

1. En esta práctica usted ejercitará y explorará algunas características del poderoso módulo Numpy. Para ello, cargue el módulo usando:

```
from numpy import *
```

2. En Numpy existe una función llamada `arange` que es muy similar a la ya conocida función `range`. La diferencia es que `range` genera una **lista** mientras que `arange` genera un **arreglo** de Numpy. Para comprobar esto, ejecute:

```
x=range(10)
y=arange(10)
print(x,type(x))
print(y,type(y))
```

En otras palabras, `arange(10)` es equivalente a `array(range(10))`.

3. Usando arreglos de Numpy es posible realizar muchos cálculos en forma rápida y eficiente, sin necesidad de recurrir a ciclos (`for` o `while`). Por ejemplo, puede calcular la misma suma calculada en el punto 2 de la guía 03 usando

```
x=arange(101)
suma=sum(1./2.**x)
print(suma)
```

Verifique lo anterior y asegúrese de entender qué se está calculando.

4. Adapte la idea del cálculo en el punto anterior para implementar un cálculo alternativo para el factorial de un número  $n$  (entero positivo), pero esta vez usando un arreglo de Numpy y la función `prod()` que calcula el producto de cada componente de un arreglo de Numpy (similarmente a como `sum()` calcula la suma).
5. Verifique que, a diferencia de su pariente `range()`, la función `arange()` también funciona con pasos decimales, por ejemplo

```
print(arange(1,10,0.3))
```

6. Otra función muy útil para crear arreglos de valores en un intervalo es `linspace()`, que tiene el formato `linspace(desde,hasta,numerodeelementos)`. Por ejemplo, ejecute los siguientes comandos:

```
x=linspace(1,10,20)
y=linspace(-pi,pi,100)
print(x,size(x))
print(y,size(y))
```

7. Otra propiedad importante de los arreglos es que sus elementos pueden usarse para iterar en un ciclo `for`. Para ver esto, ejecute:

```
x=arange(11)
y=x**2
for i in x:
    print ("la componente "+str(i)+" de y es igual a "+str(y[i]))
```

8. Considere el archivo de datos `datos.txt`. Numpy contiene la función `genfromtxt`, que lee datos desde un archivo y los asigna a un arreglo, de la dimensión apropiada. Ejecute (en la misma carpeta donde está el archivo `datos.txt`) los siguientes comandos:

```
d=genfromtxt("datos.txt")
x=d[:,0]
y=d[:,1]
```

La primera línea carga los datos al arreglo `d`. Las últimas dos líneas asignan la primera columna de datos al arreglo `x` y la segunda columna a `y`. Usando las funciones `shape` y `size` de Numpy, verifique la forma y tamaño de los arreglos `d`, `x` e `y`. Asegúrese de entender que es lo que realiza exactamente cada comando anterior.

9. Usando lo anterior, calcule e imprima:
- (a) El promedio de los valores de la primera columna. (puede usar la función `sum` y `len` para calcular el promedio, o bien la función `mean` de Numpy).
  - (b) El promedio de los cuadrados de los valores de la segunda columna.
  - (c) La suma de los productos de cada elemento de la primera con la segunda columna (es decir,  $0.1 * 0.738 + 0.25 * 0.826 + 0.41 * 0.981 + \dots$ ).
10. **Bonus Track (opcional):** Los arreglos de Numpy también pueden contener componentes con valores booleanos (siempre que todos los componentes lo sean). Por ejemplo,

```
x=arange(11)
y=(x>5)
print(x)
print(y)
```

muestra que `y` es un arreglo cuyas componentes son verdaderas o falsas dependiendo si la condición impuesta (`x>5`) se satisface en cada componente del arreglo `x`.

11. **Bonus Track (opcional):** Las funciones `any()` y `all()`, al ser aplicadas a un *arreglo de booleanos* indican si *alguna* o *todas* las componentes del arreglo son verdaderas, respectivamente. Usando el arreglo `x` definido en el punto anterior, entonces

```
any(x>5)
```

es verdadero porque sí existe una componente del arreglo `x` que es mayor que 5, en cambio

```
any(x>10)
```

es falso. Verifique en forma similar el comportamiento de `all()` (por ejemplo, viendo qué entregan los comandos `all(x>5)` y `all(x>=0)`).