

Arreglos de Numpy dos dimensiones

Fundamentos de Programación
FIEC04341

Terminología

¿Qué son los arreglos de dos dimensiones?

En matemáticas es similar a una matriz, que es un arreglo rectangular de elementos de n filas y m columnas.

- En Python es una lista cuyos elementos son listas que tienen la misma longitud y contienen elementos del mismo tipo.
- En Python existen módulos especiales para extender el uso de vectores y matrices. Ej: NumPy

¿Para qué sirven?

- Se puede utilizar para representar cuadros de datos organizados por filas y columnas. Ejemplo:
- La siguiente tabla contiene el número de productos sobre las ventas de una compañía de 3 tipos de productos distintos, cada fila representa, los meses de ene, feb, mar, abril

	1	2	3
ene	23	45	63
feb	72	81	91
mar	56	64	37
abril	34	75	26

Representación Matricial

$$A = \begin{bmatrix} 23 & 45 & 63 \\ 72 & 81 & 91 \\ 56 & 64 & 37 \\ 34 & 75 & 26 \end{bmatrix}_{4 \times 3}$$

##Python

```
A= [ [23, 45, 63] , [72, 81, 91] , [56, 64, 37] , [34, 75, 26] ]
```

- **A** es una lista que contiene 4 lista de enteros que representan a las 4 filas
- Estas listas, tienen todas 3 elementos del mismo tipo (enteros)

Representación de listas

0	23	45	63
1	72	81	91
2	56	64	37
3	34	75	26
	0	1	2

##Consola Python

```
>> A=[[23,45,63],[72,81,91],[56,64,37],[34,75,26]]
>> A[0][1] #elemento de la fila 1 columna 2 A(0,1)
45
>> A[2][0] #elemento de la fila 3 columna 1 A(2,0)
56
>> A[1] #imprimir la fila 2
[ 72, 81, 91 ]
>>
```

Representación de listas

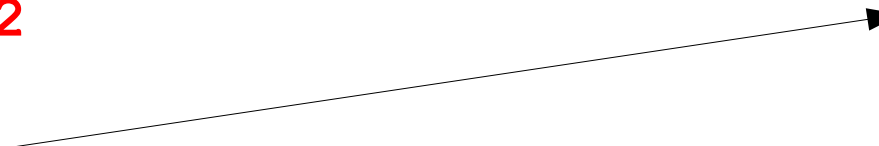
23	45	63
72	81	91
56	64	37
34	75	26

##Script en Python

```
A= [[23,45,63],[72,81,91],[56,64,37],[34,75,26]]
```

##imprimir la columna 2

```
for i in range(4):  
    print(A[i][1])  
print('Done')
```



45
81
64
75
Done

Crear Matrices

Lista de listas

##Crea una lista de 5 listas vacías

```
a=[]  
for i in range(5):  
    a=a+[[[]]]  
print(a)
```

```
>>>[[[]], [], [], [], []]
```

##Crea una matriz de 4x3

```
a=[]  
for i in range(4):  
    a.append([])  
    for j in range(3):  
        a[i].append(j)  
print(a)
```

```
>>>[[0, 1, 2], [0, 1, 2], [0, 1, 2], [0, 1, 2]]
```

Lista de listas

##Crea una matriz de 4x3 con números aleatorios

```
from random import*
```

```
a=[]
```

```
for i in range(4):
```

```
    a.append([])
```

```
    for j in range(3):
```

```
        ale = randint(0,9)
```

```
        a[i].append(ale)
```

```
print(a)
```

```
>>>[[7, 2, 8], [5, 1, 0], [5, 8, 9], [0, 8, 3]]
```

NumPy

NumPy

- Es una librería de soporte para aplicaciones matemáticas, científicas y de ingeniería.
- Provee una estructura de datos para manejo de vectores y matrices denominada **array** o arreglo, similares a las listas con la diferencia de que sus elementos deben ser del mismo tipo
- Para utilizar esta librería es necesario instalar los binarios e importar la librería:

```
(1) from numpy import*
```

```
(2) import numpy as np
```

Crear vectores y matrices

##Crea un vector fila

```
from numpy import*  
v=array([3,6,7,2,8])  
print(v)
```

```
>>>  
[3, 6, 7, 2, 8]
```

##Crea un vector columna

```
v=array([[3],[6],[7],[2],[8]])  
print(v)
```

```
>>>  
[[3]  
 [6]  
 [7]  
 [8]]
```

##Crea una matriz de 3x3

```
a=array([[4,20,5],[2,8,4],[6,9,5]])  
print(a)
```

```
>>>  
[[4 20 5]  
 [2  8 4]  
 [6  9 5]]
```

Manipulación de vectores y matrices

##Se puede especificar el tipo de dato

```
a=array([[4,2,5],[2,8,4],[6,9,5]], float)
```

```
print(a)
```

```
>>>
```

```
[[4. 2. 5.]
```

```
 [2. 8. 4.]
```

```
 [6. 9. 5.]]
```

##Los elementos se acceden

```
print(a[1][2])
```

```
>>>
```

```
4.0
```

```
print(a[1,2])
```

##Se pueden manipular filas o columnas completas

```
print(a[1,:])
```

```
>>>
```

```
[2. 8. 4.]
```

```
print(a[:,1])
```

```
>>>
```

```
[2. 8. 9.]
```

Manipulación de vectores y matrices (2)

##Dimensiones de una matriz

```
n, m = a.shape
```

```
print(n, m)
```

```
>>>
```

```
3 3
```

##El operador in

```
>>>8 in a
```

```
True
```

##Un arreglo se puede generar

```
b = array(range(9), int)
```

```
print(b)
```

```
>>>
```

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

##Un arreglo se puede redimensionar a matriz

```
b = b.reshape((3,3))
```

```
print(b)
```

```
>>>
```

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

Manipulación de vectores y matrices (3)

##Convertir un array a lista

```
c = b.tolist()
```

```
>>>[[0,1,2],[3,4,5],[6,7,8]]
```

##Convertir una lista a un arreglo

```
v=[[5,4],[7,3]]
```

```
u=array(v)
```

```
print(u)
```

```
>>>[[5 4]
      7 3]]
```

##Rellenar con un valor

```
b = b.fill(1)
```

```
print(b)
```

```
>>>
```

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


Operaciones matemáticas con matrices

```
a=array([[2,3],[4,5]])
```

```
b=array([[5,2],[1,4]])
```

##Operadores aritméticos entre matrices

```
>>>a + b
```

```
>>>a - b
```

```
>>>a * b #multiplicación elemento a elemento
```

```
>>>dot(a, b) #multiplicación de matrices
```

##Funciones

```
>>>sum(a) #suma todos los elementos
```

```
>>>prod(a) #producto de todos los elementos
```

```
>>>mean(a) #media aritmética
```

```
>>>std(a) #desviación estándar
```

```
>>>sort(a) #ordenamiento por filas
```

Operaciones matemáticas con matrices

##Mas Funciones

```
>>>linalg.det(a)      #determinante
>>>linalg.inv(a)      #inversa
>>>diagonal(a)        #elementos de la diagonal
>>>transpose(a)       #media aritmética
>>>tril(a)            #triangular inferior
>>>triu(a)            #triangular superior

a=zeros([3,4], int)    #matriz cero 3x4 tipo entero
a=ones([3,4], float)   #matriz de unos 3x4 floats
a=identity(5)          #matriz identidad 5x5

>>>np.random.randint(1,6,4)
array([4, 3, 2, 2])    #vector de 4 nums aleatorios
>>>np.random.randint(1,6,[2, 2])
array([4, 3],[2, 2])  #matriz de 2x2 de aleatorios
```

Funciones con el mismo nombre en Python

Para llamar a estas funciones se debe importar numpy de la segunda forma sino el interprete utilizará la funciones dirigidas a listas o colecciones

```
import numpy as np
a=np.array([[4,2,5],[2,8,4],[6,9,5]])
np.sum(a)
np.min(a)
np.max(a)
np.argmin(a) #devuelve indice contado por fila
np.argmax(a) #igual que anterior pero el max
```