Importar librerías

```
import numpy
B = numpy.sin(3)
```

Importamos la librería junto a un "alias" para usarla cómodamente

```
import numpy as np
C = np.sin(3)
```

De la librería solo importamos la función "sin

```
from nump"y import sin
D = sin(3)
D2 = cos(3) # En este caso, la función del coseno no funciona, pues no la hemos importado
```

Importamos todas las funciones de la librería "una a una". Podemos usar todas las funciones sin nombrar la librería. No recomendable para programas complejos

```
from numpy import *
E = sin(3)
E2= cos(3)
E3 = pi
```

Algunas funciones de numpy

```
A = np.pi

B = np.sin()    # Seno
C = np.cos()    # Coseno
D = np.tan()    # Tangente
# Existen sus variantes hiperbólicas
# OJO: estas funciones trabajan en RADIANES

B2= np.arcsin(1)    # Arco seno
C2 = np.arccos(0)   # Arco coseno
D2 = np.arctan(0)   # Arco tangente
A = np.exp(1)       // e^1
B = np.exp2(3)      // 2^3

A2 = np.log(1)      // ln(1)
B2 = np.log2(3)     // log(3) (Base 2)
C2 = np.log10(4)    // log(4) (Base 10)
```

Vectores

Vector básico de 3 dimensiones

```
v = np.array([ 1 , 2 , 3])

v.shape      // Dimensiones de la matriz
v.size       // Número total de elementos
v.ndim       // Número de dimensiones

v[0]         // Primera componente
v[1]         // Segunda componente.
```

```
u = np.zeros(4)         // [ 0 , 0 , 0 , 0 ]
w = np.ones(4)          // [ 1 , 1 , 1 , 1 ]
z = np.random.rand(3)   // Valores aleatorios
```

```
list = [ 2 , 5 , 12]
v2 = array(list)
```

Vectores de valores consecutivos y equiespaciados

Vector de números entre el primer y segundo valor. Espaciados según el tercer argumento

```
V1 = np.arange(0 , 7 , 2)
```

Vector de tantos números igualmente espaciados como dice el tercer argumento, entre el primer y segundo valor

```
V2 = np.linspace( 1 , 8 , 3 )
```

```
a = np.array([1, 2])
b = np.array([3, 4])
np.concatenate([a, b])    // [1, 2, 3, 4]
np.vstack([a, b])         // [[1, 2], [3, 4]]
np.hstack([a, b])         // [1, 2, 3, 4]
```

Operaciones

```
a = np.array([1, 2, 3])
b = np.array([4, 5, 6])

A = 10*a             // Producto de todos los elementos de a por un escalar
SUMA = a + b         // [5, 7, 9]
RESTA = a - b        // [-3, -3, -3]
PROD = a * b         // [4, 10, 18] (elemento a elemento)
DIV = a / b          // [0.25, 0.4, 0.5]
P_ESC = a @ b        // Producto escalar
```

Concatenación y apilamiento

```
a = np.array([1, 2])
b = np.array([3, 4])

np.concatenate([a, b])    // [1, 2, 3, 4]
np.vstack([a, b])         // [[1, 2], [3, 4]]
np.hstack([a, b])         // [1, 2, 3, 4]
```

Índices y condiciones

```
x = np.array([3, 7, 1, 9, 2])

np.where(x > 3)        # Índices donde se cumple la condición → (array([1, 3]),)
np.where(x > 3, 1, 0)  # Máscara binaria → [0, 1, 0, 1, 0]

np.nonzero(x)          # Índices de elementos no nulos
np.unique(x)           # Valores únicos ordenados
np.sort(x)             # Ordenar array
np.argsort(x)          # Índices que ordenarían el array
```

Trozeado de vectores y listas

Esta "función" tambien se puede utilizar con las listas de Python y otros elementos

```
a = np.random.rand(5)
print(f"Vector base: {a}")

B1 = a[:]       # Muestra el conjunto entero, sin cambios

B2 = a[1:3]     # Muestra desde el índice 1 al 3 ( sin incluirlo ).

B3 = a[2:-1]    # Muestra del índice 2 al penúltimo

B4 = a[2:]      # Muestra del índice 2 al último

# Recuerda cómo son los índices; [ 0 , 1 , 2 , ...] / [ ... , -2 , -1 ]

B5 = a[:3]      # Muestra del inicio al índice 3

B6 = a[4:]      # Muestra del índice 4 al final
```

Funciones estadísticas

```
x = np.array([1, 2, 3, 4, 5])

np.mean(x)      // Media
np.median(x)    // Mediana
np.std(x)       // Desviación típica
np.var(x)       // Varianza
np.min(x)       // Valor mínimo
np.max(x)       // Valor máximo
np.sum(x)       // Suma total
```

Álgebra lineal, `numpy.linalg`

```
A = np.array([[1, 2], [3, 4]])

np.linalg.det(A)       # Determinante
np.linalg.inv(A)       # Inversa

np.linalg.solve(A, [5, 6])  # Resolver sistema Ax=b
```