ANyP, Práctica 1c.

Elementos de programación Python: Arreglos NumPy.

Datos compuestos indexados: arreglos NumPy.

Los objetos matemáticos como *vectores*, *matrices* (y con más generalidad, *tensores*) pueden ser almacenados en Python en *arreglos* unidimensionales, bidimensionales (y multidimensionales), respectivamente, utilizando el módulo `numpy` que deberemos importar haciendo,

```
import numpy as np
```

Un arreglo (*array*, en inglés) *NumPy* es una estructura de datos *compuesta* formada por una colección ordenada de elementos del mismo tipo de datos (por lo que se dice que la estructura de datos es *homogénea*) y tal que los elementos se pueden acceder en cualquier orden simplemente indicando la posición que ocupa dentro de la estructura (por lo que se dice que es una estructura *indexada*).
Un arreglo, como un todo, se identifica en el programa con un *nombre*, y un elemento particular del mismo es accedido a partir de cierto conjunto de *índices*, los cuales son, en Python, de tipo entero. La *dimensión* del arreglo es el número de índices necesarios para especificar un elemento, y su *tamaño* es el número total de elementos que contiene.
Consideremos la matriz

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$$

Para definir esta matriz usando el módulo NumPy hacemos,

```
import numpy as np
A = np.array( [[1,2,3], [4,5,6]] )
```

Utilizando los elementos del lenguaje dados en la teoría, procure resolver los siguientes ejercicios.

Ejercicio 1. Usando NumPy, escriba un programa Python para:

a) Crear un arreglo unidimensional con los números del 1 al 10 utilizando la función `arange`.
b) Crear un arreglo bidimensional (3x3) con los números del 1 al 9 utilizando las funciones `arange` y `reshape`.
c) obtener las propiedades de los arreglos de los ítems anteriores: forma (`shape`), tamaño (`size`), y tipo de datos (`dtype`).
d) Crear el vector nulo de tamaño 5 y luego modificar el tercer elemento por 1 (puede serle útil el método `zeros`).
e) Crear un arreglo de 7x7 con unos en la primera y última filas y columnas, y cero en los demás elementos,
f) Crear la matriz identidad de 3x3 utilizando el método `eye`.
g) Crear una matriz de 5x5 con filas tomando valores del 0 al 4,

Ejercicio 2. Considerar un vector definido como $x = (0.1, 1.3, -1.5, 0, 12.3)$. Escriba sus propias implementaciones usando bucles en Python para los siguientes problemas:

a) Calcule la suma $s_1 = \sum_{i=1}^{5} x_i$.
b) Calcule la suma $s_2 = \sum_{i=1}^{5} x_i^2$.
c) Encuentre el elemento de mayor valor absoluto $M = \text{máx}\{|x_i|, i = 1, 2, ..., 5\}$.
d) Encuentre el elemento de menor valor absoluto $M = \min\{|x_i|, i = 1, 2, ..., 5\}$.

Ejercicio 3. Compare los resultados del ejercicio anterior con los obtenidos a partir de utilizar las funciones `sum()`, `max()` y `min()` de NumPy.

Ejercicio 4. Escriba los comandos para cada una de las siguientes operaciones:

a) Cree el vector fila x de 5 elementos equiespaciados cuyo valor se encuentre entre 2 y 3.
b) Sume 1 al segundo elemento.
c) Cree el vector fila y, de igual dimensión, con elementos pares, comenzando por el 4.
d) Cree una matriz A cuya primera fila sea igual a x, su segunda fila contenga unos, y cuya tercera fila sea igual a y.
e) Defina el vector fila z cuyos elementos sean iguales al valor medio de cada columna de A.

Ejercicio 5. Dada una matriz $A^{m\times n} \in \mathbb{R}$ de su elección, analice el resultado de:

a) Sumarle un escalar.

b) Sumarle un vector fila $b^{1\times n}$.

c) Sumarle un vector columna $c^{m\times 1}$.

¿Se da un comportamiento análogo con las operaciones resta (`-`), multiplicación (`*`) y división (`/`)?

Ejercicio 6. Cree las matrices

$$A = \begin{pmatrix} 1 & 2 \\ 4 & -1 \end{pmatrix} \quad \text{y} \quad B = \begin{pmatrix} 4 & -2 \\ -6 & 3 \end{pmatrix}$$

a) Calcule $C_1 = A + B$ y $C_2 = A - B$.

b) Calcule las matrices producto $D_1 = A \cdot B$ y $D_2 = B \cdot A$ utilizando lazos `for`.

c) Calcule los productos utilizando el operador @. ¿Coinciden los resultados?

Ejercicio 7. Usando la función `random.randint` cree un vector con 10 elementos enteros generados al azar entre 1 y 100. Ordene los elementos de mayor a menor mediante la función `sort` de `numpy` y muestre el vector resultante en pantalla.

Ejercicio 8. Defina una matriz cuadrada de 4x4 y cree un programa que verifique si es simétrica o no. *Ayuda*: puede ser de utilidad la función `transpose` de NumPy.

Ejercicio 9. Sean los vectores $x = (5, -3, 2)$ e $y = (2, 3, 4)$. Cree un programa que, usando bucles `for`, calcule el producto escalar de ambos vectores. Realice otro programa que utilice la función `.dot` de NumPy para realizar el mismo cálculo.

Ejercicio 10. Utilizando la función `random.rand` genere una matriz aleatoria $A^{50\times 50}$ con elementos entre $-\pi$ y π.

a) Utilizando bucles `for` y un bloque `if`, genere una nueva matriz, reemplazando por -1 los números negativos, y por 1 los positivos.

b) Hacer lo mismo, usando la función `where`.

c) Compare los tiempos de cómputo de los incisos a) y b).

Nota: En el último inciso podemos usar la función `perf_counter()` del módulo *Time*. Invocándola antes y después del bloque de código a evaluar, asignando el resultado a sendas variables y restándolas, se obtiene el tiempo empleado.

Ejercicio 11. Mediante comprensión de listas, genere un arreglo numpy `A` de reales que comience en 10. y finalice en 30.

a) Considere las asignaciones `B = A[0::2]` y `C = np.array(A[0::2])`.

b) Cambie el primer elemento de `B` por 200. ¿Qué sucede con `A`?

c) Cambie el primer elemento de `C` por 300. ¿Qué sucede con `A`?

d) Cambie ahora el último elemento de `A` por 500. ¿Qué sucede con `B` y `C`?

e) Explique estas resultados. ¿Qué método propio de arreglos podría haber aplicado al definir a `B` para obtener el mismo comportamiento que el de `C`?