Y el segundo índice accede a un elemento de esa lista, que es un entero:

```
>>> M[0][0] ↲
1
```

EJERCICIOS

▶ 244 Una matriz nula es aquella que sólo contiene ceros. Construye una matriz nula de 5 filas y 5 columnas.

▶ 245 Una matriz identidad es aquella cuyos elementos en la diagonal principal, es decir, accesibles con una expresión de la forma `M[i][i]`, valen uno y el resto valen cero. Construye una matriz identidad de 4 filas y 4 columnas.

▶ 246 ¿Qué resulta de ejecutar este programa?

```
M = [ [1, 0, 0], [0, 1, 0], [0, 0, 1] ]
print M[-1][0]
print M[-1][-1]
print '--'
for i in range(0, 3):
    print M[i]
print '--'
for i in range(0, 3):
    for j in range(0, 3):
        print M[i][j]
```

▶ 247 ¿Qué resulta de ejecutar este programa?

```
M = [ [1, 0, 0], [0, 1, 0], [0, 0, 1] ]
s = 0.0
for i in range(0, 3):
    for j in range(0, 3):
        s += M[i][j]
print s / 9.0
```

5.4.1. Sobre la 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:

```
>>> m = [ [0, 0], [0, 0] ] ↲
```

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!

Recuerda que hay una forma de crear listas (vectores) de cualquier tamaño, siempre que tengan el mismo valor, utilizando el operador `*`:

```
>>> a = [0] * 6 ↲
>>> a ↲
[0, 0, 0, 0, 0, 0]
```

Si una matriz es una lista de listas, ¿qué ocurrirá si creamos una lista con 3 duplicados de la lista `a`?

```
>>> [a] * 3 ↲
[[0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0]]
```

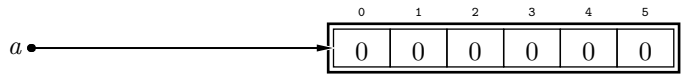
¡Estupendo! Ya tenemos una matriz nula de 3×6. Trabajemos con ella:

```
>>> a = [0] * 6 ↲
>>> M = [a] * 3 ↲
>>> M[0][0] = 1 ↲
>>> print M ↲
[[1, 0, 0, 0, 0, 0], [1, 0, 0, 0, 0, 0], [1, 0, 0, 0, 0, 0]]
```

¿Qué ha ocurrido? ¡No se ha modificado únicamente el componente 0 de la primera lista, sino *todos* los componentes 0 de todas las listas de la matriz!

Vamos paso a paso. Primero hemos creado a:

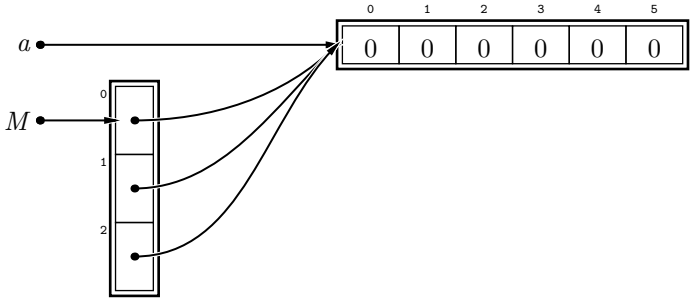
```
>>> a = [0] * 6 ↲
```



A continuación hemos definido la lista M como la copia por triplicado de la lista a:

```
>>> M = [a] * 3 ↲
```

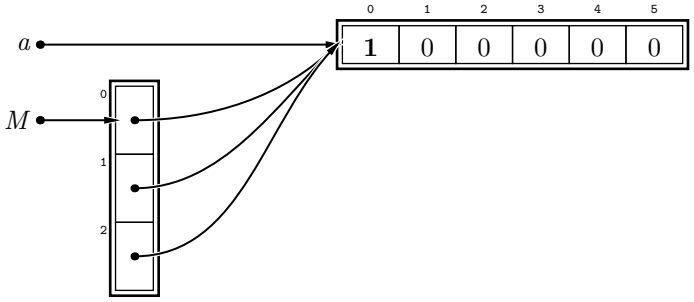
Python nos ha obedecido copiando tres veces... ¡la referencia a dicha lista!:



Y hemos modificado el elemento `M[0][0]` asignándole el valor 1:

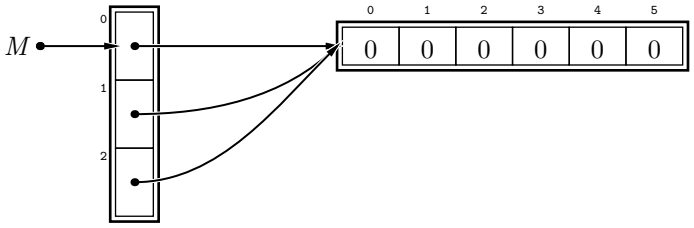
```
>>> M[0][0] = 1 ↲
```

así que hemos modificado también `M[1][0]` y `M[2][0]`, pues *son el mismo elemento*:



Por la misma razón, tampoco funcionará este modo más directo de crear una matriz:

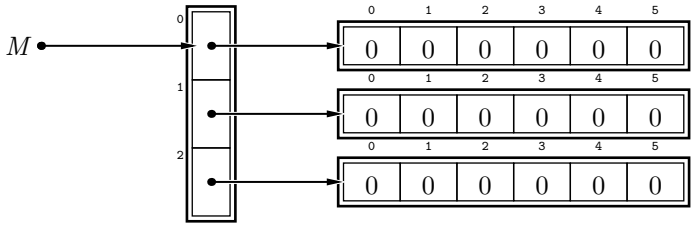
```
>>> M = [ [0] * 6 ] * 3 ↲
```



Hay que construir matrices con más cuidado, asegurándonos de que cada fila es una lista diferente de las anteriores. Intentémoslo de nuevo:

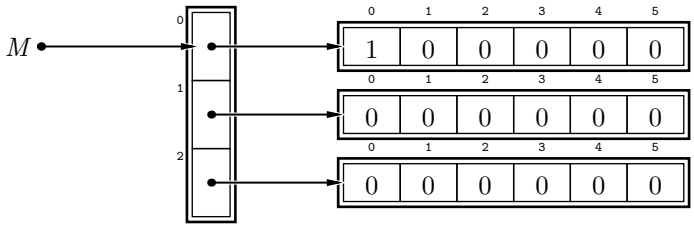
```
>>> M = [] ↲
>>> for i in range(3): ↲
...     a = [0] * 6 ↲
...     M.append( a ) ↲
... ↲
>>> print M ↲
[[0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0]]
```

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:



Lo cierto es que no es necesario utilizar la variable auxiliar a:

```
>>> M = [] ↲
>>> for i in range(3): ↲
...     M.append( [0] * 6 ) ↲
... ↲
>>> print M ↲
[[0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0]]
>>> M[0][0] = 1 ↲
>>> print M ↲
[[1, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0]]
```



EJERCICIOS

▶ 248 Crea la siguiente matriz utilizando la técnica del bucle descrita anteriormente.

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

▶ 249 Haz un programa que pida 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).