# Implementaciones en Python

En algunas aplicaciones científicas o de ingeniería, es necesario hacer muchas operaciones sobre grandes conjuntos de datos de manera eficiente. La estructura de datos que sirve para este tipo de tareas, como hemos visto, son los arreglos. Sin embargo, en el lenguaje Python no tenemos ese manejo de forma directa. Afortunadamente, se tiene la librería NumPy, la cual nos da soporte para trabajar vectores y matrices, junto con una gran colección de funciones matemáticas de alto nivel para operarlos.

**Instalación:** desde la consola DOS se debe dar el comando: pip3 install numpy, y así de forma sencilla se instalará esta librería. Ver la siguiente figura:

```
C:\Users\herca>pip3 install numpy
Collecting numpy
  Downloading numpy-1.20.2-cp38-cp38-win_amd64.whl (13.7 MB)
     |████████████████████████████████| 13.7 MB 289 kB/s
Installing collected packages: numpy
Successfully installed numpy-1.20.2
```

Luego de la instalación ya se puede importar el módulo de la misma forma que se explicó cuando se estudió el tema de funciones.

**Creación de arreglos:** Se puede hacer directamente si conocemos los elementos del arreglo, por ejemplo, en el siguiente código se crean tres arreglos, uno entero, otro decimal y el último de cadena de caracteres.

```
import numpy as np
a = np.array([6,1,3,9,8])
b = np.array([3,4.5,8.3,5,2])
c = np.array(['Juan','Camilo','María','Verónica','Jesús'])
print(a)
print(b)
print(c)
```

```
[6 1 3 9 8]
[3.  4.5 8.3 5.  2. ]
['Juan' 'Camilo' 'María' 'Verónica' 'Jesús']

Process finished with exit code 0
```

Cuando los datos deben ser capturados por consola, se debe primero definir el tipo de dato del arreglo y luego establecer un ciclo para su captura. Por ejemplo, el siguiente código crea un arreglo de máximo 10 posiciones de cadenas de carácter, de máximo 50 caracteres cada una, que captura por consola y luego lo muestra uno por uno.

```
import numpy as np
ne = 10
a = np.zeros(ne,dtype='U50')
for i in range(ne):
    print('Digite el dato '+str(i)+': ',end='')
    a[i] = input()
for i in range(ne):
    print('Valor del dato '+str(i)+': '+a[i])
```

```
Digite el dato 0: Jesús
Digite el dato 1: María
Digite el dato 2: José
Digite el dato 3: Felipe
Digite el dato 4: Luis
Digite el dato 5: Eduardo
Digite el dato 6: Esperanza
Digite el dato 7: Daniela
Digite el dato 8: Lorena
Digite el dato 9: Julio
Valor del dato 0: Jesús
Valor del dato 1: María
Valor del dato 2: José
Valor del dato 3: Felipe
Valor del dato 4: Luis
Valor del dato 5: Eduardo
Valor del dato 6: Esperanza
Valor del dato 7: Daniela
Valor del dato 8: Lorena
Valor del dato 9: Julio

Process finished with exit code 0
```

**Operaciones sobre arreglos:** Las limitaciones que tienen los arreglos comparados con las listas, son ampliamente compensadas por la cantidad de operaciones convenientes que permiten realizar sobre ellos. Por ejemplo, las operaciones aritméticas entre arreglos se aplican elemento a elemento.

```
import numpy as np
a = np.array([55, 21, 19, 11,  9])
b = np.array([12, -9,  2, 22, -9])
c = a + b
d = a - b
e = a * b
f = a / b
print(c)
print(d)
print(e)
print(f)
```

```
[67 12 21 33  0]
[ 43  30  17 -11  18]
[ 660 -189   38  242  -81]
[ 4.58333333 -2.33333333  9.5        0.5       -1.        ]
```

Las operaciones entre un valor simple y un arreglo funcionan aplicando a todos los elementos del arreglo el valor simple como operando. También las operaciones relacionales se aplican elemento a elemento y retornan un arreglo booleano.

```
import numpy as np
a = np.array([55, 21, 2, 11,  9])
b = np.array([12, -9,  2, 22, -9])
c = a + 10
d = a < b
e = a > b
f = a == b
print(c)
print(d)
print(e)
print(f)
```

```
[65 31 12 21 19]
[False False False  True False]
[ True  True False False  True]
[False False  True False False]

Process finished with exit code 0
```

**Funciones sobre arreglos:** NumPy brinda muchas funciones matemáticas que también operan elemento a elemento, por ejemplo, si generamos 9 muestras espaciadas uniformemente entre 0 y pi/2, con una sola llamada a la función seno (sin) se obtiene.

```
import numpy as np
x = np.linspace(0,np.pi/2,9)
print(np.sin(x))
```

```
[0.         0.19509032 0.38268343 0.55557023 0.70710678 0.83146961
 0.92387953 0.98078528 1.        ]

Process finished with exit code 0
```

**Obtener elementos de un arreglo:** Cada elemento del arreglo tiene un índice, el primero elemento tiene índice cero (0) y también pueden numerarse desde el final al principio con índices negativos, -1 será el último elemento. Una sección de un arreglo se puede obtener usando el operador de rebanado a[ i : j ], los índices i y j indican el rango de valores que serán entregados. Si se omite el primer índice el rebanado se hace desde el comienzo y si el segundo índice es omitido el rebanado termina en el final del arreglo.

```
import numpy as np
a = np.array([55, 21, 2, 11,  9])
b = np.array([12, -9,  2, 22, -9])
c = a[1:4]
d = b[2:-2]
e = a[:3]
f = b[2:]
print(c)
print(d)
print(e)
print(f)
```

```
[21  2 11]
[2]
[55 21  2]
[ 2 22 -9]

Process finished with exit code 0
```

**Algunos métodos interesantes:** hay varios métodos interesantes que se deben conocer cuando se trabaja con arreglos. Los métodos min y max, que entregan el mínimo y máximo de los elementos, argmin y argmax, que entregan la posición del mínimo y del máximo y los métodos sum y prod, que retornan la suma y el producto de los elementos.

```
import numpy as np
a = np.array([55, 21, 2, 11,  9])
b = a.min()
c = a.max()
d = a.sum()
e = a.prod()
print(b)
print(c)
print(d)
print(e)
```

```
2
55
98
228690

Process finished with exit code 0
```