

Programación para Analítica de Datos MODULO EN PYTHON – Numpy

POR: Guillermo Andres De Mendoza Corrales



Listas – Matrices

Ejemplo matriz 3x3

```
matriz = [[1,2,3],[4,5,6],[7,8,9]]  
print(matriz)
```

```
[[1, 2, 3], [4, 5, 6], [7, 8, 9]]
```

	0	1	2
0	1	2	3
1	4	5	6
2	7	8	9



Listas – Matrices

Ejemplo matriz 3x3

```
matriz = [[1,2,3],[4,5,6],[7,8,9]]  
print(matriz)
```

```
[[1, 2, 3], [4, 5, 6], [7, 8, 9]]
```

	0	1	2
0	1	2	3
1	4	5	6
2	7	8	9



Listas – Matrices

```
matriz = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
```

Diagram illustrating the structure of the matrix `matriz`. The matrix is a list of three sub-lists. The first sub-list is `[1, 2, 3]`, the second is `[4, 5, 6]`, and the third is `[7, 8, 9]`. Red brackets above the sub-lists are labeled 0, 1, and 2 respectively. Green brackets below the elements of each sub-list are labeled 0, 1, and 2 respectively.

Obtener elemento individual

```
print(matriz[1][1])
```

5

	0	1	2
0	1	2	3
1	4	5	6
2	7	8	9



Listas – Matrices

Imprimir todos los elementos

```
for fila in range(0,len(matriz)):
    for columna in range(0,len(matriz[fila])):
        print("[%d][%d] = %d"%(fila,columna,matriz[fila][columna]))
```

```
[0][0] = 1
[0][1] = 2
[0][2] = 3
[1][0] = 4
[1][1] = 5
[1][2] = 6
[2][0] = 7
[2][1] = 8
[2][2] = 9
```

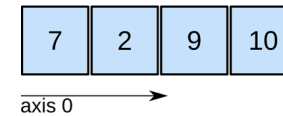


Libreria

Numpy: <https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#numpy.ndarray>

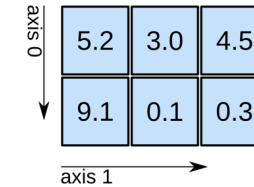
$$\begin{bmatrix} A & B \\ C & D \\ E & F \end{bmatrix} \times \begin{bmatrix} G & H \end{bmatrix} = \begin{bmatrix} A \times G + B \times H \\ C \times G + D \times H \\ E \times G + F \times H \end{bmatrix}$$

1D array



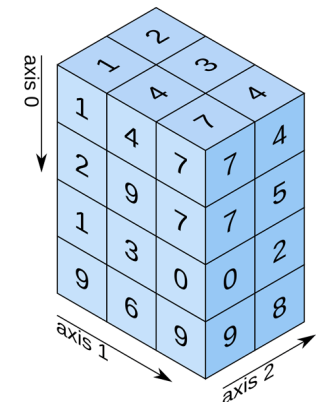
shape: (4,)

2D array



shape: (2, 3)

3D array



shape: (4, 3, 2)



Libreria

Numpy: <https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#numpy.ndarray>

```
import numpy as np
```



Numpy – crear matriz desde una lista

```
M1 = np.array( [[1,2,3],[4,5,6],[7,8,9]] )  
print(M1)
```

```
[[1 2 3]  
 [4 5 6]  
 [7 8 9]]
```

```
M2 = np.array( [[1,0,0],[0,1,0],[0,0,1]] )  
print(M2)
```

```
[[1 0 0]  
 [0 1 0]  
 [0 0 1]]
```


*las listas y los numpy se comportan igual pero son objetos diferentes



Numpy – crear matriz automatica

```
M3 = np.zeros( [3,4] )  
print(M3)
```

filas columnas



```
[[0.  0.  0.  0.]  
 [0.  0.  0.  0.]  
 [0.  0.  0.  0.]]
```



Numpy – crear matriz automatica

filas columnas

```
M3 = np.ones( [3,4] )  
print(M3)
```

```
[[1.  1.  1.  1.]  
 [1.  1.  1.  1.]  
 [1.  1.  1.  1.]]
```



Numpy – shape

```
M3 = np.zeros( [3,4] )  
print(M3)
```

```
[[0.  0.  0.  0.]  
 [0.  0.  0.  0.]  
 [0.  0.  0.  0.]
```

```
M3.shape
```

```
(3, 4)
```



Numpy – sumar

```
M1 = np.array( [[1,2,3],[4,5,6],[7,8,9]] )  
print(M1)
```

```
[[1 2 3]  
 [4 5 6]  
 [7 8 9]]
```

```
M2 = np.array( [[1,0,0],[0,1,0],[0,0,1]] )  
print(M2)
```

```
[[1 0 0]  
 [0 1 0]  
 [0 0 1]]
```

```
Msuma = M1 + M2  
print(Msuma)
```

```
[[ 2  2  3]  
 [ 4  6  6]  
 [ 7  8 10]]
```



Numpy – multiplicar

```
M1 = np.array( [[1,2,3],[4,5,6],[7,8,9]] )  
print(M1)
```

```
[[1 2 3]  
 [4 5 6]  
 [7 8 9]]
```

```
M2 = np.array( [[1,0,0],[0,1,0],[0,0,1]] )  
print(M2)
```

```
[[1 0 0]  
 [0 1 0]  
 [0 0 1]]
```

```
Mmultiplicacion = np.dot(M1, M2)  
print(Mmultiplicacion)
```

```
[[1 2 3]  
 [4 5 6]  
 [7 8 9]]
```



Numpy – multiplicar

```
M1 = np.array( [[1,2,3],[4,5,6],[7,8,9]] )  
print(M1)
```

```
[[1 2 3]  
 [4 5 6]  
 [7 8 9]]
```

```
Mmultiplicacion = 3 * M1  
print(Mmultiplicacion)
```

```
[[ 3  6  9]  
 [12 15 18]  
 [21 24 27]]
```



Numpy – determinante

```
M1 = np.array( [[1,2,3],[4,5,6],[7,8,9]] )  
print(M1)
```

```
[[1 2 3]  
 [4 5 6]  
 [7 8 9]]
```

```
M1determinante = np.linalg.det(M1)  
print(M1determinante)
```

```
-9.51619735392994e-16
```



Numpy – invertir

```
M1 = np.array( [[1,2,3],[4,5,6],[7,8,9]] )  
print(M1)
```

```
[[1 2 3]  
 [4 5 6]  
 [7 8 9]]
```

```
M1invertida = np.linalg.inv(M1)  
print(M1invertida)
```

```
[[ 3.15251974e+15 -6.30503948e+15  3.15251974e+15]  
 [-6.30503948e+15  1.26100790e+16 -6.30503948e+15]  
 [ 3.15251974e+15 -6.30503948e+15  3.15251974e+15]]
```



Numpy – transpuesta

```
M1 = np.array( [[1,2,3],[4,5,6],[7,8,9]] )  
print(M1)
```

```
[[1 2 3]  
 [4 5 6]  
 [7 8 9]]
```

```
M1transpuesta = M1.T  
print(M1transpuesta)
```

```
[[1 4 7]  
 [2 5 8]  
 [3 6 9]]
```



Numpy – modify value

```
M1 = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])  
M1[1][1] = 0  
print(M1)
```

```
[[1 2 3]  
 [4 0 6]  
 [7 8 9]]
```



Numpy – generar samples X

Init End Samples

```
X = np.linspace(-1,1,50)  
print(X)
```

```
[-1.          -0.95918367 -0.91836735 -0.87755102 -0.83673469 -0.79591837  
 -0.75510204 -0.71428571 -0.67346939 -0.63265306 -0.59183673 -0.55102041  
 -0.51020408 -0.46938776 -0.42857143 -0.3877551  -0.34693878 -0.30612245  
 -0.26530612 -0.2244898  -0.18367347 -0.14285714 -0.10204082 -0.06122449  
 -0.02040816  0.02040816  0.06122449  0.10204082  0.14285714  0.18367347  
  0.2244898   0.26530612  0.30612245  0.34693878  0.3877551   0.42857143  
  0.46938776  0.51020408  0.55102041  0.59183673  0.63265306  0.67346939  
  0.71428571  0.75510204  0.79591837  0.83673469  0.87755102  0.91836735  
  0.95918367  1.          ]
```



Numpy – generar samples FNormal

Distribución normal: of mean 0 and variance 1

```
M = np.random.randn(10)  
print(M)
```

```
[ 1.17226855 -0.15268709  0.98457968  0.17050347 -0.86151484  0.2404382  
 0.87838827  0.07127004  0.43030506  1.0120156 ]
```



Numpy – contiene multiples distribuciones

np.random.

f	beta	function
f	binomial	function
m	bit_generator	module
c	BitGenerator	class
f	bytes	function
f	chisquare	function
f	choice	function
f	default_rng	function
f	dirichlet	function
f	exponential	function

np.random.|

f	hypergeometric	function
f	laplace	function
f	logistic	function
f	lognormal	function
f	logseries	function
c	MT19937	class
m	mtrand	module
f	multinomial	function
f	multivariate_normal	function
f	negative_binomial	function



Numpy – generar samples FNormal

Distribución normal: of mean 0 and variance 1

```
M = np.random.randn(3,3)
print(M)
```

```
[[ 1.61582641  1.10031642 -0.4840879 ]
 [ 0.73353939 -1.76448134  2.08425915]
 [-0.90475599 -1.12374196 -0.20284926]]
```



Numpy – append

vector

```
arr = np.array([1, 2, 3])  
arr_nuevo = np.append(arr, 4)  
print(arr_nuevo)
```

```
[1 2 3 4]
```

matrix

```
M = np.array([[1, 2, 3], [4, 5, 6]])  
M_nueva = np.append(M, [[7, 8, 9]], axis=0)  
print(M_nueva)
```

```
[[1 2 3]  
 [4 5 6]  
 [7 8 9]]
```



Numpy – insert

```
M = np.array([[1, 2, 3], [4, 5, 6]])  
M_nueva = np.insert(M, 1, [[7, 8, 9]], axis=0)  
print(M_nueva)
```

```
[[1 2 3]  
 [7 8 9]  
 [4 5 6]]
```



Numpy – eliminar

```
M = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])  
M_new = np.delete(M, 1, axis=0)  
print(M_new)
```

```
[[1 2 3]  
 [7 8 9]]
```

