

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

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:

En otras palabras, `np.arange(10)` es equivalente a `np.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 considerada en el problema 9 de la guía 8, es decir,

$$1 + 2 + 3 + 4 + \cdots + 999 + 1000, \quad (1)$$

pero ahora usando la función `sum` de NumPy (que suma todos los elementos de un arreglo):

o, en una sola línea

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

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:

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

8. Considere el archivo de datos `datos.txt`. NumPy contiene una función llamada `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:

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. Los arreglos de NumPy también pueden contener componentes con valores booleanos (siempre que todos los componentes lo sean). Por ejemplo,

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. Las funciones `np.any()` y `np.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

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

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