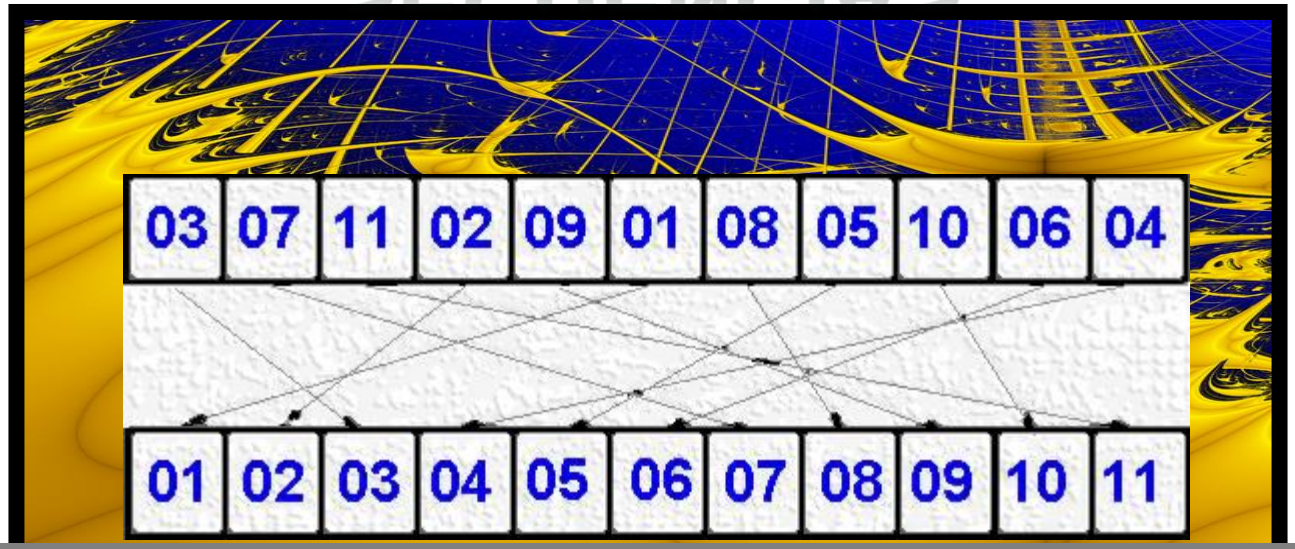
Cátedra de Informática – Dpto. Computación
Universidad Nacional de Córdoba.

Ing. Ventre, Luis O.

CAPITULO 5:

TIPOS ESTRUCTURADOS

SECUENCIAS



- 5.2 Listas
  - 5.2.1 Cosas que ya sabemos
  - 5.2.2 Comparación de listas
  - 5.2.3 El operador IS
  - 5.2.4 Modificación de elementos de Listas
  - 5.2.5 Mutabilidad, inmutabilidad y representación de la info
  - 5.2.6 Adición de elementos a una lista
  - 5.2.7 Lectura de listas por teclado
  - 5.2.8 Borrado de elementos de una lista
  - 5.2.9 Pertenencia de un elemento a una lista
  - 5.2.10 Ordenación de una lista

Listas

- La idea de SECUENCIA de valores es muy POTENTE y UTIL!.
- Python no se limita a SECUENCIAS de caracteres = CADENAS.
- Python PERMITE definir secuencias de valores de cualquier tipo.
- Puede ser una secuencia de números enteros, o flotantes o incluso una secuencia de cadenas, hablamos de **LISTAS**!

Listas

- Python para las LISTAS tiene una notación ESPECIAL.
- Las mismas deben estar encerradas entre corchetes y separar sus elementos con «,»

- Veamos un par de ejemplos:

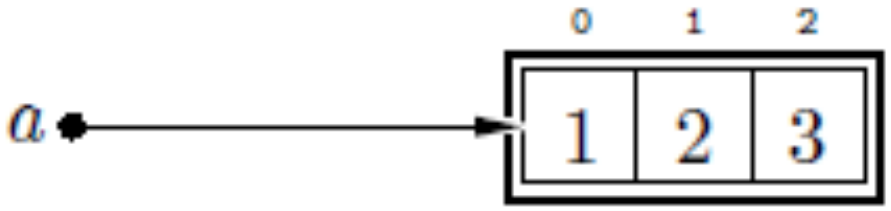
```
>>> a = [1, 2, 3] ↲
>>> a ↲
[1, 2, 3]
```

```
>>> nombres = ['Juan', 'Antonia', 'Luis', 'María'] ↲
```

- También pueden usarse expresiones para cada elemento de la lista!...

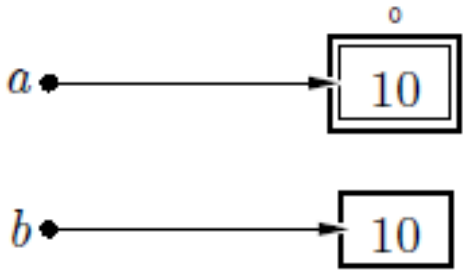
```
>>> a = [1, 1+1, 6/2] ↲
>>> a ↲
[1, 2, 3]
```

- Python almacena las LISTAS, con PUNTEROS igual que las cadenas….por lo tanto:



- Una lista con un UNICO elemento no es lo mismo que una variable.

```
>>> a = [10]
>>> b = 10
```



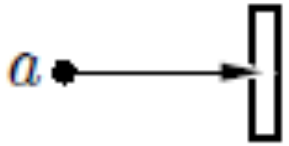
- Al solicitar la impresión se verán diferentes!...

```
>>> print a
[10]
>>> print b
10
```

- Las listas siempre se imprimen entre **CORCHETES**!.

- También existe una LISTA VACIA.

```
>>> a = [ ]
```



- Cosas que YA SABEMOS DE LAS LISTAS!!

  - La función **LEN**, también opera sobre LISTAS.

```
>>> a = [ 1,2,3 ]
>>> len(a)
3
```

- 5.2 Listas
  - 5.2.1 Cosas que ya sabemos
  - 5.2.2 Comparación de listas
  - 5.2.3 El operador IS
  - 5.2.4 Modificación de elementos de Listas
  - 5.2.5 Mutabilidad, inmutabilidad y representación de la info
  - 5.2.6 Adición de elementos a una lista
  - 5.2.7 Lectura de listas por teclado
  - 5.2.8 Borrado de elementos de una lista
  - 5.2.9 Pertenencia de un elemento a una lista
  - 5.2.10 Ordenación de una lista

- El operador + CONCATENA LISTAS.

```
>>> a = [ 1,2,3 ]
>>> a + [4]
[1,2,3,4]
```

- El operador * REPITE LISTAS

```
>>> a = [ 1,2,3 ]
>>> a * 2
[1,2,3,1,2,3]
```

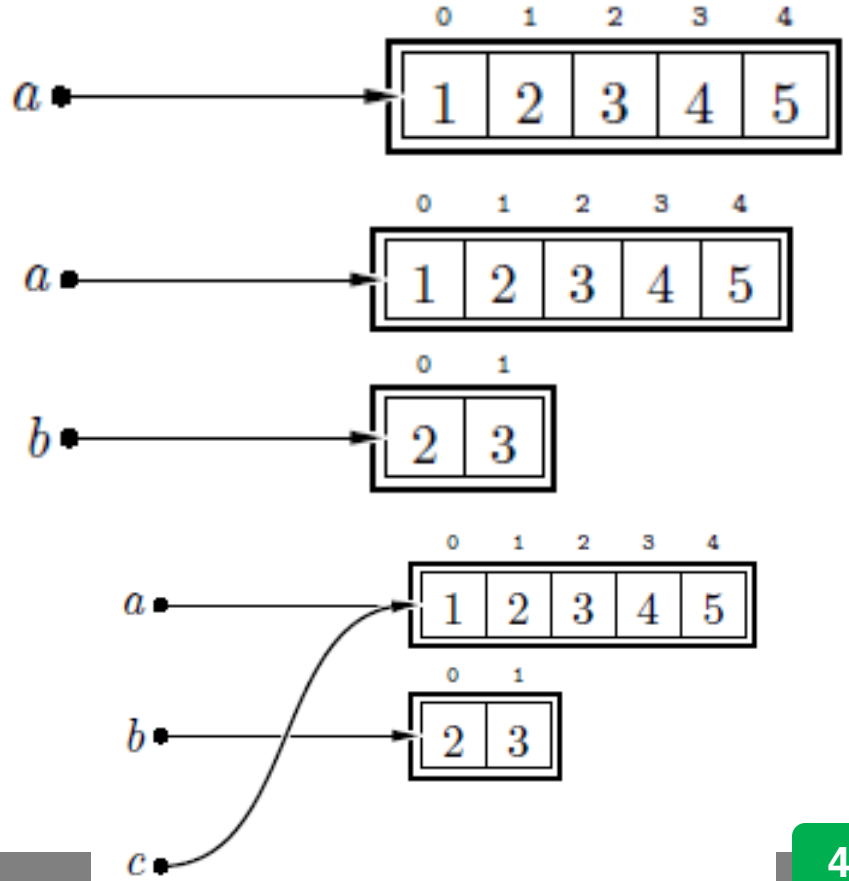
- El operador de INDEXACION también opera en LISTAS.

```
>>> a [0]
1
```

```
>>> [1,2,3] [0]
1
```

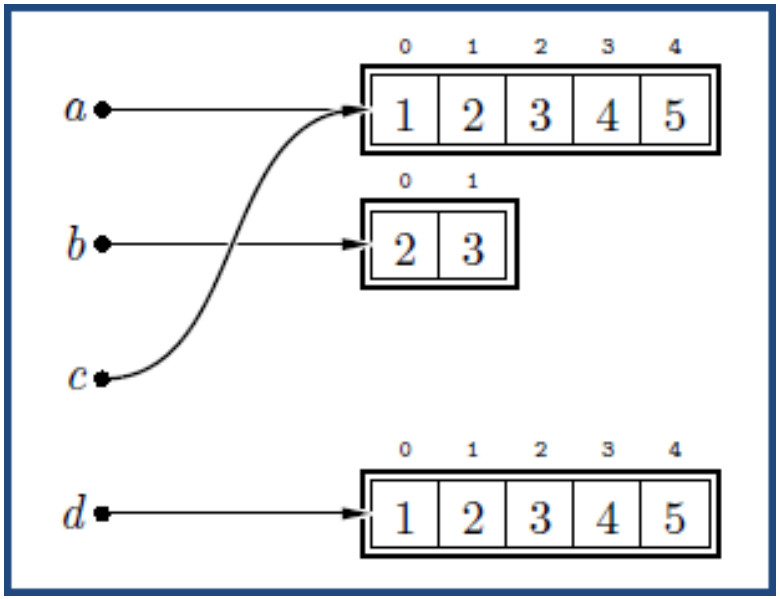
- El operador de CORTE también se aplica en LISTAS. Debes tener en cuenta, que el mismo IMPLICA COPIAR UN FRAGMENTO O TODA LA LISTA.

```
>>> a = [ 1,2,3,4,5 ]
>>> b = a [1:3]
>>>c = a
```



- Una forma de asegurarnos de COPIAR toda la lista es usar el siguiente OPERADOR DE CORTE.

```
>>> a = [ 1,2,3,4,5 ]
>>> d = a [ : ]
```



- También podemos RECORRER una lista

```
>>> a = [ 1,2,3,4,5 ]
>>> for i in a:
>>>       print i
```

- 5.2 Listas
  - 5.2.1 Cosas que ya sabemos
  - 5.2.2 Comparación de listas
  - 5.2.3 El operador IS
  - 5.2.4 Modificación de elementos de Listas
  - 5.2.5 Mutabilidad, inmutabilidad y representación de la info
  - 5.2.6 Adición de elementos a una lista
  - 5.2.7 Lectura de listas por teclado
  - 5.2.8 Borrado de elementos de una lista
  - 5.2.9 Pertenencia de un elemento a una lista
  - 5.2.10 Ordenación de una lista

Comparación de LISTAS

- Los operadores de IGUALDAD – DESIGUALDAD también operan en listas.

```
>>> [1, 2, 3] == [1, 2] ↲
False
>>> [1, 2, 3] == [1, 2, 3] ↲
True
```

- Los operadores < , > , <= , >= también operan con LISTAS. Comparando ELEMENTO A ELEMENTO.

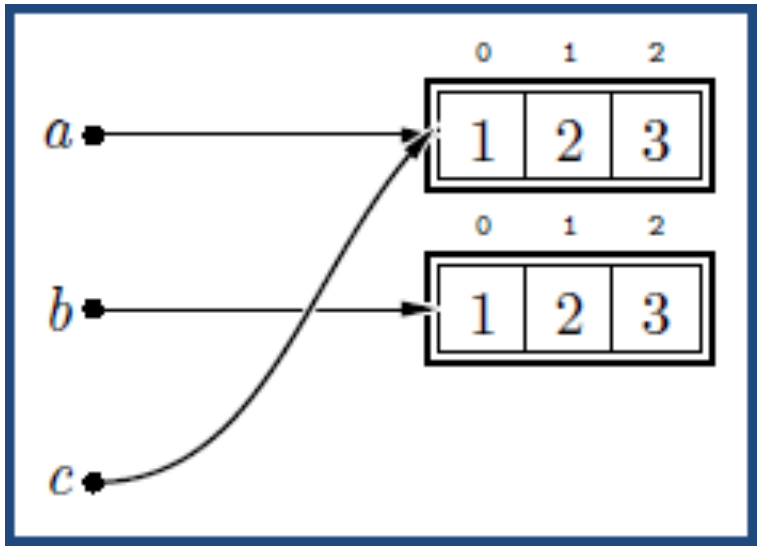
```
>>> [1,2,3] < [1,3,2]
True
```

- 5.2 Listas
  - 5.2.1 Cosas que ya sabemos
  - 5.2.2 Comparación de listas
  - **5.2.3 El operador IS**
  - 5.2.4 Modificación de elementos de Listas
  - 5.2.5 Mutabilidad, inmutabilidad y representación de la info
  - 5.2.6 Adición de elementos a una lista
  - 5.2.7 Lectura de listas por teclado
  - 5.2.8 Borrado de elementos de una lista
  - 5.2.9 Pertenencia de un elemento a una lista
  - 5.2.10 Ordenación de una lista

El operador IS

- Algunas veces, los elementos se duplican, otras solo dos elementos APUNTAN a la misma lista.

- Para diferenciar esto, python posee un operador llamado **IS**, el cual devuelve True, si ambos elementos son en realidad el MISMO elemento; o sea RESIDEN en el mismo lugar de memoria.



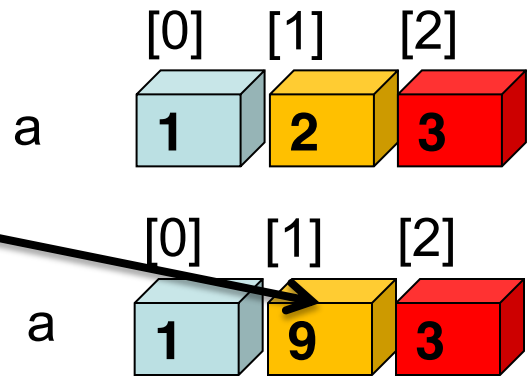

```
>>> a == b
True
>>> a == c
True
>>> a is b
False
>>> a is c
True
```

- 5.2 Listas
  - 5.2.1 Cosas que ya sabemos
  - 5.2.2 Comparación de listas
  - 5.2.3 El operador IS
  - **5.2.4 Modificación de elementos de Listas**
  - 5.2.5 Mutabilidad, inmutabilidad y representación de la info
  - 5.2.6 Adición de elementos a una lista
  - 5.2.7 Lectura de listas por teclado
  - 5.2.8 Borrado de elementos de una lista
  - 5.2.9 Pertenencia de un elemento a una lista
  - 5.2.10 Ordenación de una lista

Modificación de Elementos de Listas

- Hasta el momento solo creamos, y recorrimos LISTAS.
- Es posible, MODIFICAR el valor de UN ELEMENTO de la lista a voluntad.
- Se debe acceder con el operador de INDEXACION al elemento.

```
>>> a = [1,2,3]
>>> a [1] = 9
>>> a
[1, 9 ,3]
```



Modificación de Elementos de Listas

- Analice la siguiente salida por pantalla!

copias_2.py

copias.py

```
1  a = range(0, 5)
2  b = range(0, 5)
3  c = a
4  d = b[:]
5  e = a + b
6  f = b[:1]
7  g = b[0]
8  c[0] = 100
9  d[0] = 200
10 e[0] = 300
11 print a, b, c, d, e, f, g
```

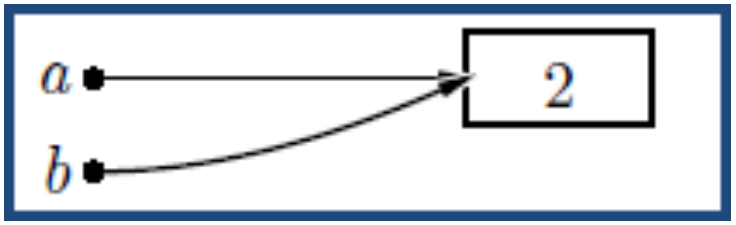
- 5.2 Listas
  - 5.2.1 Cosas que ya sabemos
  - 5.2.2 Comparación de listas
  - 5.2.3 El operador IS
  - 5.2.4 Modificación de elementos de Listas
  - 5.2.5 Mutabilidad, inmutabilidad y representación de la info
  - 5.2.6 Adición de elementos a una lista
  - 5.2.7 Lectura de listas por teclado
  - 5.2.8 Borrado de elementos de una lista
  - 5.2.9 Pertenencia de un elemento a una lista
  - 5.2.10 Ordenación de una lista

Mutabilidad e Inmutabilidad

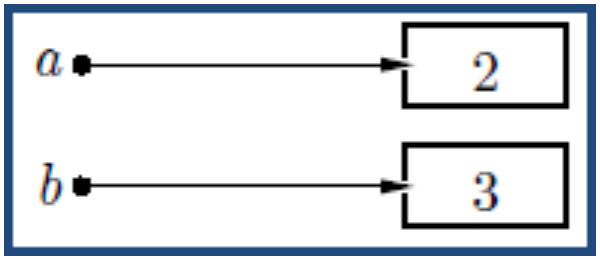
- El objetivo de python es NO consumir mas memoria de lo necesario.

- Por lo tanto, ESCALARES y CADENAS son INMUTABLES. Esto implica que NO pueden modificar su valor. Por lo que python almacena en memoria SOLO una vez cada valor inmutable.

```
>>> a = 1 + 1
>>> b = 2 * 1
```




```
>>> b = b + 1
```

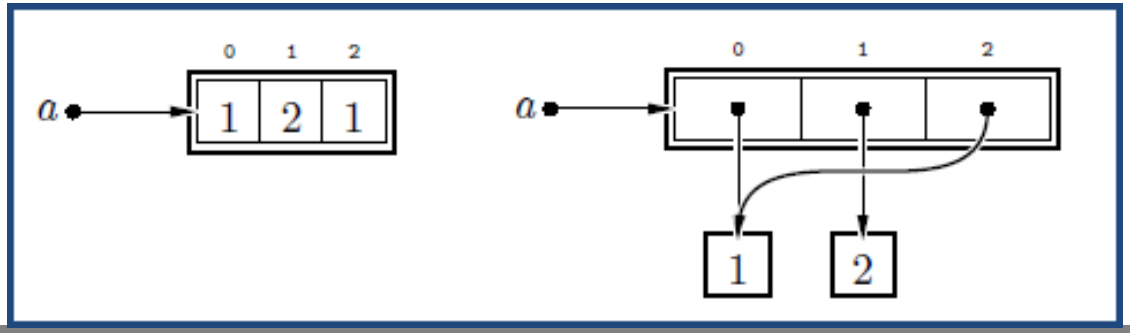


Mutabilidad e Inmutabilidad

- Las cadenas son INMUTABLES no se pueden modificar, python generara un error:

```
>>> a = 'Hola' ↲
>>> a[0] = 'h' ↲
Traceback (innermost last):
  File "<stdin>", line 1, in ?
TypeError: object doesn't support item assignment
```

- Las listas son MUTABLES



- 5.2 Listas
  - 5.2.1 Cosas que ya sabemos
  - 5.2.2 Comparación de listas
  - 5.2.3 El operador IS
  - 5.2.4 Modificación de elementos de Listas
  - 5.2.5 Mutabilidad, inmutabilidad y representación de la info
  - **5.2.6 Adición de elementos a una lista**
  - 5.2.7 Lectura de listas por teclado
  - 5.2.8 Borrado de elementos de una lista
  - 5.2.9 Pertenencia de un elemento a una lista
  - 5.2.10 Ordenación de una lista

Adición de elementos a una Lista

- Es posible hacer crecer una lista.

- Para esto se puede utilizar el operador «+». Pero para ello lo que queremos concatenar debe ser del mismo TIPO de la lista:

```
>>> a = [1, 2, 3] ↲
>>> a = a + 4 ↲
Traceback (innermost last):
  File "<stdin>", line 1, in ?
TypeError: illegal argument type for built-in operation
```

```
>>> a = a + [4] ↲
>>> a ↲
[1, 2, 3, 4]
```

Adición de elementos a una Lista

- Existe otro método correcto y efectivo, usando el operador APPEND.

```
>>> a = [1, 2, 3] ↲
>>> a.append(4) ↲
>>> a ↲
[1, 2, 3, 4]
```

- La diferencia fundamental con la CONCATENACION es que esta, duplica las listas mientras que APPEND **modifica la original.**

- 5.2 Listas
  - 5.2.1 Cosas que ya sabemos
  - 5.2.2 Comparación de listas
  - 5.2.3 El operador IS
  - 5.2.4 Modificación de elementos de Listas
  - 5.2.5 Mutabilidad, inmutabilidad y representación de la info
  - 5.2.6 Adición de elementos a una lista
  - 5.2.7 Lectura de listas por teclado
  - 5.2.8 Borrado de elementos de una lista
  - 5.2.9 Pertenencia de un elemento a una lista
  - 5.2.10 Ordenación de una lista

Lectura de Listas por teclado

- La funcion raw_input funcionará con listas?.....

- Recordemos que raw_input genera una «cadena». Pero a esta cadena la convertíamos en INT o FLOAT? Existe algo similar? SI

- Podemos convertir la cadena ingresada en una LISTA con **LIST**.

```
>>> lista = list(raw_input('Dame una lista: '))
Dame una lista: [1, 2, 3]
>>> lista
['[', '1', ',', ' ', '2', ',', ' ', '3', ']']
```

- Es lo que esperabamos???....

Lectura de Listas por teclado

- La forma de leer una lista por teclado es ELEMENTO A ELEMENTO. Y construirla paso a paso.

```
1 lista = []
2 for i in range(5):
3     elemento = int(raw_input('Dame un elemento:'))
4     lista = lista + [elemento]
```

- O mejor aun, con append!.

```
1 lista = []
2 for i in range(5):
3     elemento = int(raw_input('Dame un elemento:'))
4     lista.append(elemento)
```

Lectura de Listas por teclado

- Otra forma es utilizar el subíndice i

```
1 lista = [0] * 5
2 for i in range(5):
3     elemento[i] = int(raw_input('Dame un elemento:'))
```

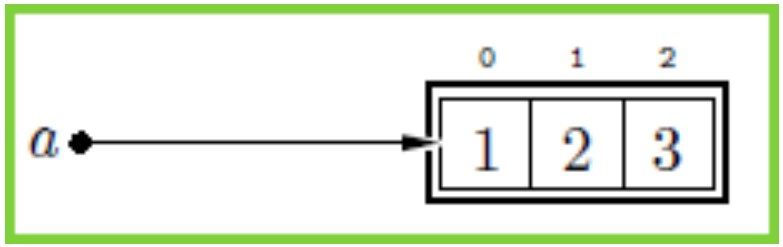
- Y crear una lista de longitud desconocida?

```
1 lista = []
2 numero = int(raw_input('Dame un número: '))
3 while numero >= 0:
4     lista.append(numero)
5     numero = int(raw_input('Dame un número: '))
```

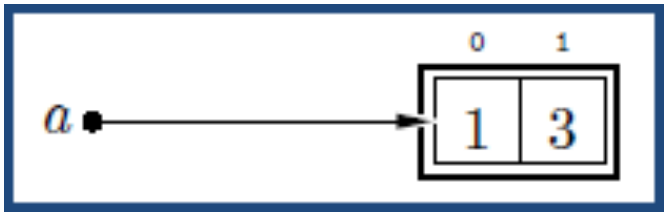
- 5.2 Listas
  - 5.2.1 Cosas que ya sabemos
  - 5.2.2 Comparación de listas
  - 5.2.3 El operador IS
  - 5.2.4 Modificación de elementos de Listas
  - 5.2.5 Mutabilidad, inmutabilidad y representación de la info
  - 5.2.6 Adición de elementos a una lista
  - 5.2.7 Lectura de listas por teclado
  - 5.2.8 Borrado de elementos de una lista
  - 5.2.9 Pertenencia de un elemento a una lista
  - 5.2.10 Ordenación de una lista

Borrado de elementos de una Lista

- Podemos eliminar elementos de una lista, a través del comando **DEL**.



>>>del a[1]



- Se debe indicar el ELEMENTO a borrar.
- La sentencia DEL trabaja sobre la lista original, no duplica.
- Recuerda las cadenas son INMUTABLES, no podemos usar DEL sobre las mismas!.

Borrado de elementos de una Lista

- El borrado de elementos de una lista es PELIGROSO.
- Debes tener especial cuidado con SUBINDICES.
- Al borrar un elemento, CAMBIAN algunos!

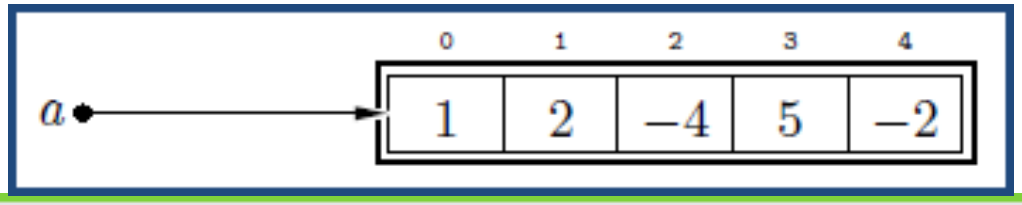
solo_positivos_6.py — solo_positivos.py

```
1 a = [1, 2, -1, -4, 5, -2]
2
3 for i in range(0, len(a)):
4     if a[i] < 0:
5         del a[i]
6
7 print a
```

- Este programa da ERROR, por FUERA DE RANGO!!!

Borrado de elementos de una Lista

- La forma de resolverlo correcta es la siguiente, la entiende?



solo_positivos_8.py

solo_positivos.py

```
1  a = [1, 2, -1, -4, 5, -2]
2
3  i = 0
4  while i < len(a):
5      if a[i] < 0:
6          del a[i]
7      else:
8          i += 1
9
10 print a
```

Repasando!...

- **Tipos de datos ATOMICOS y ESTRUCTURADOS.**
- **Secuencias de CARACTERES = CADENAS.**
- **Operadores sobre CADENAS**
  - **+**
  - *****
  - **%**
  - **int – float – str**
  - **ord – chr…etc**

Lo visto!

- **Secuencias de ESCAPE «\»**
- **La función LEN. Longitud de cadenas.**
- **Indexación de CADENAS…Subíndices positivos y negativos.**
- **Recorrido de CADENAS. FOR in.**
- **Sub-cadenas, el operador de corte [:]**
- **Referencias a cadenas.**

- **Listas!**
- **Operadores y funciones que se COMPARTEN.**
- **Comparación de LISTAS.**
- **Modificación de un elemento de una LISTA.**
- **Mutabilidad – Inmutabilidad.**
- **Adición de elementos a una lista = + o append.**

- **Lectura de listas por teclado.**
- **Borrado de un elemento de una lista!...**
- **Recorrer listas borrando elementos!.**