Numpy

October 10, 2017

1 Numpy

Numpy se trata de un paquete fundamental para computación científica en Python, ya que contiene: objetos de tipo array N-dimensionales muy potentes, sofisticadas funciones. El principal problema que tenemos con las lista es que no podemos realizar operaciones matemáticas entre ellas por ejemplo si tenemos una lista con el precio por unidad de diferentes productos y otra que contiene la cantidad de ventas de cada producto no podemos hacer una operación del tipo lista_precios * lista_ventas, para poder hacer eso debemos de hacer uso de Numpy. Para ello antes debemos de importar numpy a nuestro espacio de trabajo y tras esto hacer uso del método array(), que lo que hace es transformar un objeto de tipo lista a un objeto de tipo array que es con lo que trabaja Numpy.

```
In [6]: #Importamos Numpy
        import numpy as np
        #Nos creamos dos listas: una que contiene el precio de un producto y otra que contiene e
        #de los productos
        lista_productos = ['Boligrafo', 'Libreta', 'Agenda', 'Grapadora']
        lista_precios = [1.90, 2.45, 3.89, 10.90]
        lista_ventas = [10, 15, 20, 5]
        #Convertimos las listas a objetos de tipo array
        np_lista_precios = np.array(lista_precios)
        np_lista_ventas = np.array(lista_ventas)
        np_lista_productos = np.array(lista_productos)
```

Ahora podemos operar con los objetos de tipo array, por ejemplo nos podemos crear un array denominado ganancias que nos indique la cantidad que hemos ganados sin más que multiplicar las dos arrays.

```
In [2]: #Nos creamos la array np_ganancias
        np_ganancias = np_lista_precios * np_lista_ventas
        print(np_ganancias)
```

```
[ 19.    36.75  77.8   54.5 ]
```

Al igual que en las listas podemos hacer uso de expresiones regulares para filtrar y quedarnos con lo que nos interesa. Por ejemplo supongamos que estamos interesados en quedarnos con los productos cuyas ganancias fueron superiores a 40.

```
In [8]: productos  = np_lista_productos[np_ganancias > 40]
        print(productos)
```

```
['Agenda' 'Grapadora']
```

En el caso de arrays el operador suma, resta y división actuan como operadores matemáticos, de forma que realizan la operación componente a componente. Para operar entre dos arrays estas deben tener la misma dimensión.

```
In [13]: #Nos creamos dos arrays
         array1 = np.array([1,2,3])
         array2 = np.array([3,4,5])
         #Sumamos las dos arrays
         array_sum = array1 + array2
         print(array_sum)
```

```
[4 6 8]
```

```
In [14]: #Restamos ambas arrays
         array_dif = 2*array1 - array2
         print(array_dif)
```

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

```
In [15]: #Dividimos ambas arrays
         array_div = array2/array1
         print(array_div)
```

```
[ 3.          2.          1.66666667]
```

A la hora de seleccionar elementos de un array actuamos igual que en las listas. Para seleccionar un elemento concreto de un array lo hacemos mediante el índice, teniendo en cuenta que le primer índice es el cero. También podemos acceder a un array en el orden inverso haciendo uso de índices negativos empezando en -1. Finalmente podemos seleccionar múltiples elementos de un array haciendo uso del operador :

```
In [17]: #Nos creamos un array
         paises = np.array(['USA', 'España', 'Croacia', 'Tailandia'])
         #Accedemos al primer elemento
         print(paises[0])
```

```
USA
```

```
In [18]: #Accedemos al último elemento
         print(paises[-1])
```

```
Tailandia
```

```
In [19]: #Accedemos al segundo y el tercer elemento
         print(paises[1:3])
```

```
['España' 'Croacia']
```

Nos podemos crear arrays bidimensionales, podemos ver que en este caso los valores numéricos son tomados por string.

```
In [22]: #Nos creamos nuestra primer array bidimensional
         array_champions = np.array([['Manchester United', 3], ['Milan', 7], ['Barcelona', 5], [
         print(array_champions)
```

```
[['Manchester United' '3']
 ['Milan' '7']
 ['Barcelona' '5']
 ['Real Madrid' '12']]
```

El atributo .shape nos permite conocer la dimensión de una array. Este resultado retorna una tupla donde el primer valor es el número de filas (rows) y el segundo valor es el número de columnas (cols).

```
In [23]: print(array_champions.shape)
```

```
(4, 2)
```

A la hora de seleccionar elementos de una array bidimensional debemos indicar mediante índices la fila y la columna donde se encuentra el elemento al que deseamos acceder.

```
In [25]: #Accedemos al número de champions league ganadas por el milan
         print(array_champions[1,1])
```

```
7
```

También podemos seleccionar filas y columnas completas de una array bidimensional.

```
In [26]: #Seleccionamos la segunda columna completa
         print(array_champions[:,1])
```

```
['3' '7' '5' '12']
```

```
In [27]: #Seleccionamos la segunda fila completa
         print(array_champions[1,:])
```

```
['Milan' '7']
```

```
In [28]: #Seleccinamos la segunda y la tercera columna
         print(array_champions[1:3,:])
```

```
[['Milan' '7']
 ['Barcelona' '5']]
```

También podemos sumar, restar y multiplicar elementos de un array, teniendo en cuenta que la operación aplicada actua sobre todos los elementos.

```
In [33]: array_numerica = np.array([[1,2], [3,4]])
         #Sumamos uno a cada uno de los elementos
         print(array_numerica + 1)
```

```
[[2 3]
 [4 5]]
```

```
In [34]: #Multiplicamos por dos todos los elementos
         print(array_numerica * 2)
```

```
[[2 4]
 [6 8]]
```