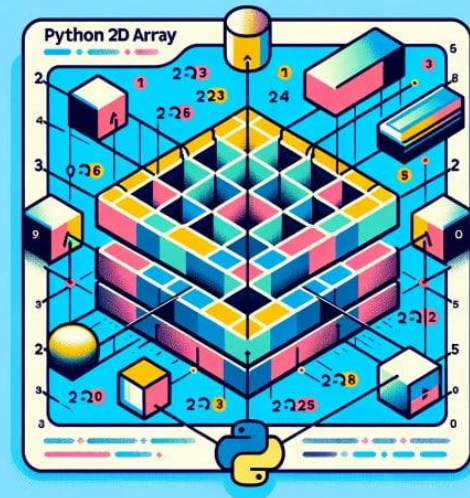


Arreglos bidimensionales: Matrices



¿Qué es una matriz?

Es una estructura de datos bidimensional que contiene elementos dispuestos en filas y columnas formando una tabla.

		Columnas			
		0	1	2	3
Filas	0	2	4	5	8
	1	6	3	1	9

Matrices en Python: Una de las formas que nos ofrece Python para representar matrices es con listas anidadas.

¿Cómo se componen?

Las matrices se componen por listas internas que representan filas donde cada uno de los elementos de esa lista interna compone los elementos individuales de las columnas de esa matriz.

Tipos de datos: Pueden contener tipos de datos básicos, como números y cadenas de texto, también podemos incluir otras estructuras como listas, diccionarios u objetos personalizados creados por el programador.

Nota: Tener en cuenta que Python también permite almacenar distintos tipos de datos en una misma lista/matriz. Por ejemplo: una matriz donde cada fila representa una persona y cada columna los datos individuales de esa persona como nombre (str), edad (int), estado (bool), etc.


Cómo declaramos una matriz en Python:

		Columnas			
		0	1	2	3
Filas	0	2	4	5	8
	1	6	3	1	9




```
matrix = [[2, 4, 5, 8],  
          [6, 3, 1, 9]]
```

Cómo mostrar una matriz por terminal :



```
matrix = [[2, 4, 5, 8],  
          [6, 3, 1, 9]]  
  
print(matrix)
```



```
for i in range(len(matrix)):  
    for j in range(len(matrix[i])):  
        print(matrix[i][j], end=" ")  
    print("")
```



```
[[2, 4, 5, 8], [6, 3, 1, 9]]
```



```
2 4 5 8  
6 3 1 9
```

Cómo mostrar un elemento de la matriz :

Posición: [índice fila] [índice columna]

		Columnas			
		0	1	2	3
Filas	0	2	4	5	8
	1	6	3	1	9

```
print(matrix[1][0])
```



6

Cómo modificar un elemento de la matriz :

Posición: [índice fila] [índice columna]

		Columnas			
		0	1	2	3
Filas	0	2	4	5	8
	1	6	3	1	9



```
print(matrix[1][0]) # 6
```

```
matrix[1][0] = 15
```

```
print(matrix[1][0]) # 15
```


Cómo crear e inicializar una matriz en Python :

```
def inicializar_matriz(cantidad_filas:int, cantidad_columnas:int, valor_inicial:any) -> list:  
    matriz = []  
    for i in range(cantidad_filas):  
        fila = [valor_inicial] * cantidad_columnas  
        matriz += [fila]  
    return matriz  
  
mi_matriz = inicializar_matriz(2, 4, 0)
```



```
[0, 0, 0, 0]  
[0, 0, 0, 0]
```

Carga secuencial :

Es el proceso de agregar valores a una matriz de manera ordenada, desde la primera celda hacia la última.

```
def cargar_matriz_secuencialmente(matriz:list):  
    # Agregar las validaciones/retorno que sean necesarias  
    for i in range(len(matriz)):  
        for j in range(len(matriz[i])):  
            matriz[i][j] = int(input(f"Fila {i} Columna {j}: "))  
  
cargar_matriz_secuencialmente(mi_matriz)
```

Carga aleatoria :

```
def cargar_matriz_aleatoriamente(matriz:list):  
    # Agregar las validaciones/retorno que sean necesarias  
    seguir = "S"  
    while seguir == "S":  
        fila = int(input("Indice de fila: "))  
        columna = int(input("Indice de columna: "))  
        dato = int(input("Ingrese el número a cargar: "))  
        matriz[fila][columna] = dato  
        seguir = input("Desea seguir cargando? S/N: ")  
  
cargar_matriz_aleatoriamente(mi_matriz)
```

Es posible cargar valores a una matriz inicializada de forma aleatoria, indicando el valor a incorporar y su ubicación en la matriz (fila y columna).

Búsqueda en matrices :

```
mi_matriz = [[2, 4, 5, 8],  
             [6, 3, 1, 9]]  
  
def buscar_valor_entero(matriz:list, valor:int):  
    for i in range(len(matriz)):  
        for j in range(len(matriz[i])):  
            if matriz[i][j] == valor:  
                print(f"Se encontró el número en fila {i} columna {j}!")  
  
buscar_valor_entero(mi_matriz, 5)
```

Mediante las búsquedas tenemos la posibilidad de comprobar si el dato existe en la matriz y además obtener su ubicación exacta.

Se encontró el número en fila 0 columna 2!



Suma de matrices :

$$\begin{bmatrix} 3 & 8 \\ 4 & 6 \end{bmatrix} + \begin{bmatrix} 4 & 0 \\ 1 & -9 \end{bmatrix} = \begin{bmatrix} 7 & 8 \\ 5 & -3 \end{bmatrix}$$

The diagram illustrates the addition of two 2x2 matrices. A yellow curved arrow connects the element 3 in the first matrix to the element 7 in the resulting matrix. Another yellow curved arrow connects the element 4 in the second matrix to the same element 7 in the resulting matrix. Above these arrows, the calculation $3 + 4 = 7$ is shown in orange text. The elements 3, 4, and 7 are highlighted with yellow circles.

Multiplicación matriz - escalar :

$$k \times \begin{bmatrix} a & b \\ c & d \end{bmatrix} = \begin{bmatrix} a \cdot k & b \cdot k \\ c \cdot k & d \cdot k \end{bmatrix}$$

Multiplicación de Matrices

Mat 1 Mat 2 Mat Resultante

3	7	5
12	2	11

x

3	16
1	4
4	22

=

36	

Video explicativo: [enlace](#).

Contenido adicional :



Matrices (en inglés): [Geek for geeks](#)



Algoritmos matemáticos: [Apunte](#)