

NIVEL 4

MATRICES





¿QUÉ SON LAS MATRICES?

Las matrices son estructuras bidimensionales de datos dentro de las cuales los valores se organizan en filas y columnas

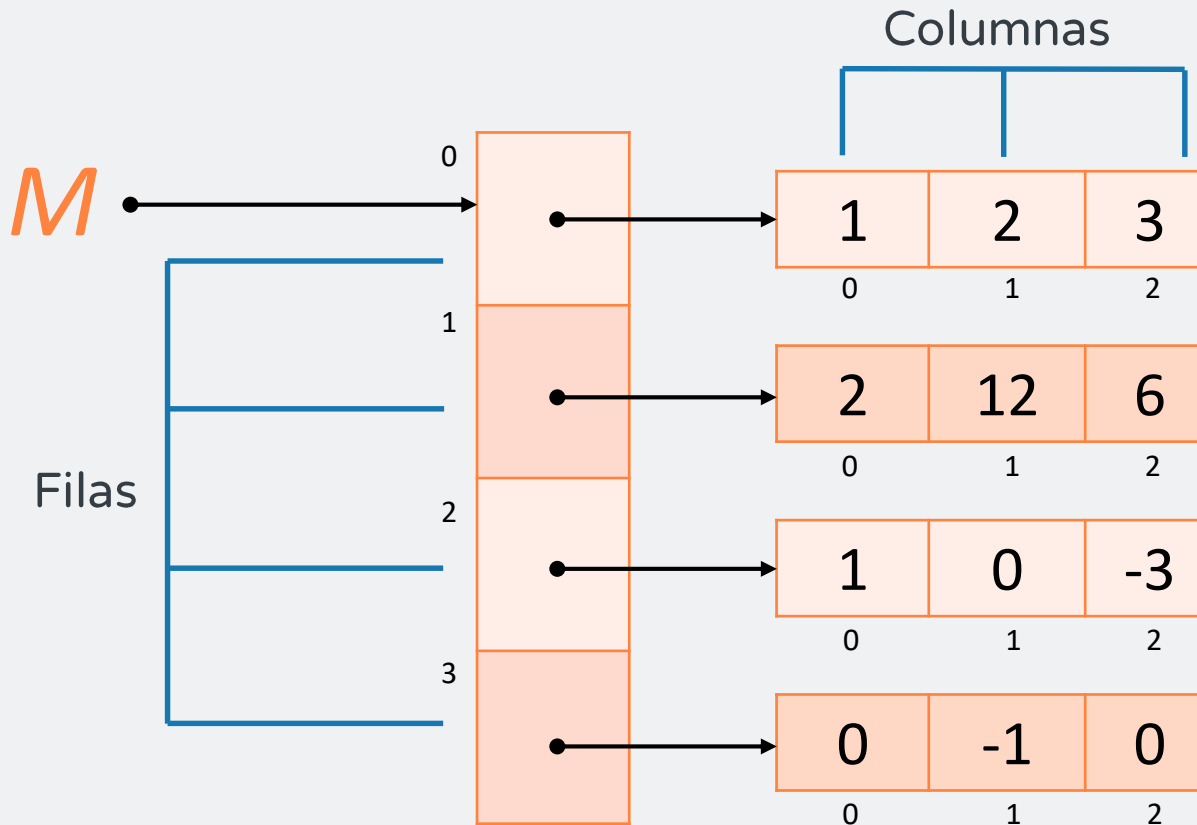
$$M = \begin{array}{|c|c|c|} \hline 1 & 2 & 3 \\ \hline 2 & 12 & 6 \\ \hline 1 & 0 & -3 \\ \hline 0 & -1 & 0 \\ \hline \end{array}$$

Esta matriz tiene **4 filas** y **3 columnas**, lo cual abreviamos diciendo que es una matriz de **dimensión 4 × 3**

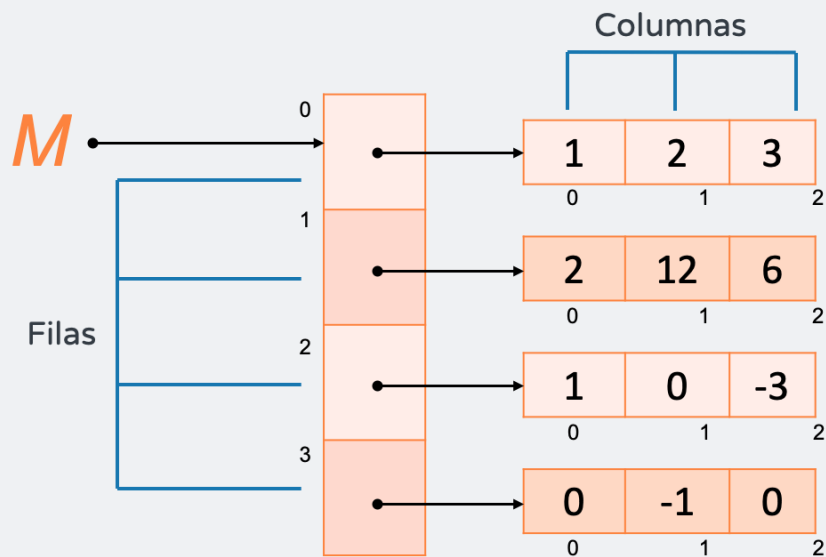
¿CÓMO SE REPRESENTAN LAS MATRICES EN PYTHON?

```
Console 1/A  
In [4]: M = [ [1, 2, 3], [2, 12, 6], [1, 0, -3], [0, -1, 0] ]
```

Con una lista de listas !!!




PARA ACCEDER A ELEMENTOS DE LA MATRIZ SE USA UNA DOBLE INDEXACIÓN



```
Console 1/A  
In [4]: M = [ [1, 2, 3], [2, 12, 6], [1, 0, -3], [0, -1, 0] ]  
  
In [5]: M[0][0]  
Out[5]: 1  
  
In [6]: M[0][1]  
Out[6]: 2  
  
In [7]: M[0][2]  
Out[7]: 3  
  
In [8]: M[1][0]  
Out[8]: 2  
  
In [9]: M[1][1]  
Out[9]: 12
```

1. ¿Qué resulta de ejecutar este programa?

```
Ejemplo1Matrices.py   
1 M = [ [1, 0, 0], [0, 1, 0], [0, 0, 1] ]  
2 print(M[-1][0])  
3 print(M[-1][-1])  
4 print("-")  
5 for i in range(0,3):  
6     print(M[i])  
7 print("-")  
8 for i in range(0,3):  
9     for j in range(0,3):  
10        print(M[i][j])
```



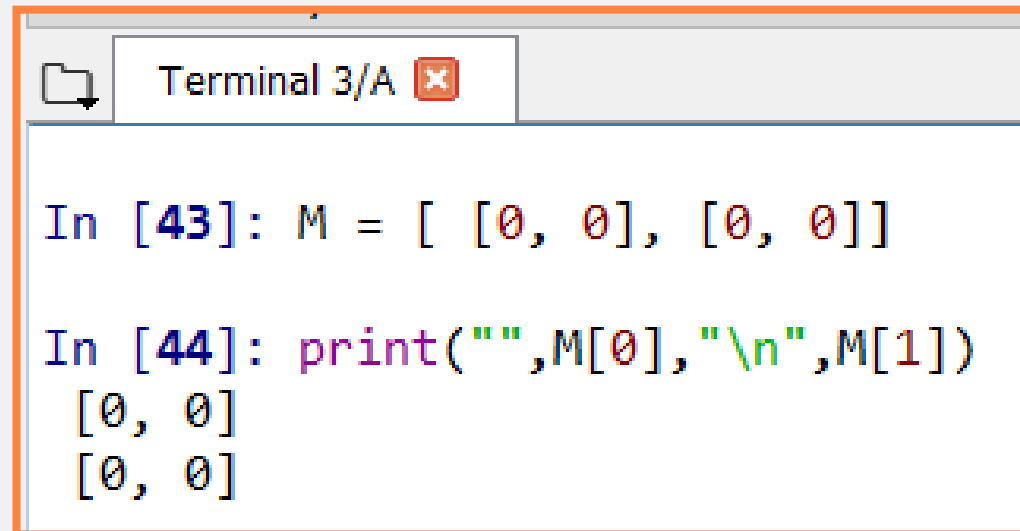
2. ¿Qué resulta de ejecutar este programa?

```
Ejemplo2Matrices.py ✕  
1 M = [ [1, 0, 0], [0, 1, 0], [0, 0, 1] ]  
2 s = 0.0  
3  
4 for i in range(0,3):  
5     for j in range(0,3):  
6         s += M[i][j]  
7  
8 print (s/9)
```



CREACIÓN DE MATRICES

Crear una matriz consiste, pues, en crear una lista de listas. Si deseamos crear una **matriz nula** (una matriz cuyos componentes sean todos igual a 0) de tamaño **2×2**, bastará con escribir:



```
Terminal 3/A ✕  
  
In [43]: M = [ [0, 0], [0, 0]  
  
In [44]: print("",M[0],"\\n",M[1])  
[0, 0]  
[0, 0]
```

CREACIÓN DE MATRICES

Parece sencillo, pero ¿y si nos piden una **matriz nula de 6×6** ? Tiene 36 componentes y escribirlos explícitamente resulta muy tedioso. ¡Y pensemos en lo inviable de definir así una matriz de dimensión **10×10 o 100×100** !

¿Qué hacer?



CREACIÓN DE MATRICES

Recordemos que hay una forma de crear listas de cualquier tamaño, siempre que tengan el mismo valor, utilizando el operador `*`

Como una matriz es una lista de listas, si creamos una lista con 3 duplicados de la lista `a`, obtenemos la matriz de **3 x 6 !!!**

```
Terminal 3/A ✕  
  
In [46]: a = [0] * 6  
  
In [47]: a  
Out[47]: [0, 0, 0, 0, 0, 0]
```

```
Terminal 3/A ✕  
  
In [51]: a = [0] * 6  
  
In [52]: M = [a] * 3  
  
In [53]: M  
Out[53]: [[0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0]]  
  
In [54]: print("",M[0],"\n",M[1],"\n",M[2])  
[0, 0, 0, 0, 0, 0]  
[0, 0, 0, 0, 0, 0]  
[0, 0, 0, 0, 0, 0]
```

CREACIÓN DE MATRICES

¿Qué pasa si ahora hacemos `M[0][0] = 1`?

```
Terminal 3/A ✕  
In [55]: M[0][0] = 1  
  
In [56]: M  
Out[56]: [[1, 0, 0, 0, 0, 0], [1, 0, 0, 0, 0, 0], [1, 0, 0, 0, 0, 0]]  
  
In [57]: print("",M[0],"\n",M[1],"\n",M[2])  
[1, 0, 0, 0, 0, 0]  
[1, 0, 0, 0, 0, 0]  
[1, 0, 0, 0, 0, 0]
```



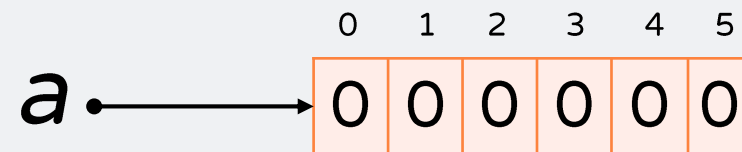
No se ha modificado únicamente el elemento 0 de la primera lista, sino todos los elementos 0 de todas las listas de la matriz!!!

CREACIÓN DE MATRICES

Entendamos qué fue lo que pasó, paso por paso ...

- Creamos la lista `a`:

```
Terminal 3/A [X]  
  
In [58]: a = [0] * 6  
  
In [59]: a  
Out[59]: [0, 0, 0, 0, 0, 0]
```

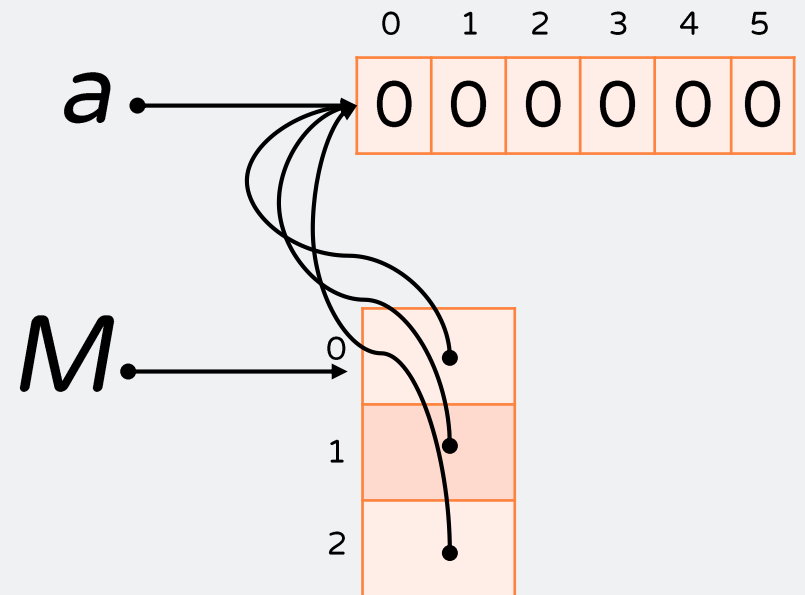


CREACIÓN DE MATRICES

Creemos la lista **M** como la copia por triplicado de la lista **a**:

```
Terminal 3/A ✕  
  
In [61]: a = [0] * 6  
  
In [62]: M = [a] * 3  
  
In [63]: M  
Out[63]: [[0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0]]
```

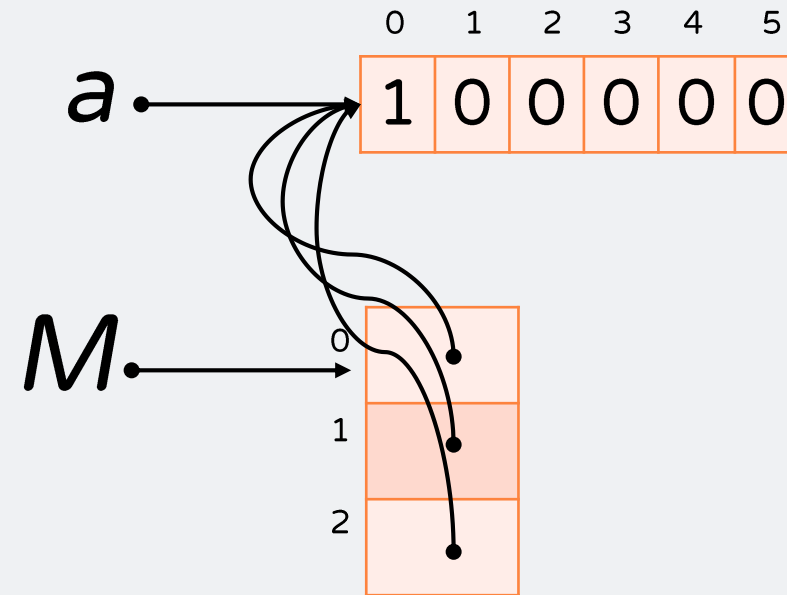
Python nos ha obedecido **copiando tres veces** ¡la referencia a dicha lista!



CREACIÓN DE MATRICES

Como modificamos el elemento $M[0][0]$ asignándole el valor 1:

```
Terminal 3/A  
  
In [65]: a = [0] * 6  
In [66]: M = [a] * 3  
In [67]: M[0][0] = 1
```



Hemos modificado también $M[1][0]$ y $M[2][0]$,
pues son el mismo elemento

CREACIÓN DE MATRICES

ATENCIÓN: hay que construir matrices con más cuidado, asegurándonos de que cada fila es una lista diferente de las anteriores !!!



```
Terminal 3/A x

In [69]: M = []

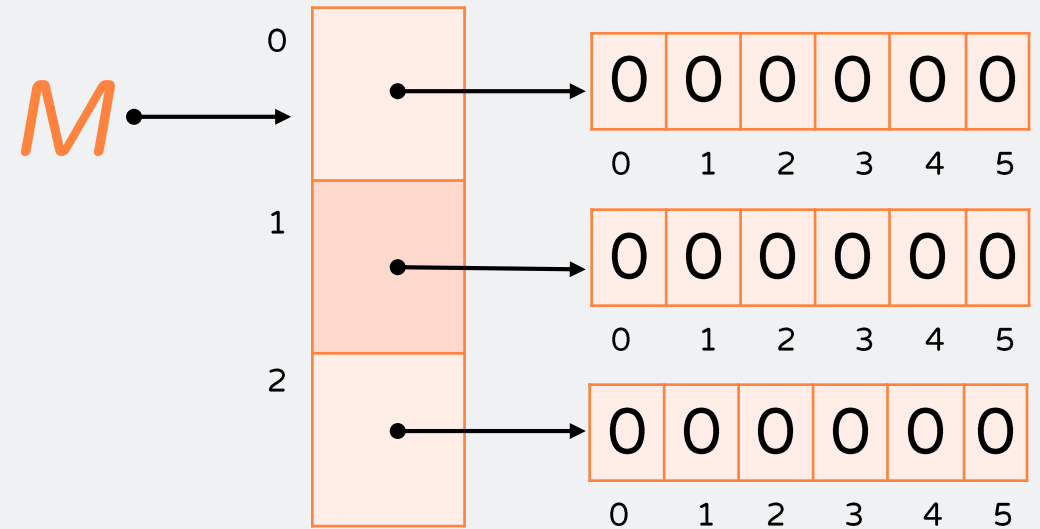
In [70]: for i in range(0,3):
...:     a = [0] * 6
...:     M.append(a)
...:

In [71]: print(M)
[[0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0]]
```

CREACIÓN DE MATRICES

```
Terminal 3/A ✕  
  
In [69]: M = []  
  
In [70]: for i in range(0,3):  
...:     a = [0] * 6  
...:     M.append(a)  
...:  
  
In [71]: print(M)  
[[0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0]]
```

La lista creada en la asignación `a = [0] * 6` es diferente con cada iteración, así que estamos añadiendo a `M` una lista nueva cada vez. La memoria queda así:



CREACIÓN DE MATRICES

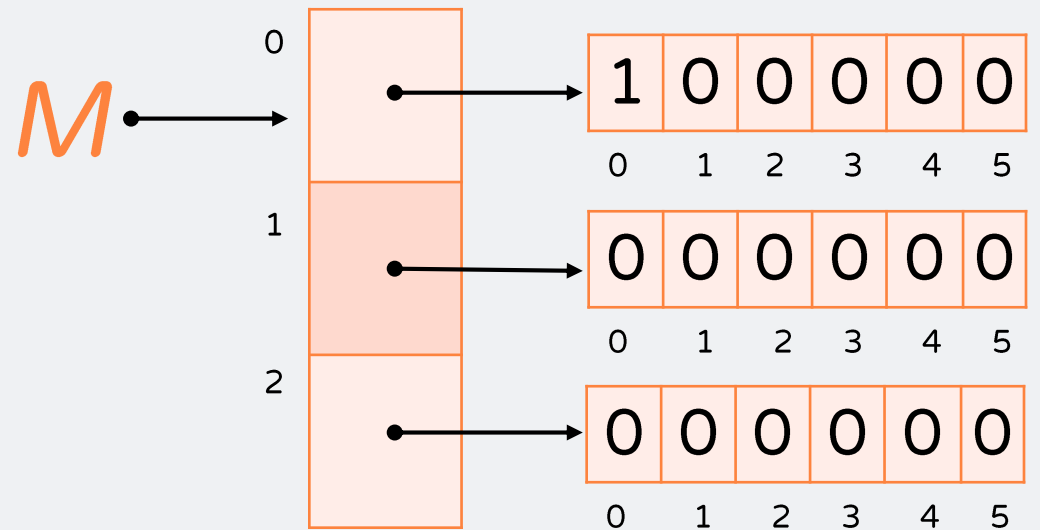
Terminal 3/A

```
In [72]: M[0][0] = 1
```

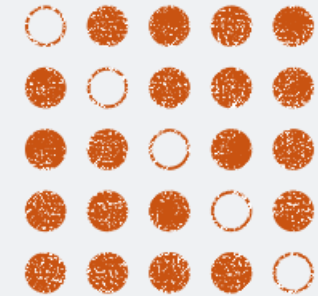
```
In [73]: print(M)
```

```
[[1, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0]]
```

Modificar ahora una casilla de una fila no afecta a las demás:



Escriba un programa que pida al usuario un entero positivo n y almacene en una variable M la matriz identidad de $n \times n$ (la que tiene unos en la diagonal principal y ceros en el resto de celdas).



LECTURA DE MATRICES

Si deseamos leer una matriz de tamaño determinado, podemos **crear una matriz nula** como hemos visto anteriormente y, a continuación, **rellenar** cada uno de sus componentes

```
EjemploLecturaMatrices.py x
1 #Pedimos la dimensión de la matriz
2 m = int(input("Digite el número de filas: "))
3 n = int(input("Digite el número de columnas: "))
4
5 #Creamos una matriz nula...
6 M = []
7 for i in range(0,m):
8     M.append([0] * n)
9
10 # ... y leemos su contenido de teclado
11 for i in range(0, m):
12     for j in range (0, n):
13         M[i][j] = float(input("Digite el contenido de la casilla ["+str(i)+"]["+str(j)+"]: "))
14
15 print("La matriz completa es: ")
16
17 for i in range(0, m):
18     print(M[i])
```

¿QUÉ DIMENSIÓN TIENE UNA MATRIZ?

```
Terminal 4/A [X]  
  
In [10]: a = [ [1, 0], [0, 1], [0, 0] ]  
  
In [11]: len(a)  
Out[11]: 3  
  
In [12]: len(a[0])  
Out[12]: 2
```

`len(a)` devuelve el número de filas

`len(a[0])` devuelve el número de columnas

Escriba las siguientes funciones que reciben por parámetro una matriz de enteros y retornan:

1. La suma de todos los valores
2. La suma de los valores de una fila dada por parámetro
3. La suma de los valores de una columna dada por parámetro
4. Indique verdadero o falso si en la matriz hay al menos un valor negativo
5. La fila con el mayor número de ceros

