Mas sobre listas de Python: Matrices

Introducción a la Computación

Clase 14

Patricia Borensztejn

Recordemos

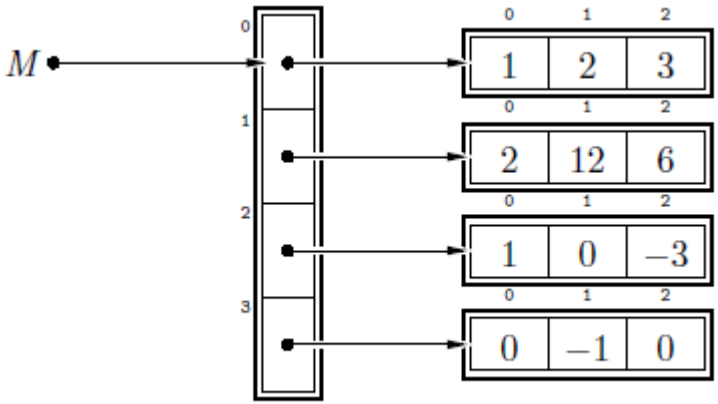
- Las listas en Python se parecen a los vectores de C, pero son dinámicos siempre.
- O sea que, podemos comenzar con una lista vacía simbolizada [] e ir agregándole elementos sin necesidad de invocar a ninguna función malloc! Usamos el operador de concatenación o el método append.
- Las listas, a pesar de que son estructuras heterogéneas, es decir, que pueden almacenar cualquier tipo de elemento, se utilizan para almacenar estructuras homogéneas. Ejemplo: listas de enteros, listas de alumnos, etc….

Matrices

- Para representar una matriz en Python usamos listas de listas. Así:

$$M = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 12 & 6 \\ 1 & 0 & -3 \\ 0 & -1 & 0 \end{pmatrix}$$

```
>>> M = [ [1, 2, 3], [2, 12, 6], [1, 0, -3], [0, -1, 0] ]
```



Matrices

- Si queremos acceder al elemento que está en la fila 0, columna 1 hacemos:

```
>>> M[0][1] ↲
2
```

- Y si ponemos :

```
>>> M[0] ↲
[1, 2, 3]
```

  - Entonces Python devuelve una lista..
  - ¿Que pasaría en C si ponemos solo la primera dimensión?

Creación de Matrices

- Si quiero una matriz de 3*3 inicializada a cero:

```
>>> m = [ [0, 0], [0, 0] ]
```

- Si es de 100x100? Usamos el operador de repetición *

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

Aquí tenemos una matriz de 3x6

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

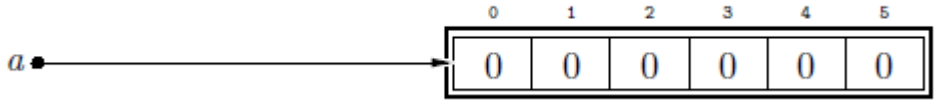
Creación de Matrices

```
>>> 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]]
```

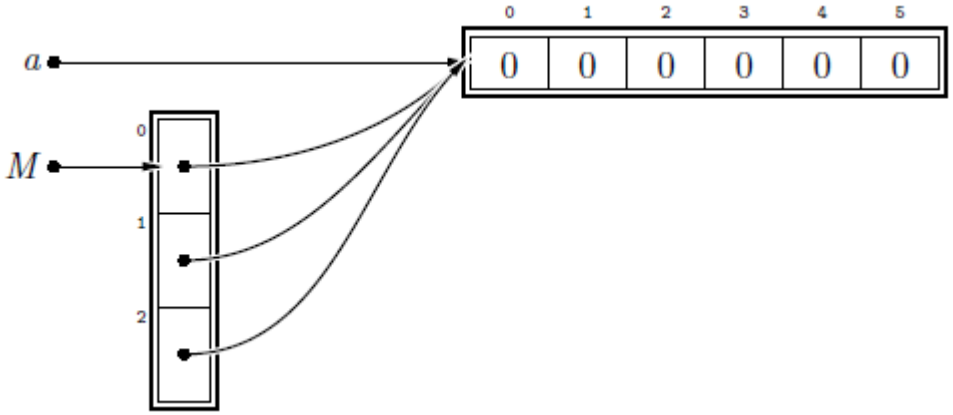
- Algo no está bien… ¿porqué?

Creación de Matrices

- La sentencia : a=[0]*6 crea una lista de seis elementos (todos ellos valen o apuntan al cero).



- La sentencia : M=[a]*3 crea una lista de 3 elementos, todos ellos apuntan a la lista (única)



Creación de Matrices

- La sentencia : M[0][0]=1 modifica las tres filas porque son la misma!
- Sin embargo, si asignamos nuevos valores a una fila… veamos que sucede…¿Qué pasó ahora?

```
>>> lista=[0]*6
>>> lista
[0, 0, 0, 0, 0, 0]
>>> M=[lista]*3
>>> M
[[0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0]]
>>> M[1]=[2,2,2,2,2,2]
>>> M
[[0, 0, 0, 0, 0, 0], [2, 2, 2, 2, 2, 2], [0, 0, 0, 0, 0, 0]]
```

Creación de Matrices

- Si ahora modificamos el elemento M[0][0]…

```
>>> M[0][0]=1
>>> M
[[1, 0, 0, 0, 0, 0], [2, 2, 2, 2, 2, 2], [1, 0, 0, 0, 0, 0]]
>>>
```

- Hagamos un dibujo de la matriz M después de ejecutar el código anterior….

Creación de Matrices

- Para estar seguros que una matriz tiene asignada memoria para cada una de las filas:

```
>>> 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]]
```

Ejemplo: Leer una matriz

```
1  # Pedimos la dimensión de la matriz,
2  m = int(raw_input('Dime␣el␣número␣de␣filas:␣'))
3  n = int(raw_input('Dime␣el␣número␣de␣columnas:␣'))
4
5  # Creamos una matriz nula...
6  M = []
7  for i in range(m):
8      M.append( [0] * n )
9
10 # ... y leemos su contenido de teclado
11 for i in range(m):
12     for j in range(n):
13         M[i][j] = float(raw_input('Dame␣el␣componente␣(%d,%d):␣' % (i, j)))
```

Ejercicios Matrices:

- *Problema 1:* Hacer un programa en Python que lea dos matrices A y B de NxM y calcule e imprima por pantalla su suma C tal que:
  - C[i][j]=A[i][j]+B[i][j] para todo i<N,j<M
- *Problema 2:* Hacer un programa que lea una matriz A de NxM y calcule su traspuesta.
- Nota: Para ambos problemas las dimensiones también deberán leerse de la entrada estándar.