😎 Listas

Las listas son una estructura de datos muy flexible. Python permite manipular listas de muchas maneras. Las listas son conjuntos ordenados de elementos (números, cadenas, listas, etc). Las listas se delimitan por corchetes ([ ]) y los elementos se separan por comas.

Las listas pueden contener elementos del mismo tipo:

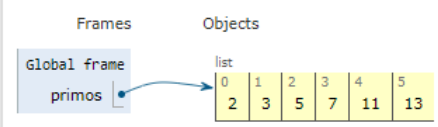
In [1]:
```
primos = [2, 3, 5, 7, 11, 13]
primos
```

Out[1]:
```
[2, 3, 5, 7, 11, 13]
```

```
1 primos = [2, 3, 5, 7, 11, 13]
2 print(primos)
```

Print output (drag lower right corner to resize)

```
[2, 3, 5, 7, 11, 13]
```



In [2]:
```
diasLaborables = ["Lunes", "Martes", "Miércoles", "Jueves", "Viernes"]
diasLaborables
```

Out[2]:
```
['Lunes', 'Martes', 'Miércoles', 'Jueves', 'Viernes']
```

O pueden contener elementos de tipos distintos:

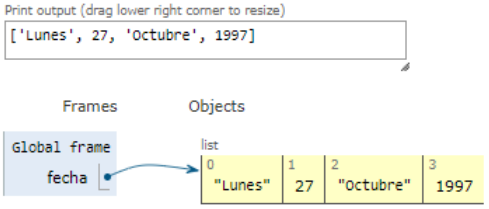
In [3]:
```
fecha = ["Lunes", 27, "Octubre", 1997]
fecha
```

Out[3]:
```
['Lunes', 27, 'Octubre', 1997]
```

```
1 fecha = ["Lunes", 27, "Octubre", 1997]
2 print(fecha)
```

Print output (drag lower right corner to resize)

```
['Lunes', 27, 'Octubre', 1997]
```



O pueden contener listas:

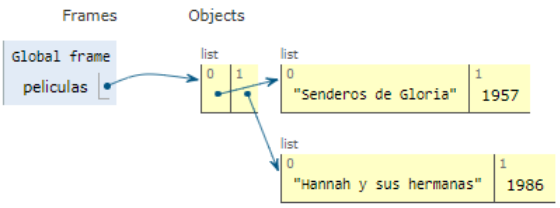
In [4]:
```
peliculas = [ ["Senderos de Gloria", 1957], ["Hannah y sus hermanas", 1986]]
peliculas
```

Out[4]:
```
[['Senderos de Gloria', 1957], ['Hannah y sus hermanas', 1986]]
```

```
1 peliculas = [["Senderos de Gloria", 1957],["Hannah y sus
2 print(peliculas)
```

Print output (drag lower right corner to resize)

```
[['Senderos de Gloria', 1957], ['Hannah y sus hermanas', 1986]]
```



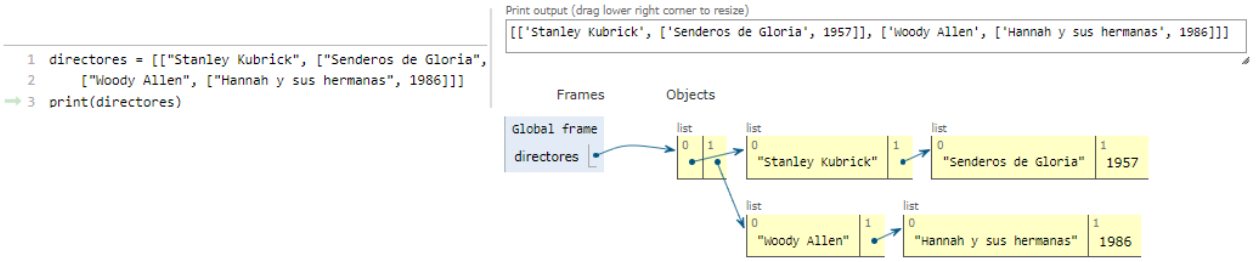
In [5]:
```
type(peliculas)
```

Out[5]:
```
list
```

Las listas pueden tener muchos niveles de anidamiento:

In [6]:
```
directores = [["Stanley Kubrick", ["Senderos de Gloria", 1957]], ["Woody Allen", ["Hannah y sus hermanas", 1986]]]
directores
```

Out[6]:
```
[['Stanley Kubrick', ['Senderos de Gloria', 1957]],
 ['Woody Allen', ['Hannah y sus hermanas', 1986]]]
```



Las variables de tipo lista hacen referencia a la lista completa:

In [7]:
```
lista = [1, "a", 45]
lista
```

Out[7]:
```
[1, 'a', 45]
```

Una lista que no contiene ningún elemento se denomina lista vacía:

In [8]:
```
lista = []
lista
```

Out[8]:
```
[]
```

Al definir una lista se puede hacer referencia a otras variables

In [9]:
```
nombre = "Pepe"
edad = 25
lista = [nombre, edad]
lista
```

Out[9]:
```
['Pepe', 25]
```

In [10]:
```
nombre = "Pepito"
nombre
```

Out[10]:
```
'Pepito'
```

Hay que tener cuidado al modificar una variable que se ha utilizado para definir otras variables, porque esto puede afectar al resto

Si el contenido se trata de objetos inmutables, no resulta afectado, como muestra el siguiente ejemplo:

In [11]:
```
nombre = "Pepe"
edad = 25
lista = [nombre, edad]
lista
```

Out[11]:
```
['Pepe', 25]
```

In [12]:
```
nombre = "Juan"
lista
```

Out[12]:
```
['Pepe', 25]
```

Pero si se trata de objetos mutables al modificar la variable se modifica el objeto, como muestra el siguiente ejemplo:

In [13]:
```
nombres = ["Ana", "Bernardo"]
edades = [22, 21]
lista = [nombres, edades]
lista
```

Out[13]:
```
[['Ana', 'Bernardo'], [22, 21]]
```

In [14]:
```
nombres += ["Cristina"]
nombres
```

Out[14]:
```
['Ana', 'Bernardo', 'Cristina']
```

In [15]:
```
lista
```

Out[15]:
```
[['Ana', 'Bernardo', 'Cristina'], [22, 21]]
```

Una lista puede contener listas (que a su vez pueden contener listas, que a su vez etc...):

In [16]:
```
persona1 = ["Ana", 25]
persona2 = ["Benito", 23]
lista = [persona1, persona2]
lista
```

Out[16]:
```
[['Ana', 25], ['Benito', 23]]
```

Se puede acceder a cualquier elemento de una lista escribiendo el nombre de la lista y entre corchetes el número de orden en la lista. El primer elemento de la lista es el número 0

In [17]:
```
lista = [10, 20, 30, 40]
lista[2], lista[0]
```

Out[17]:
```
(30, 10)
```

In [18]:
```
lista[4]
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
Input In [18], in <cell line: 1>()
----> 1 lista[4]

IndexError: list index out of range
```

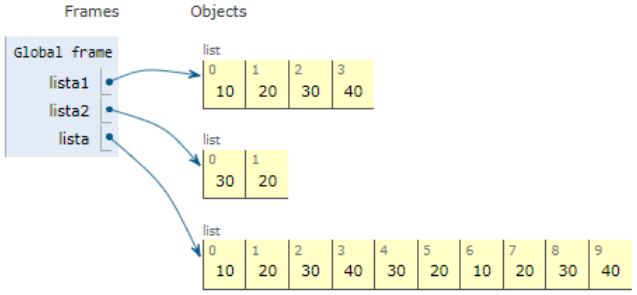
Concatenar listas

In [19]:
```
lista1 = [10, 20, 30, 40]
lista2 = [30, 20]
lista = lista1 + lista2 + lista1
lista
```

Out[19]:
```
[10, 20, 30, 40, 30, 20, 10, 20, 30, 40]
```

```
1 lista1 = [10, 20, 30, 40]
2 lista2 = [30, 20]
3 lista = lista1 + lista2 + lista1
4 print(lista)
```

Print output (drag lower right corner to resize)
```
[10, 20, 30, 40, 30, 20, 10, 20, 30, 40]
```

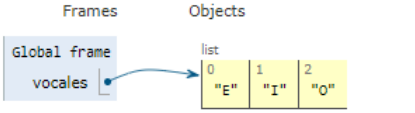


In [20]:
```
vocales = ["E", "I", "O"]
vocales
```

Out[20]:
```
['E', 'I', 'O']
```

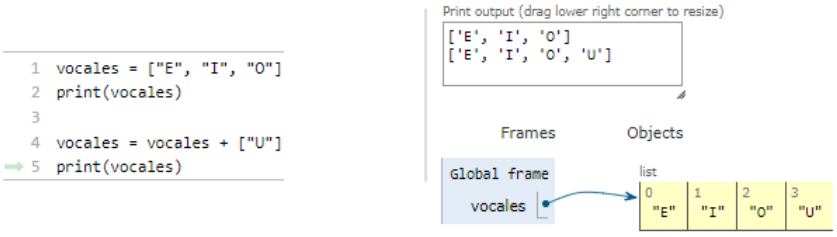
```
1 vocales = ["E", "I", "O"]
2 print(vocales)
3
4 vocales = vocales + ["U"]
5 print(vocales)
```

Print output (drag lower right corner to resize)
```
['E', 'I', 'O']
```



In [21]:
```
vocales = vocales + ["U"]
vocales
```

Out[21]:
```
['E', 'I', 'O', 'U']
```



El operador suma (+) necesita que los 2 operandos sean listas:

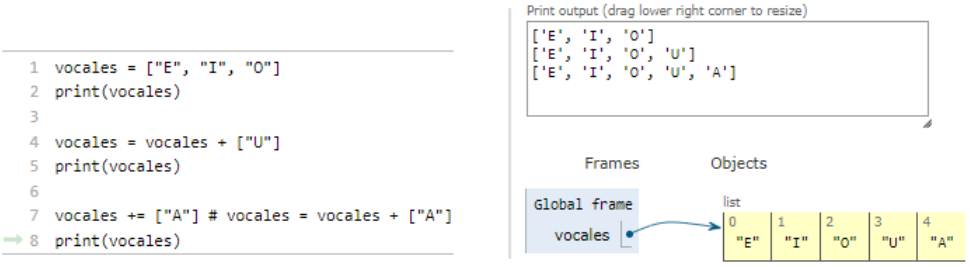
In [22]:
```
vocales = vocales + "Y"
vocales
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Input In [22], in <cell line: 1>()
----> 1 vocales = vocales + "Y"
      2 vocales

TypeError: can only concatenate list (not "str") to list
```

También se puede utilizar el operador += para agregar elementos a una lista:

In [23]:
```
vocales += ["A"] # vocales = vocales + ["A"]
vocales
```

Out[23]:
```
['E', 'I', 'O', 'U', 'A']
```



In [24]:
```
vocalesycons = ["Z"] + vocales
vocalesycons
```

Out[24]:
```
['Z', 'E', 'I', 'O', 'U', 'A']
```

In [25]:
```
vocales[-1]
```

Out[25]:
```
'A'
```

In [26]:
```
vocales[-3]
```

Out[26]:
```
'O'
```

In [27]:
```
vocales[-4]
```

Out[27]:
```
'I'
```

In [28]:
```
vocales[-5]
```

Out[28]:
```
'E'
```

In [29]:
```
vocales[-4] = 'Viernes'
vocales
```

Out[29]:
```
['E', 'Viernes', 'O', 'U', 'A']
```

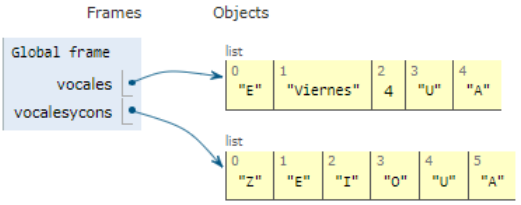
In [30]:
```
vocales[2] = 4
vocales
```

Out[30]:
```
['E', 'Viernes', 4, 'U', 'A']
```

```
10 vocalesycons = ["Z"] + vocales
11 print(vocalesycons)
12
13 print(vocales[-1])
14 print(vocales[-3])
15 print(vocales[-4])
16 print(vocales[-5])
17
18 vocales[-4] = 'Viernes'
19 print(vocales)
20
21 vocales[2] = 4
22 print(vocales)
```

Print output (drag lower right corner to resize)

```
['E', 'I', 'O']
['E', 'I', 'O', 'U']
['E', 'I', 'O', 'U', 'A']
['Z', 'E', 'I', 'O', 'U', 'A']
A
O
I
E
['E', 'Viernes', 'O', 'U', 'A']
['E', 'Viernes', 4, 'U', 'A']
```



Manipular elementos individuales de una lista

Cada elemento se identifica por su posición en la lista, teniendo en cuenta que comienzan con índice 0

In [31]:
```
fecha = [2, "Octubre", 1990]
fecha[0], fecha[1], fecha[2]
```

Out[31]:
```
(2, 'Octubre', 1990)
```

No se puede hacer referencia a elementos fuera de la lista:

In [32]:
```
fecha[3]
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
Input In [32], in <cell line: 1>()
----> 1 fecha[3]

IndexError: list index out of range
```

Se pueden utilizar números negativos (el último elemento tiene el índice -1 y los elementos anteriores tienen valores descendentes):

In [33]:
```
fecha[-1]
```

Out[33]:
```
1990
```

In [34]:
```
fecha[-2]
```

Out[34]:
```
'Octubre'
```

In [35]:
```
fecha[-3]
```

Out[35]:
```
2
```

Se puede modificar cualquier elemento de una lista haciendo referencia a su posición:

In [36]:
```
fecha[2] = 2000
fecha
```

Out[36]:
```
[2, 'Octubre', 2000]
```

Manipular sublistas

De una lista se pueden extraer sublistas, utilizando la notación **nombreDeLista[inicio : límite]**, donde inicio y límite hacen el mismo papel que en el tipo range(inicio, límite).

In [37]:
```
dias = ["Lunes", "Martes", "Miércoles", "Jueves", "Viernes", "Sábado", "Domingo"]
dias
```

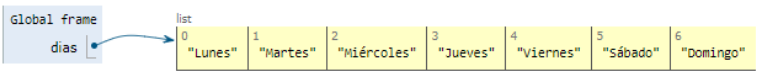
Out[37]:
```
['Lunes', 'Martes', 'Miércoles', 'Jueves', 'Viernes', 'Sábado', 'Domingo']
```

```
In [38]: dias[1 : 4] # Se extrae una lista con los valores 1, 2 y 3
Out[38]: ['Martes', 'Miércoles', 'Jueves']

In [39]: dias[4 : 5] # Se extrae una lista el valor 4
Out[39]: ['Viernes']

In [40]: dias[4 : 4] # Se extrae una lista vacía
Out[40]: []

In [41]: dias[: 4] # Se extrae hasta el valor 3 (o 4 no incluido)
Out[41]: ['Lunes', 'Martes', 'Miércoles', 'Jueves']

In [42]: dias[4: ] # Se extrae desde la posición 4
Out[42]: ['Viernes', 'Sábado', 'Domingo']

In [43]: dias[:] # Se extrae toda la lista
Out[43]: ['Lunes', 'Martes', 'Miércoles', 'Jueves', 'Viernes', 'Sábado', 'Domingo']

In [44]: dias[1 : -1] # Se extrae desde la posición 1 hasta la anteúltima
Out[44]: ['Martes', 'Miércoles', 'Jueves', 'Viernes', 'Sábado']
```

```
1  dias = ["Lunes", "Martes", "Miércoles", "Jueves",
2      "Viernes", "Sábado", "Domingo"]
3  print(dias)
4
5  print(dias[1 : 4])
6
7  print(dias[4 : 5]) # Se extrae una lista el valor 4
8
9  print(dias[4 : 4])
10
11 print(dias[: 4])
12
13 print(dias[4: ])
14
15 print(dias[ : ])
16
17 print(dias[1 : -1])
```

Print output (drag lower right corner to resize)

```
['Lunes', 'Martes', 'Miércoles', 'Jueves', 'Viernes', 'Sábado', 'Domingo']
['Martes', 'Miércoles', 'Jueves']
['Viernes']
[]
['Lunes', 'Martes', 'Miércoles', 'Jueves']
['Viernes', 'Sábado', 'Domingo']
['Lunes', 'Martes', 'Miércoles', 'Jueves', 'Viernes', 'Sábado', 'Domingo']
['Martes', 'Miércoles', 'Jueves', 'Viernes', 'Sábado']
```




```
In [45]: list_ = [10, 20, 30, 40]
         list_
Out[45]: [10, 20, 30, 40]

In [46]: list_[-1]
Out[46]: 40

In [47]: list_[1:-1]
Out[47]: [20, 30]

In [48]: list_[1:4]
Out[48]: [20, 30, 40]

In [49]: list_[0:4]
Out[49]: [10, 20, 30, 40]

In [50]: # lista[ 5 ] == lista[ -1 ]
         list_[ 3 ]
Out[50]: 40

In [51]: list_[ -1 ]
Out[51]: 40

In [52]: list_[ 1 : ]
Out[52]: [20, 30, 40]
```

In [53]:
```
len(list_)
```
Out[53]: 4

In [54]:
```
list_[1:-2]
```
Out[54]: [20]

In [55]:
```
list_[ 1 : ]
```
Out[55]: [20, 30, 40]

In [56]:
```
list_[ : 2 ]
```
Out[56]: [10, 20]

In [57]:
```
list_[ : ]
```
Out[57]: [10, 20, 30, 40]

In [58]:
```
list_[ 1 :  4]
```
Out[58]: [20, 30, 40]

Se puede modificar una lista modificando sublistas. De esta manera se puede modificar un elemento o varios a la vez e insertar o eliminar elementos.

In [59]:
```
letras = ["A", "B", "C", "D", "E", "F", "G", "H"]
letras
```
Out[59]: ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H']

In [60]:
```
letras[1:4] = ["X"] # Se sustituye la sublista ['B','C','D'] por ['X']
letras
```
Out[60]: ['A', 'X', 'E', 'F', 'G', 'H']

In [61]:
```
letras[1:4] = ["Y", "Z"] # Se sustituye la sublista ['X','E','F'] por ['Y','Z']
letras
```
Out[61]: ['A', 'Y', 'Z', 'G', 'H']

In [62]:
```
letras[0:1] = ["Q"] # Se sustituye la sublista ['A'] por ['Q']
letras
```
Out[62]: ['Q', 'Y', 'Z', 'G', 'H']

In [63]:
```
letras[3:3] = ["U","V"] # Inserta la lista ['U','V'] en la posición 3
letras
```
Out[63]: ['Q', 'Y', 'Z', 'U', 'V', 'G', 'H']

In [64]:
```
letras[0:3] = [] # Elimina la sublista ['Q','Y', 'Z']
letras
```
Out[64]: ['U', 'V', 'G', 'H']

In [65]:
```
letras = ["D", "E", "F"]
letras[3:3] = ["G", "H"] # Añade ["G", "H"] al final de la lista
letras
```
Out[65]: ['D', 'E', 'F', 'G', 'H']

In [66]:
```
letras[100:100] = ["I", "J"] # Añade ["I", "J"] al final de la lista
letras
```
Out[66]: ['D', 'E', 'F', 'G', 'H', 'I', 'J']

In [67]:
```
letras[-100:-50] = ["A", "B", "C"] # Añade ["A", "B", "C"] al principio de la lista
letras
```
Out[67]: ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J']

In [68]:
```
letras [6:8] = []
letras
```
Out[68]: ['A', 'B', 'C', 'D', 'E', 'F', 'I', 'J']

La palabra reservada del

La palabra reservada **del** permite eliminar un elemento o varios elementos a la vez de una lista, e incluso la misma lista:

In [69]:
```
letras = ["A", "B", "C", "D", "E", "F", "G", "H"]
letras
```

Out[69]:
```
['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H']
```

In [70]:
```
del letras[4] # Elimina la sublista ['E']
letras
```

Out[70]:
```
['A', 'B', 'C', 'D', 'F', 'G', 'H']
```

In [71]:
```
del letras[1:4] # Elimina la sublista ['B', 'C', 'D']
letras
```

Out[71]:
```
['A', 'F', 'G', 'H']
```

In [72]:
```
del letras # Elimina completamente la lista
letras
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Input In [72], in <cell line: 2>()
      1 del letras # Elimina completamente la lista
----> 2 letras

NameError: name 'letras' is not defined
```

Si se intenta borrar un elemento que no existe, se produce un error:

In [73]:
```
letras = ["A", "B", "C", "D", "E", "F", "G", "H"]
letras
```

Out[73]:
```
['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H']
```

In [74]:
```
del letras[10]
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
Input In [74], in <cell line: 1>()
----> 1 del letras[10]

IndexError: list assignment index out of range
```

Aunque si se hace referencia a sublistas, Python sí que acepta valores fuera de rango, pero lógicamente no se modifican las listas.

In [75]:
```
letras = ["A", "B", "C", "D", "E", "F", "G", "H"]
del letras[100:200] # No elimina nada
letras
```

Out[75]:
```
['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H']
```

Copiar una lista

Con variables de tipo entero, decimal o de cadena, es fácil tener una copia de una variable para conservar un valor que en la variable original se ha perdido:

In [76]:
```
a = 5
b = a # Hacemos una copia del valor de a
a, b
```

Out[76]:
```
(5, 5)
```

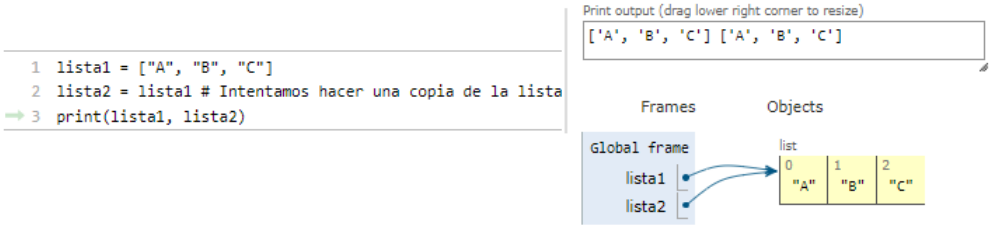
In [77]:
```
a = 4 # de manera que aunque cambiemos el valor de a ...
a, b # ... b conserva el valor anterior de a en caso de necesitarlo
```

Out[77]:
```
(4, 5)
```

Pero si hacemos esto mismo con listas, nos podemos llevar un sorpresa:

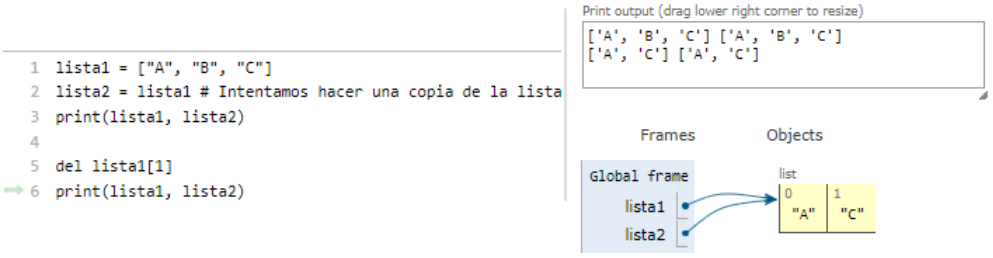
In [78]:
```
lista1 = ["A", "B", "C"]
lista2 = lista1 # Intentamos hacer una copia de la lista lista1
lista1, lista2
```

Out[78]:
```
(['A', 'B', 'C'], ['A', 'B', 'C'])
```



In [79]:
```
del lista1[1] # Eliminamos el elemento ['B'] de la lista lista1 ...
lista1, lista2 # ... pero descubrimos que también ha desaparecido de la lista lista2
```

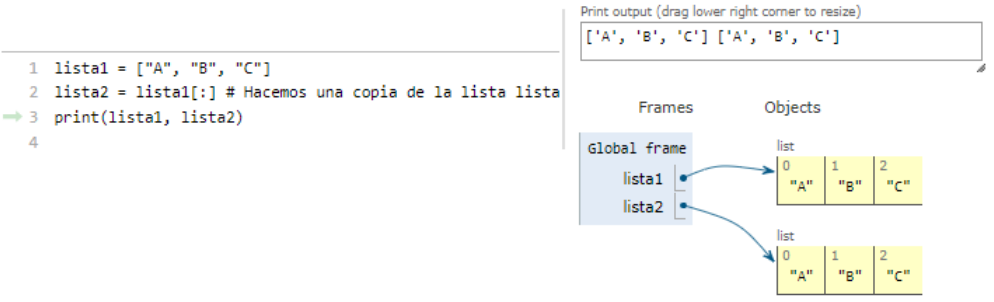
Out[79]: (['A', 'C'], ['A', 'C'])



El motivo de este comportamiento es que los enteros, decimales y cadenas son objetos inmutables y las listas son objetos mutables. Si queremos copiar una lista, de manera que conservemos su valor aunque modifiquemos la lista original debemos utilizar la notación de sublistas.

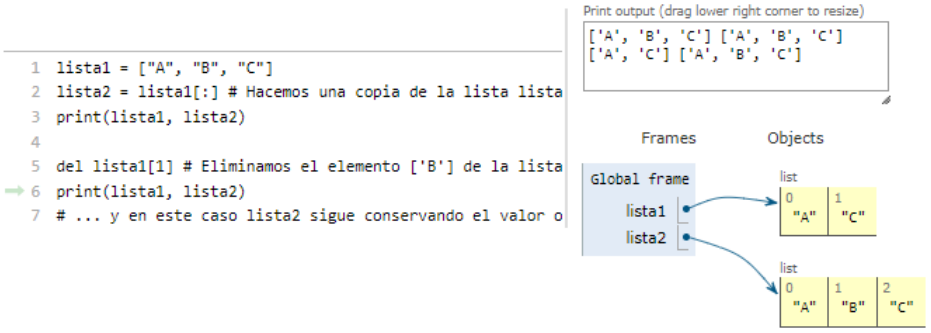
In [80]:
```
lista1 = ["A", "B", "C"]
lista2 = lista1[:] # Hacemos una copia de la lista lista1
lista1, lista2
```

Out[80]: (['A', 'B', 'C'], ['A', 'B', 'C'])



In [81]:
```
del lista1[1] # Eliminamos el elemento ['B'] de la lista lista1 ...
lista1, lista2 # ... y en este caso lista2 sigue conservando el valor original de lista1
```

Out[81]: (['A', 'C'], ['A', 'B', 'C'])



En el primer caso las variables lista1 y lista2 hacen referencia a la misma lista almacenada en la memoria. Por eso al eliminar un elemento de lista1, también desaparece de lista2.

Sin embargo en el segundo caso lista1 y lista2 hacen referencia a listas distintas (aunque tengan los mismos valores, están almacenadas en lugares distintos de la memoria). Por eso, al eliminar un elemento de lista1, no se elimina en lista2.

Recorrer una lista

Se puede recorrer una lista de principio a fin de dos formas distintas:

Una forma es recorrer **directamente** los elementos de la lista, es decir, que la variable de control del bucle tome los valores de la lista que estamos recorriendo:

In [82]:
```
letras = ["A", "B", "C"]
for i in letras:
    print(i, end=" ")
```

```
A B C
```

La otra forma es recorrer **indirectamente** los elementos de la lista, es decir, que la variable de control del bucle tome como valores los índices de la lista que estamos recorriendo (0,1 ,2 , etc.). En este caso, para acceder a los valores de la lista hay que utilizar letras[i]:

In [83]:
```
letras = ["A", "B", "C"]
for i in range(len(letras)):
    print(letras[i], end=" ")
```

```
A B C
```

In [84]:
```
letras = ["A", "B", "C"]
for i in letras:
    print(i)
```

```
A
B
C
```

La primera forma es más sencilla, pero sólo permite recorrer la lista de principio a fin y utilizar los valores de la lista. La segunda forma es más complicada, pero permite más flexibilidad, como muestran los siguientes ejemplos:

Recorrer una lista al revés

In [85]:
```
letras = ["A", "B", "C"]
for i in range(len(letras)-1, -1, -1):
    print(letras[i], end=" ")
```

```
C B A
```

Recorrer y modificar una lista

In [86]:
```
letras = ["A", "B", "C"]
print(letras)
for i in range(len(letras)):
    letras[i] = "X"
    print(letras)
```

```
['A', 'B', 'C']
['X', 'B', 'C']
['X', 'X', 'C']
['X', 'X', 'X']
```

In [87]:
```
letras = ["A", "B", "C"]
print(letras)
for i in range(len(letras)):
    letras[i] = "X"
print(letras)
```

```
['A', 'B', 'C']
['X', 'X', 'X']
```

Eliminar elementos de la lista

Para eliminar los elementos de una lista necesitamos recorrer la lista al revés. Si recorremos la lista de principio a fin, al eliminar un valor de la lista, la lista se acorta y cuando intentamos acceder a los últimos valores se produce un error de índice fuera de rango, como muestra el siguiente ejemplo en el que se eliminan los valores de una lista que valen "B":

Eliminar valores de una lista (incorrecto)

In [88]:
```
letras = ["A", "B", "C"]
print(letras)
for i in range(len(letras)):
    if letras[i] == "B":
        del letras[i]
print(letras)
```

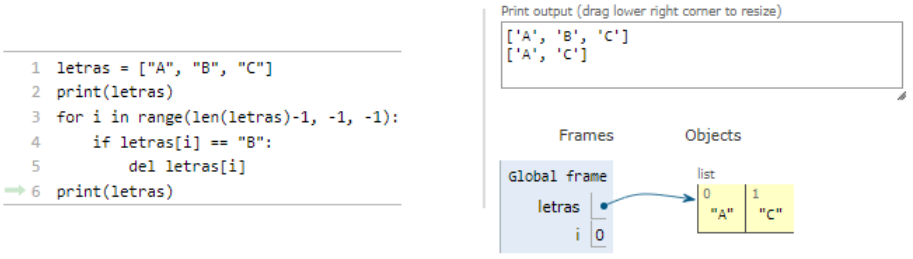
```
['A', 'B', 'C']

---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
Input In [88], in <cell line: 3>()
      2 print(letras)
      3 for i in range(len(letras)):
----> 4     if letras[i] == "B":
      5         del letras[i]
      6 print(letras)

IndexError: list index out of range
```

En este caso la primera instrucción del bloque comprueba si letras[i], es decir, letras[2] es igual a "B". Como letras[2] no existe (porque la lista tiene ahora sólo dos elementos), se produce un error y el programa se interrumpe. La solución es recorrer la lista en orden inverso, los valores que todavía no se han recorrido siguen existiendo en la misma posición que al principio.

Eliminar valores de una lista (correcto)

In [89]:
```
letras = ["A", "B", "C"]
print(letras)
for i in range(len(letras)-1, -1, -1):
    if letras[i] == "B":
        del letras[i]
print(letras)
```

```
['A', 'B', 'C']
['A', 'C']
```



Saber si un valor está o no en una lista

Para saber si un valor está en una lista se puede utilizar el operador **in**. La sintaxis sería "elemento in lista" y devuelve un valor lógico:

True si el elemento está en la lista.

False si el elemento no está en la lista.

Por ejemplo, el programa siguiente comprueba si el usuario es una persona autorizada:

In [92]:
```
personas_autorizadas = ["Alberto", "Carmen"]
nombre = input("Dígame su nombre: ")
if nombre in personas_autorizadas:
    print("Está autorizado")
else:
    print("No está autorizado")
```

```
Dígame su nombre: Pepa
No está autorizado
```

Para saber si un valor no está en una lista se pueden utilizar los operadores **not in**. La sintaxis sería "elemento not in lista" y devuelve un valor lógico:

True si el elemento no está en la lista.

False si el elemento está en la lista.

Por ejemplo, el programa siguiente comprueba si el usuario es una persona autorizada:

In [93]:
```
personas_autorizadas = ["Alberto", "Carmen"]
nombre = input("Dígame su nombre: ")
if nombre not in personas_autorizadas:
    print("No está autorizado")
else:
    print("Está autorizado")
```

```
Dígame su nombre: Carmen
Está autorizado
```