Arreglos de Numpy
Vectores (1D)

Fundamentos de Programación
FIEC04341

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

```
##Crea un vector fila
import numpy as np
v = np.array([3,6,7,2,8])
```

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

```
##A partir de una cadena
v = np.array( list('casa') )
```

```
>>> v
['c','a','s','a']
```

```
##La función array permite indicar el tipo de dato
a = np.array([4,2,5], float)
```

```
>>> a
[4.,2.,5.]
```

- Para crear un vector de numpy se necesita una lista o colección que contenga los mismos tipos de datos.

Manipulación de vectores

```
##Para acceder a los elementos se indexa
a = np.array([4,2,5], float)
print(a[1])
print(a[2])
```

```
>>>
4.0
>>>
2.0
```

```
##Se puede recortar un vector
print(a[1:])
```

```
>>>
[ 2. 5. ]
```

```
##Para obtener el tamaño del vector
print(a.size)
```

```
>>>
3
```

```
##Todas las operaciones de slicing son permitidas
c = a[::-1]
```

- El slicing retorna un nuevo vector que debe ser asignado, similar que en las listas

Manipulación de vectores (2)

```
## El uso de la función len()
>>>len(a)
3

## Generar una secuencia a partir de range()
b = np.array(range(9), int)
```

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

```
## Numpy ofrece la función arange()
b = np.arange(2, 9, dtype=float)
```

```
>>>
[ 2. 3. 4. 5. 6. 7. 8. ]
```

```
## Para generar números aleatorios
c = np.random.randint(2, 9, dtype=float)
```

Operaciones matemáticas con vectores

```
##Operadores aritméticos con escalares
>>> a + 3
>>> a // 2
>>> 3 - a

##Operadores aritméticos entre vectores
>>>a + b #operaciones son elemento a elemento
>>>a - b #los vectores deben tener igual longitud
>>>a * b
>>>np.dot(a, b)#producto punto de vectores

##Funciones
>>>a.sum()    #suma todos los elementos
>>>a.min()    #obtiene el valor mínimo
>>>a.max()    #obtiene el valor máximo
>>>a.mean()   #obtiene el promedio
>>>a.std()    #obtiene la desviación estándar
```

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
```

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]]
```

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
```