Ricardo Pérez Aragoneses

Ejercicios de Python

Ejercicio 1: Sumar dos arrays elemento por elemento

Enunciado: Crea dos arrays de Numpy de tamaño 5, llenos de números enteros. Luego, calcula la suma de ambos arrays elemento por elemento y muestra el resultado.

Motivo para usar arrays: Con Numpy, podemos sumar directamente dos arrays usando la operación +, sin necesidad de recorrer cada elemento manualmente como tendríamos que hacer con listas.

```
#Importamos la biblioteca numpy y le ponemos un alias

import numpy as np

#Creamos los dos arrays

array_1=np.array([10,20,30,40,50])

array_2=np.array([1,2,3,4,5])

#Sumamos los dos arrays(10+1, 20+2, etc...)

array_sumado = array_1 + array_2

#Mostramos el resultado sumado

print(array_sumado)
```

Resultado:

```
/ej1.py
[11 22 33 44 55]
PS C:\programacion\python\ejercicios-python\arrays\ejercicio-09>
```

Ejercicio 2: Multiplicar cada elemento de un array por un número

Enunciado: Crea un array de Numpy de tamaño 6 con valores enteros de tu elección. Luego, multiplica cada elemento del array por 3 y muestra el resultado.

Motivo para usar arrays: La multiplicación escalar con arrays es mucho más eficiente en Numpy, ya que la operación se aplica directamente a cada elemento sin necesidad de un bucle.

```python
#Importamos la biblioteca numpy y le ponemos un alias

import numpy as np

#Creamos el array

array_mult=np.array([10,2,5,9,20,15])

#Multiplicamos cada num del array POR 3

multipilicacion= array_mult * 3

print(multipilicacion)
```

Resultado:

```
[30  6 15 27 60 45]
PS C:\programacion\python\ejercicios-python\arrays\ejercicio-09>
```

Ejercicio 3: Calcular el promedio de un array

Enunciado: Crea un array de Numpy con 10 números enteros de tu elección. Calcula el promedio de los elementos y muestra el resultado.

Motivo para usar arrays: Numpy permite calcular el promedio de los elementos de un array con la función `np.mean()` de forma rápida y eficiente, algo que con listas requeriría escribir más código.

```
#Importamos la biblioteca numpy y le ponemos un alias

import numpy as np

#np.mean para calcular el promedio de todos los valores en total

array_prom=np.mean([100, 90, 80, 70, 60, 50, 40, 30 ,20 ,10])

print(array_prom)
```

Resultado:

```
55.0
PS C:\programacion\python\ejercicios-python\arrays\ejercicio-09>
```

Ejercicio 4: Filtrar elementos mayores a un valor dado

Enunciado: Crea un array de Numpy con 8 números enteros. Luego, crea un nuevo array que solo contenga los elementos mayores a 5 y muéstralo.

Motivo para usar arrays: Con Numpy, puedes aplicar filtros de manera muy rápida usando operaciones de comparación en una sola línea. Filtrar listas de esta forma requeriría escribir bucles y condicionales adicionales.

```python
#Importamos la biblioteca numpy y le ponemos un alias

import numpy as np

#Creamos el array

array_1=np.array([1, 2, 3, 4, 5, 6, 7, 8])

#filtramos para que muestre solo los mayores de 5

filtro = array_1[array_1 > 5]

print(filtro)
```

Resultado:

```
[6 7 8]
PS C:\programacion\python\ejercicios-python\arrays\ejercicio-09>
```

Ejercicio 5: Elevar al cuadrado cada elemento de un array

Enunciado: Crea un array de Numpy con 5 elementos enteros. Calcula el cuadrado de cada elemento y muestra el resultado.

Motivo para usar arrays: Numpy permite aplicar operaciones matemáticas, como la potenciación, a cada elemento de un array en una sola operación. Esto es mucho más eficiente que hacer un bucle para elevar cada elemento al cuadrado en una lista.

```python
#Importamos la biblioteca numpy y le ponemos un alias

import numpy as np

array_elevado=np.array([10,20,30,40,50])

#Elevamos cada numero del array al cuadrado con "**"

al_cuadrado = array_elevado ** 2

print (al_cuadrado)
```

Resultado:

```
[ 100  400  900 1600 2500]
PS C:\programacion\python\ejercicios-python\arrays\ejercicio-09>
```