Ing. Ventre, Luis O.

CAPITULO 5:

SECUENCIAS CONT.

MATRICES

Tipos ESTRUCTURADOS: Secuencias

Ing. Ventre, Luis O.

- Lo ultimo visto la clase pasado:

- Tipos de datos estructurados:

CADENAS

LISTAS

- Ambos son una sucesión de elementos. La cadena de caracteres, la lista de elementos que pueden ser int, float o cadenas.

- Con las listas se puede operar. Se puede acceder a cada elemento con indexacion. etc

- Se pueden borrar elementos de las listas.

TEMARIO

- 5.2 Listas Cont.
  - 5.2.9 Pertenencia de un elemento a una lista
  - 5.2.10 Ordenación de una lista
- 5.3 De cadenas a listas y viceversa.
- 5.4 Matrices
  - 5.4.1 Sobre la creación de matrices
  - 5.4.2 Lectura de matrices
  - 5.4.3 Que dimension tiene una matriz?
  - 5.4.4 Operaciones con matrices

- 5.2 Listas Cont.
  - 5.2.9 Pertenencia de un elemento a una lista
  - 5.2.10 Ordenación de una lista
- 5.3 De cadenas a listas y viceversa.
- 5.4 Matrices
  - 5.4.1 Sobre la creación de matrices
  - 5.4.2 Lectura de matrices
  - 5.4.3 Que dimension tiene una matriz?
  - 5.4.4 Operaciones con matrices

Pertenencia de un elemento a una lista

- Nuestro objetivo es determinar si un elemento pertenece o no a una lista.
- Veamos un algoritmo para hacerlo
- Que deberíamos hacer?

1) Almacenar el elemento a buscar.
2) Recorrer la lista preguntando si el elemento esta o no.
3) Hacer algo para finalizar si esta. Imprimir resultado

Pertenencia de un elemento a una lista

- Como quedaría el código:

pertenencia_5.py

pertenencia.py

```
1  elemento = 5
2  lista = [1, 4, 5, 1, 3, 8]
3
4  pertenece = False
5  for i in lista:
6      if elemento == i:
7          pertenece = True
8          break
9
10 if pertenece:
11     print 'Pertenece'
12 else:
13     print 'No pertenece'
```

Pertenencia de un elemento a una lista

Ing. Ventre, Luis O.

- Esta operación es muy usual en python. Por lo tanto existe un operador predefinido llamado **IN.**

- **Este operador recibe un elemento por el lado izquierdo y una lista por el derecho. Y devuelve True o False.**

elemento IN lista

pertenencia_8.py

pertenencia.py

```
1 conjunto = [1, 2, 3]
2 elemento = int(raw_input('Dame un número: '))
3 if elemento not in conjunto:
4     conjunto.append(elemento)
```

Pertenencia de un elemento a una lista

- Funcionara el operador IN en cadenas?...pruébalo?...
- **Haz un programa y pregunta si ‘l’ in ‘Hola’…**
- **Luego si ‘ol’ in ‘Hola’….**

```
>>> 'a' in 'cadena'
True
>>> 'ade' in 'cadena'
True
>>> 'ada' in 'cadena'
False
```

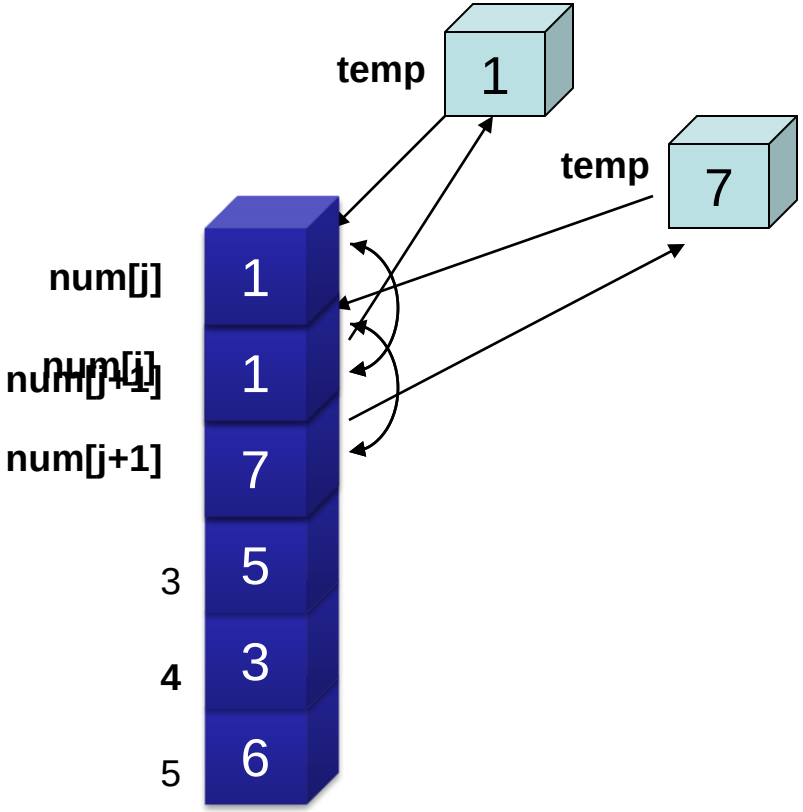
- 5.2 Listas Cont.
  - 5.2.9 Pertenencia de un elemento a una lista
  - 5.2.10 Ordenación de una lista
- 5.3 De cadenas a listas y viceversa.
- 5.4 Matrices
  - 5.4.1 Sobre la creación de matrices
  - 5.4.2 Lectura de matrices
  - 5.4.3 Que dimensión tiene una matriz?
  - 5.4.4 Operaciones con matrices

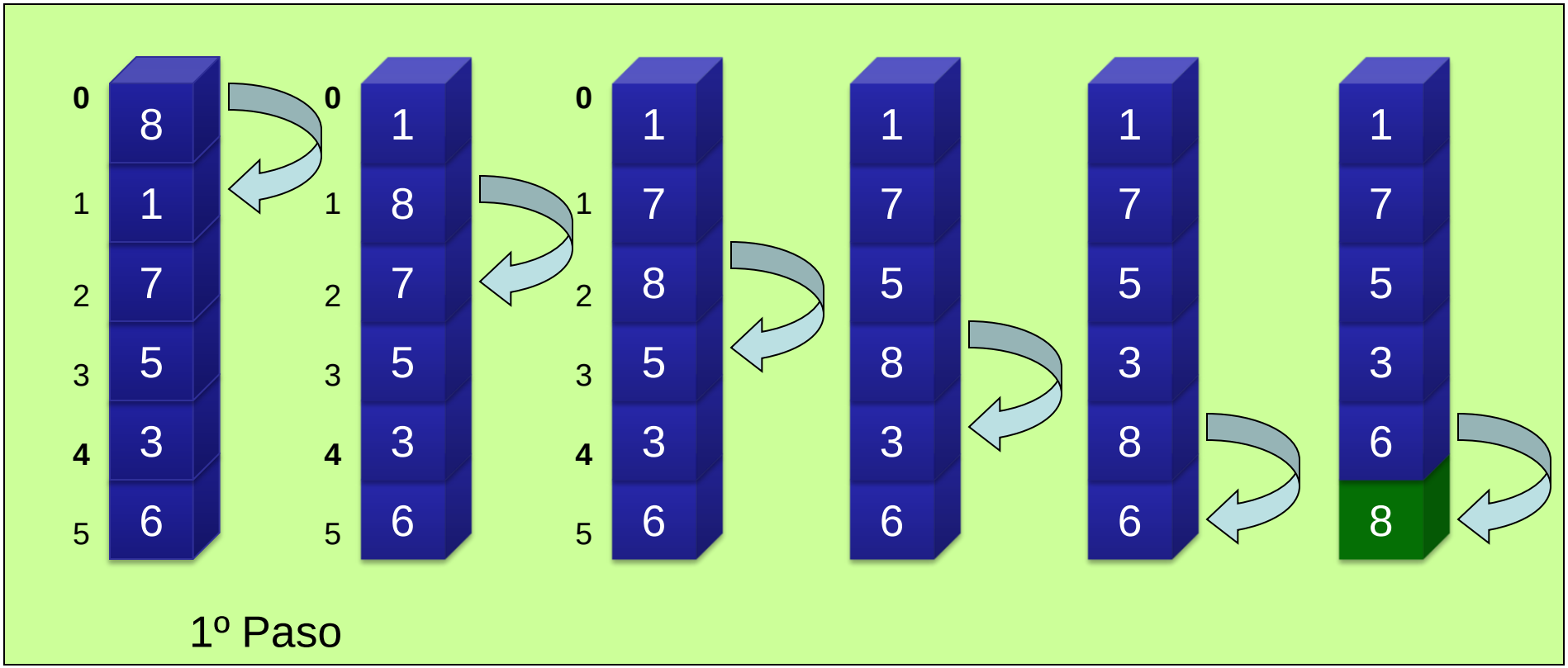
- Una historia que se repite!...**ORDENAR** una lista de menor a mayor.
- Ordenar es muy útil, en infinidad de aplicaciones.
- Estudiaremos el METODO DE LA BURBUJA. El cual es muy sencillo pero no muy eficiente.
- El método consiste en comenzar comparando los DOS primeros elementos de una lista, y luego continuar de la misma forma con los demás!....veamos un ejemplo:

- Supongamos una lista:

- Para lograrlo, el algortimo inicia comparando los primeros dos elementos de la lista. El elemento [0] y [1]. Osea 8 y 1!...

Veamos paso a paso el algoritmo:

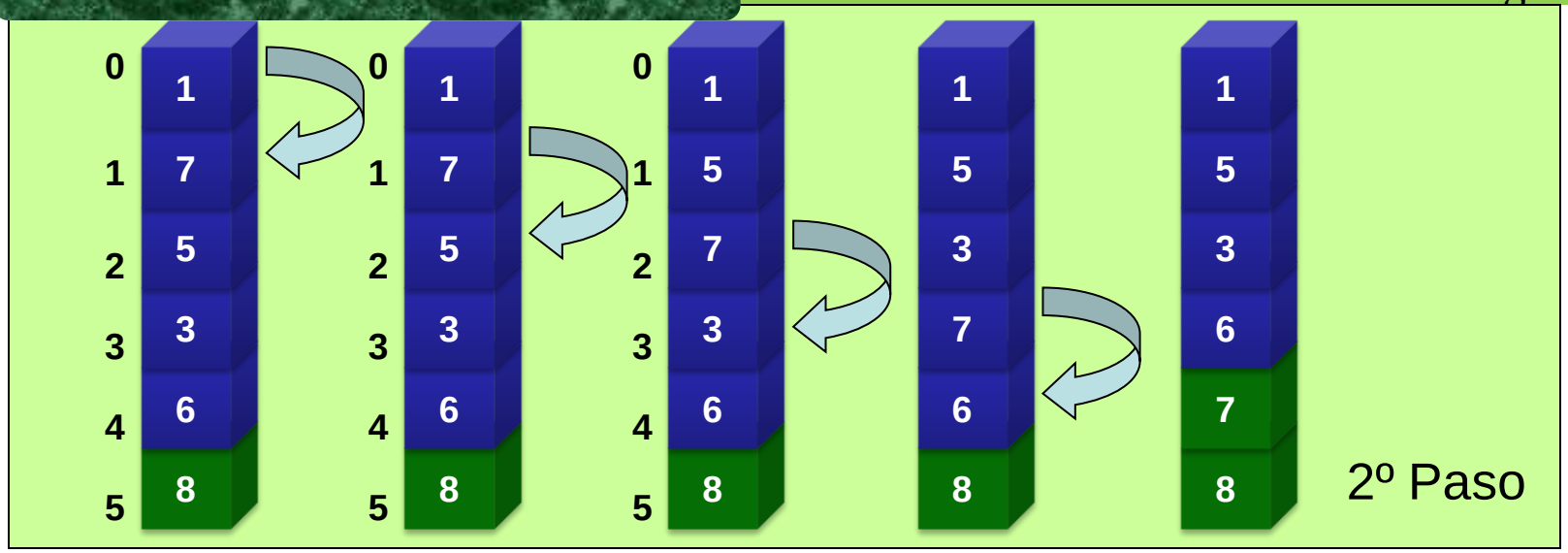


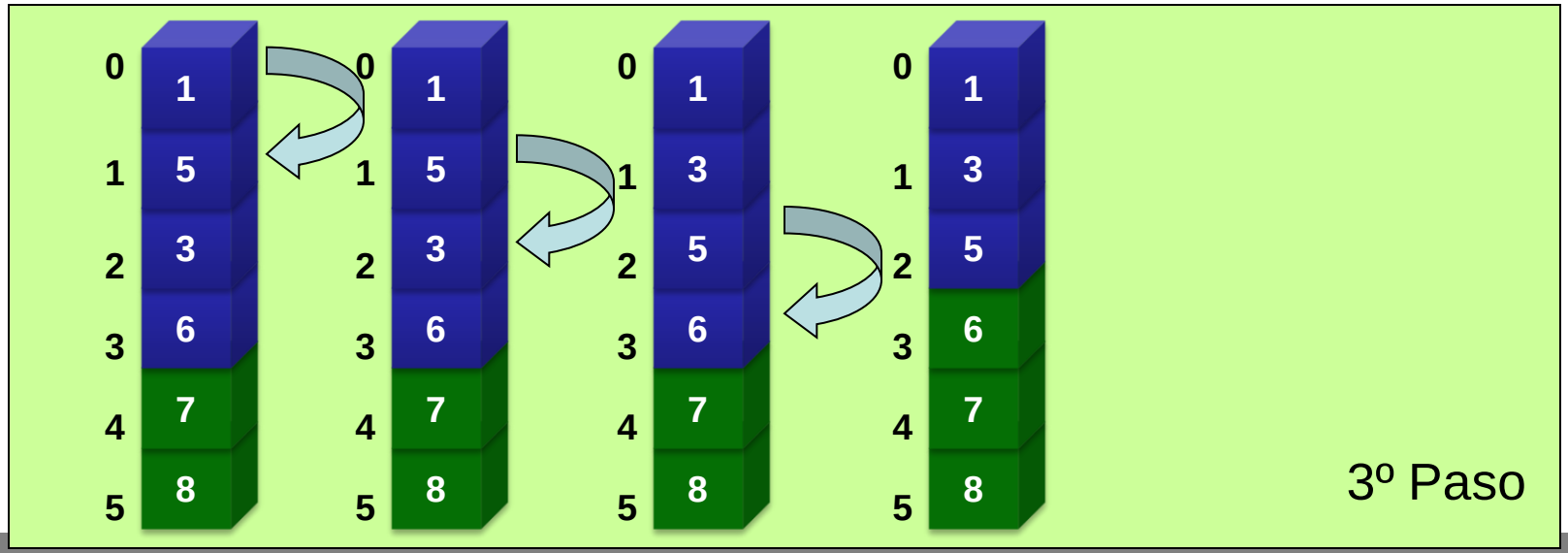

0 1 2 3 4 5
8 1 7 5 3 6
0 1 2 3 4 5
1 8 7 5 3 6
0 1 2 3 4 5
1 7 8 5 3 6
0 1 2 3 4 5
1 7 5 8 3 6
0 1 2 3 4 5
1 7 5 3 8 6
0 1 2 3 4 5
1 7 5 3 6 8

1º Paso

Ordenación de una lista

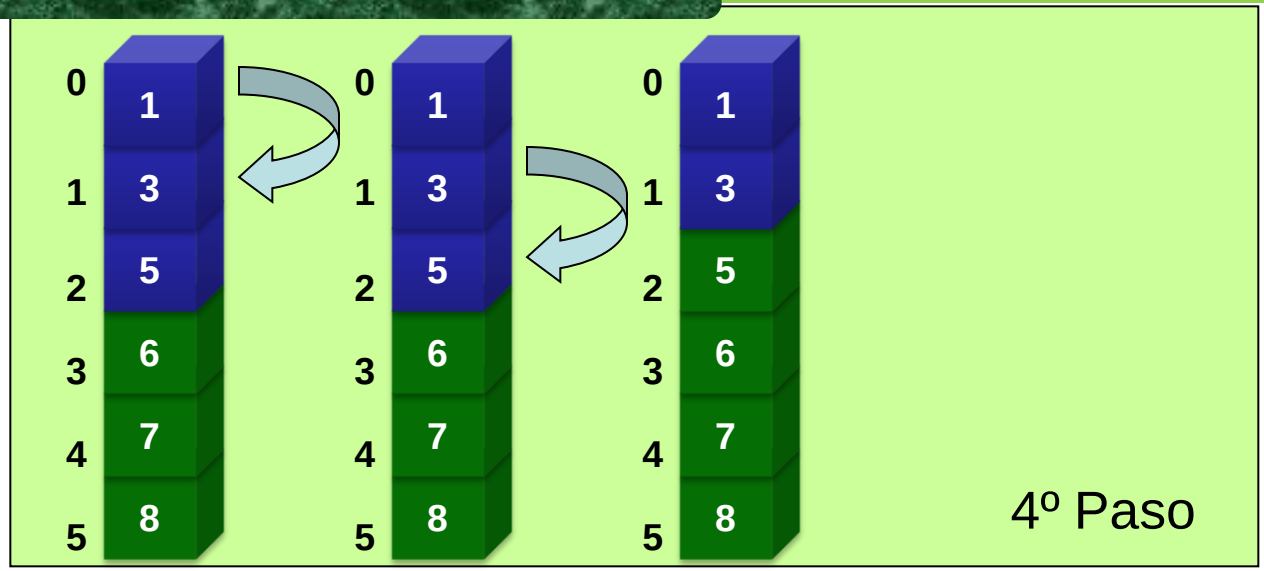
Ing. Ventre, Luis O.

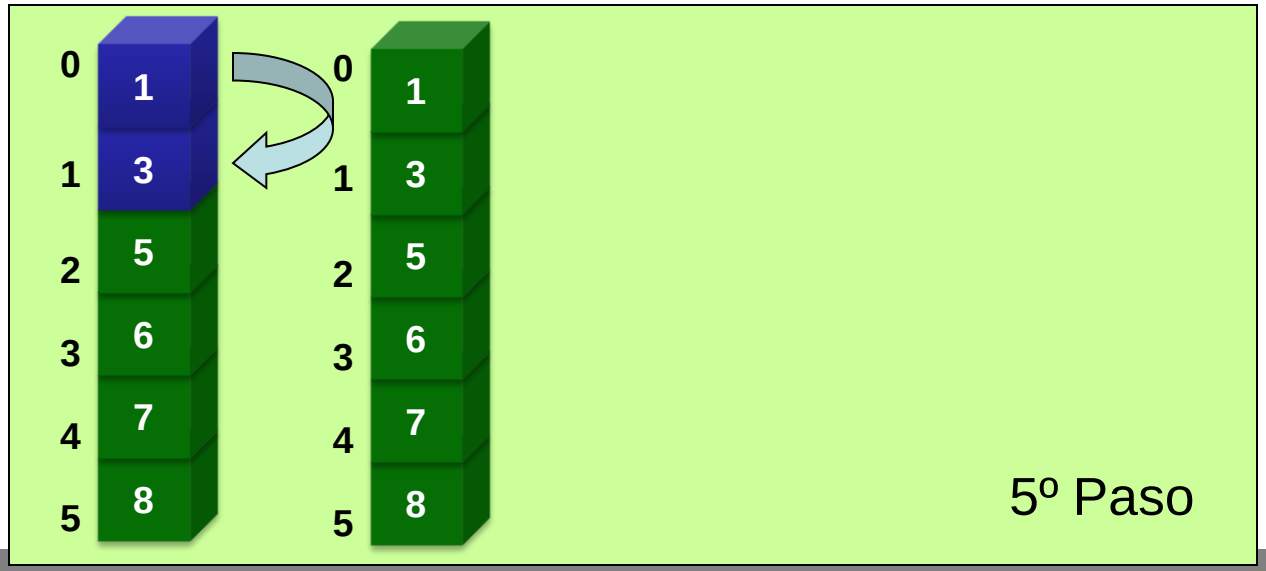




Ordenación de una lista

Ing. Ventre, Luis O.





- Repasemos un poco. Una lista de N elementos necesita N-1 pasadas.
- En cada pasada, conseguimos ordenar al menos un elemento: el mayor.
- Es por esto su nombre, el mayor valor se hunde, y los menores como burbujas van subiendo.
- Intentamos codificarlo!...

- Repasemos un poco. Una lista de N elementos necesita N-1 pasadas.

- Deberíamos iniciar con un bucle, que hagas las n-1 pasadas.

```
1 lista = [2, 26, 4, 3, 1]
2
3 for i in range(1, len(lista)): # Bucle que hace len(lista)-1 pasadas.
4     hacer una pasada
5
6 print lista
```

- En cada pasada, debemos comparar las celdas contiguas de la lista.

```
1 lista = [2, 26, 4, 3, 1]
2
3 for i in range(1, len(lista)):
4     for j in range(0, len(lista)-i):
5         comparar lista[j] y lista[j+1] y, si procede, intercambiarlos
6
7 print lista
```

- Ahora deberíamos implementar la comparación y el intercambio.

```
1  lista = [2, 26, 4, 3, 1]
2
3  for i in range(1, len(lista)):
4      for j in range(0, len(lista)-i):
5          if lista[j] > lista[j+1]:
6              elemento = lista[j]
7              lista[j] = lista[j+1]
8              lista[j+1] = elemento
9
10 print lista
```

- Observe el siguiente programa y su ejecución!...la inclusión de frases de impresión durante el programa son de gran ayuda.

```
1  lista = [2, 26, 4, 3, 1]
2
3  for i in range(1, len(lista)):
4      print 'Pasada', i
5      for j in range(0, len(lista)-i):
6          print '  Comparación de los elementos en posición %d y %d' % (j, j+1)
7          if lista[j] > lista[j+1]:
8              elemento = lista[j]
9              lista[j] = lista[j+1]
10             lista[j+1] = elemento
11             print '  Se intercambian'
12         print '  Estado actual de la lista', lista
13
14 print lista
```

- Ejecución!...

```
Pasada 1
  Comparación de los elementos en posición 0 y 1
  Estado actual de la lista [2, 26, 4, 3, 1]
  Comparación de los elementos en posición 1 y 2
  Se intercambian
  Estado actual de la lista [2, 4, 26, 3, 1]
  Comparación de los elementos en posición 2 y 3
  Se intercambian
  Estado actual de la lista [2, 4, 3, 26, 1]
  Comparación de los elementos en posición 3 y 4
  Se intercambian
  Estado actual de la lista [2, 4, 3, 1, 26]
Pasada 2
  Comparación de los elementos en posición 0 y 1
  Estado actual de la lista [2, 4, 3, 1, 26]
  Comparación de los elementos en posición 1 y 2
  Se intercambian
  Estado actual de la lista [2, 3, 4, 1, 26]
  Comparación de los elementos en posición 2 y 3
```

- 5.2 Listas Cont.
  - 5.2.9 Pertenencia de un elemento a una lista
  - 5.2.10 Ordenación de una lista
- 5.3 De cadenas a listas y viceversa.
- 5.4 Matrices
  - 5.4.1 Sobre la creación de matrices
  - 5.4.2 Lectura de matrices
  - 5.4.3 Que dimensión tiene una matriz?
  - 5.4.4 Operaciones con matrices

- **Es usual, convertir de cadenas a listas y viceversa**.

- El lenguaje propone un par de métodos para hacerlo directamente y ahorrarnos mucho tiempo de programación!.

- Para convertir una CADENA en una LISTA usamos **SPLIT**

```
>>>‘una cadena corta’.split()
```

```
>>>[‘una’, ‘cadena’, ‘corta’]
```

- Veamos un ejemplo de SPLIT, con muchos espacios en blanco funciona?

```
>>>‘uno      dos  tres’.split()
```

```
>>>[‘uno’, ‘dos’, ‘tres’]
```

Perfecto….éste método acepta un argumento opcional: el **CARACTER DIVISOR** de los elementos.
Por defecto es el ESPACIO en blanco.

De cadenas a listas y viceversa

- Existe un método para hacer lo contrario? Si y se denomina **JOIN! Este une todas las cadenas de una lista formando UNA SOLA.**

- **Para utilizarlo debemos colocar una cadena a izquierda del «.» la cual será el «nexo» entre los elementos de la lista en la futura cadena.**

```
>>>‘ ’.join([‘uno’,’dos’,’tres’])
```

```
>>>‘uno dos tres’
```

De cadenas a listas y viceversa

- Aplicaciones:
- **Podemos contar PALABRAS fácilmente con el método SPLIT y LEN! (sin importar espacios!)**

```
>>>len(‘    uno dos   tres ’ .split())
 3
```

- **El método LIST, devuelve una LISTA con todos los caracteres de una cadena.**

```
>>>list(‘cadena’)
[‘c’, ‘a’, ‘d’, ‘e’, ‘n’, ‘a’]
```

De cadenas a listas y viceversa

- Ejemplos:

```
>>> list('uno dos tres')
['u', 'n', 'o', ' ', 'd', 'o', 's', ' ', 't', 'r', 'e', 's']
>>>
>>>
>>> 'uno dos tres'.split()
['uno', 'dos', 'tres']
>>>
>>>
>>>
>>> ' '.join(['uno', 'dos', 'tres'])
'uno dos tres'
>>>
```

- 5.2 Listas Cont.
  - 5.2.9 Pertenencia de un elemento a una lista
  - 5.2.10 Ordenación de una lista
- 5.3 De cadenas a listas y viceversa.
- 5.4 Matrices
  - 5.4.1 Sobre la creación de matrices
  - 5.4.2 Lectura de matrices
  - 5.4.3 Que dimensión tiene una matriz?
  - 5.4.4 Operaciones con matrices

Matrices

- **Def: Son disposiciones bidimensionales de valores.**
- **Ej.**

$$M = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 12 & 6 \\ 1 & 0 & -3 \\ 4 & 5 & 8 \end{bmatrix}$$

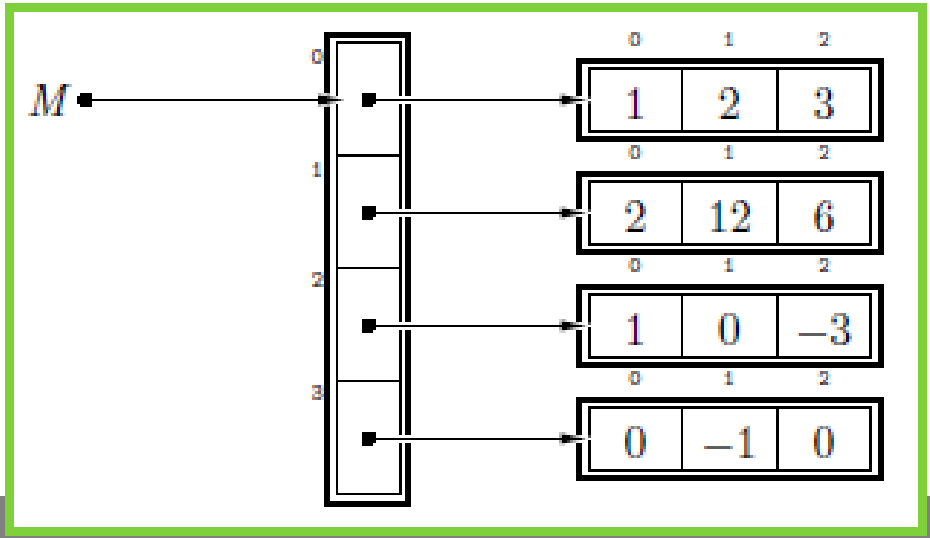
- **Esta matriz tiene 4 filas y 3 columnas.**
- **Esto se indica como Matriz de DIMENSION 4 x 3.**

Matrices

- Las listas permiten representar datos en UNA SOLA DIMENSION.

- Para representar dos dimensiones podemos implementar una LISTA de LISTAS!.

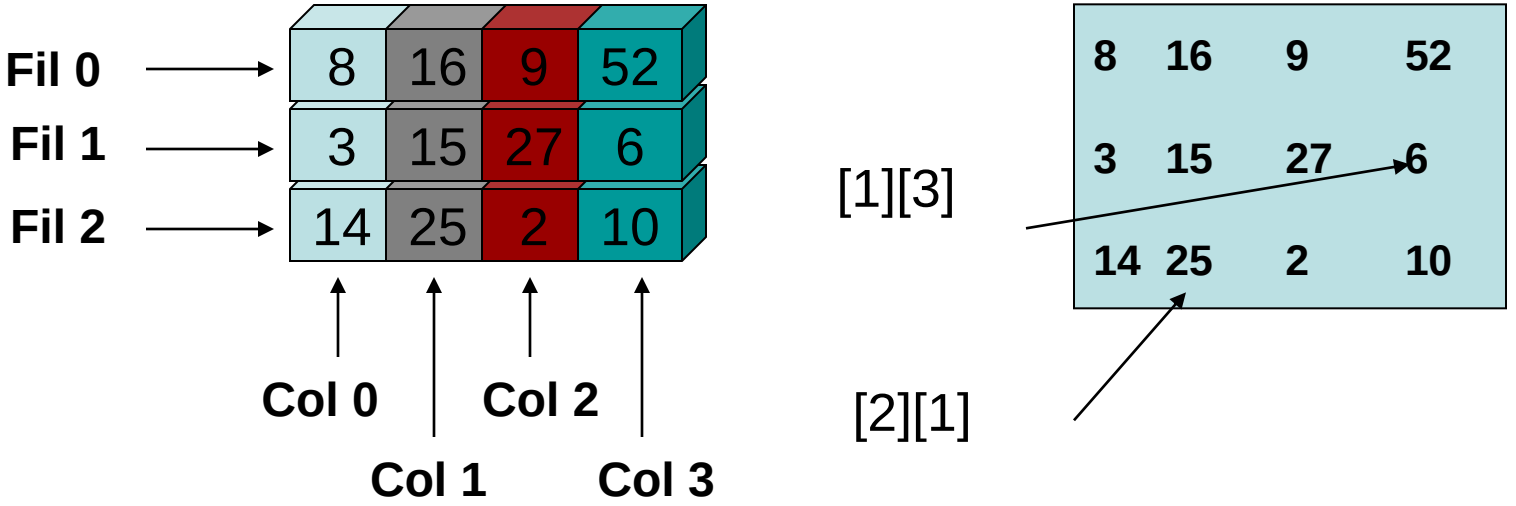
```
>>> M = [ [1, 2, 3], [2, 12, 6], [1, 0, -3], [0, -1, 0] ]
```



Matrices

- Para acceder a un elemento debemos indicar 2 subíndices.
- El primero indica una lista, (la fila) en la que se encuentra.
- El segundo indica un elemento de la lista (la columna) en la que se encuentra.

- Los subíndices empiezan en ?.....0!



- 5.2 Listas Cont.
  - 5.2.9 Pertenencia de un elemento a una lista
  - 5.2.10 Ordenación de una lista
- 5.3 De cadenas a listas y viceversa.
- 5.4 Matrices
  - 5.4.1 Sobre la creación de matrices
  - 5.4.2 Lectura de matrices
  - 5.4.3 Que dimensión tiene una matriz?
  - 5.4.4 Operaciones con matrices

Creación de Matrices

- Debemos crear, una lista de listas.

- Para hacer una matriz nula de 2 x 2, como seria?

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

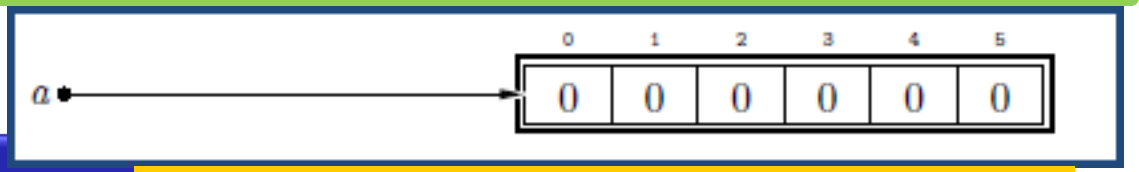
- Muy interesante!...ahora crea una matriz de 3 x 6?
- Podremos hacer uso del operador «*» ?

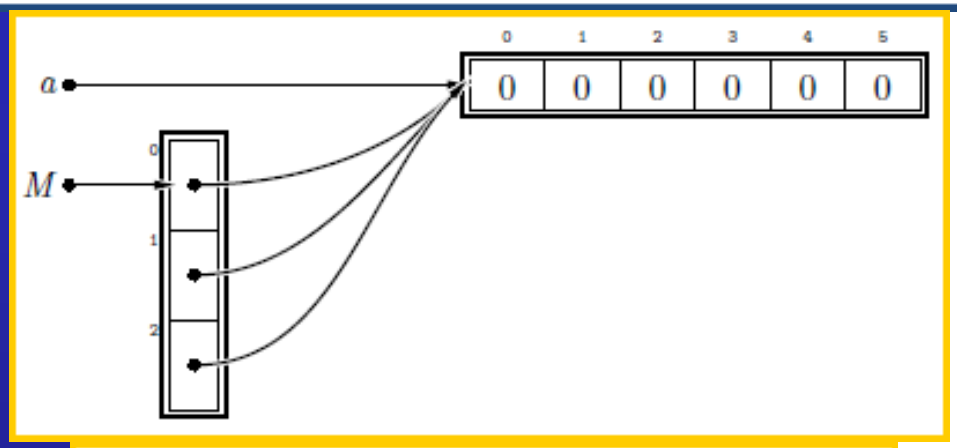
```
>>>a = [0] * 6
>>>a
[ 0, 0, 0, 0, 0, 0]
```

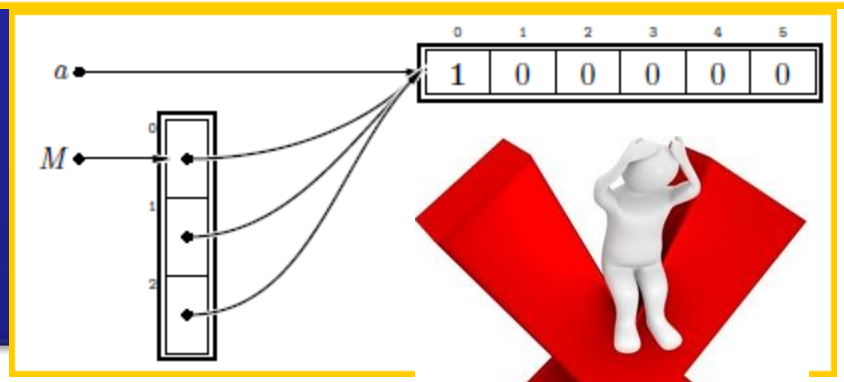
Creación de Matrices

- Todo el proceso seria:

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







Que sucede???.

Creación de Matrices

- Otra forma abreviada y errónea seria:

```
>>>m = [ [0] * 6 ] * 3
```

- Para crear matrices se debe tener mas cuidado!!

- **Cada FILA, debe ser una LISTA diferente de las anteriores!**

- **Hagamos un mejor esfuerzo!...**

Creación de Matrices

- Creando una matriz de 3 filas y 6 columnas:

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

>>>M[0][1]=48
>>>print M
```

Que sucede???.

Creación de Matrices

- Creando una matriz de 3 filas y 6 columnas:

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

>>>M[0][1]=48
>>>print M
```

Que sucede???.

- 5.2 Listas Cont.
  - 5.2.9 Pertenencia de un elemento a una lista
  - 5.2.10 Ordenación de una lista
- 5.3 De cadenas a listas y viceversa.
- 5.4 Matrices
  - 5.4.1 Sobre la creación de matrices
  - 5.4.2 Lectura de matrices
  - 5.4.3 Que dimensión tiene una matriz?
  - 5.4.4 Operaciones con matrices

Lectura de Matrices

- El proceso de creación e ingreso por teclado será:

```
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)))
```

Lectura de Matrices

- Para ingresar una matriz por teclado, debemos crear una matriz nula y recorrerla ingresando los datos:

```
fil=int (raw_input (‘Ingrese cantidad de Filas’))
col=int (raw_input (‘y ahora de Columnas’))
…

>>>for i in range (fil):
…       for j in range (col):
…           M[i][j]=float(raw_input(‘Dame el valor (%d,%d)’ %(i,j)))
```

- 5.2 Listas Cont.
  - 5.2.9 Pertenencia de un elemento a una lista
  - 5.2.10 Ordenación de una lista
- 5.3 De cadenas a listas y viceversa.
- 5.4 Matrices
  - 5.4.1 Sobre la creación de matrices
  - 5.4.2 Lectura de matrices
  - 5.4.3 Que dimensión tiene una matriz?
  - 5.4.4 Operaciones con matrices

Dimensión de una matriz

- En listas usábamos la función LEN, y ahora?

```
>>> a= [ [1,0], [0,1], [0, 0] ]
>>> len (a)
3
```

Filas

- Esto seria el **numero de filas**!, pero nos faltan las columnas:

```
>>> a= [ [1,0], [0,1], [0, 0] ]
>>> len (a[0])
2
```

Columnas

- 5.2 Listas Cont.
  - 5.2.9 Pertenencia de un elemento a una lista
  - 5.2.10 Ordenación de una lista
- 5.3 De cadenas a listas y viceversa.
- 5.4 Matrices
  - 5.4.1 Sobre la creación de matrices
  - 5.4.2 Lectura de matrices
  - 5.4.3 Que dimensión tiene una matriz?
  - 5.4.4 Operaciones con matrices

Operaciones con Matrices

- Comencemos por SUMAR matrices.
- Recuerda que solo es posible sumar dos matrices que tengan la misma DIMENSION!
- Debes pedir primero las dimensiones de la matriz.
- Luego solicitar el ingreso de una
- Luego solicitar el ingreso de la segunda.
- Finalmente generar una tercer matriz con la suma de las anteriores. Recordar: suma[0][0] = a[0][0] + b[0][0]

Operaciones con Matrices

- Creamos 3 matrices

```
6  A = []
7  for i in range(m):
8      A.append( [0] * n )
```

- Ingresamos los valores por teclado a las matrices a sumar:

```
14 # ... y leemos sus contenidos de teclado.
15 print 'Lectura de la matriz A'
16 for i in range(m):
17     for j in range(n):
18         A[i][j] = float(raw_input('Dame el componente (%d,%d): ' % (i, j)))
```

Operaciones con Matrices

- Hacemos la suma:

```
30 # Empieza el cálculo de la suma.
31 for i in range(m):
32     for j in range(n):
33         C[i][j] = A[i][j] + B[i][j]
```

- Mostramos los resultados….como imprimiría la matriz?

```
37 for i in range(m):
38     for j in range(n):
39         print C[i][j],
40     print
```

Que hace este print?

Multiplicación de matrices

- Multiplicar matrices no es tan simple. Veamos matemáticamente:
- Solo podemos multiplicar matrices de las siguientes dimensiones:
- Si la matriz A, es de P X Q elementos.
- La matriz B debe ser de Q X R.
- La matriz resultante será de: P X R

Multiplicación de matrices

- Por lo tanto las siguientes matrices son de:

- A = 2 x 3 (pxq)
- B = 3 x 2 (qxr)
- R = 2 x 2 (pxr)

B

5	2
1	8
3	9

A

2	3	4
8	3	5

R

25	64
58	85

Multiplicación de matrices

- Debemos crear las matrices de las dimensiones correctas.
- Ingresar los datos a cada matriz.
- Calcular el producto.
- Podemos observar que el resultado de cada elemento es equivalente a una sumatoria, de los productos de la primer fila de la matriz A por cada columna de la matriz B.
- Mostrar resultado

$$C_{i,j} = \sum_{k=1}^{q} A_{i,k} \cdot B_{k,j},$$

Multiplicación de matrices

- Veamos el código del calculo del producto:

```
31 # Y efectuamos el cálculo del producto.
32 for i in range(p):
33     for j in range(r):
34         for k in range(q):
35             C[i][j] += A[i][k] * B[k][j]
```

- Las líneas 34 y 35 no son mas que un sumatorio.

Repasando!...

- **Pertenencia de Elementos a una lista! Operador IN.**
- **Ordenación de una lista. Método de la burbuja.**
- **De cadenas a listas**
  - **El método SPLIT**
  - **El método LIST**
- **De listas a cadenas**
  - **El método JOIN**

- **Matrices! Lista de Listas.**
- **Como CREAR matrices.**
- **Como leer del teclado MATRICES.**
- **Operaciones con matrices.**
  - **Suma de matrices**
  - **Multiplicación de matrices**
  - **….**