

Python: Llistes(2). Diccionaris (1)

Sessió 23

LP 2019-20

Llistes (2)

Llistes: Creació, accés i recorregut

Creació de llistes

```
llista = [1, 'a', 2, 'b']  
print (llista)
```

Accés a les llistes

```
print (llista[1])  
print (llista[-1])  
print (llista[1:3])  
print (llista[1:])  
print (llista[:2])  
print (llista[0::2])
```

Recorregut

```
llista = [1, 'a', 2, 'b']  
for i in range(len(llista)):  
    print (llista[i])
```

```
for x in llista:  
    print (x)
```

```
for index, valor in enumerate(llista):  
    print (index, valor)
```

Recorregut de vàries llistes

```
llista1 = [1, 2, 3]  
llista2 = ['a', 'b', 'c']  
for x, y in zip(llista1, llista2):  
    print (x, y)
```

Exercici

Suposem que tenim un fitxer com aquest amb la informació del nom dels clients d'una empresa i dels codis de les comandes que ha fet cada client

comandes.txt

```
client1 101 102 103
client2 201
client3 301 302
client4 401 402 403 404
client5 501 502 503 504 505 506
client6 601
client7 701 702 703
client8 801 802 803 804
client9 901
```

Volem fer una funció que llegeixi el fitxer i guardi tota la informació utilitzant llistes en Python

Lectura de fitxers

Provem aquest codi...

```
linies = []  
fitxer = open("comandes.txt", "rt")  
for linia in fitxer:  
    print (linia)  
    linies.append(linia)
```

Obertura del fitxer en mode text per lectura

- Lectura línia a línia del fitxer
- `fitxer` és un conjunt iterable de línies
- Cada línia es guarda com un `string`

append: afegeix un element al final de la llista

- Quin és el contingut de `linies` després de l'execució?

I ara provem aquest altre codi...

```
paraules = linies[0].split()  
print (paraules)
```

Retorna una llista amb la separació del `string` en paraules (separades per espais en blanc)

- Es poden utilitzar altres separadors: `str.split('separador')`
- Podem utilitzar moltes funcions i operadors que treballen amb strings:

<https://docs.python.org/2/library/stdtypes.html#string-methods>


- Quin és el contingut de `paraules`?

Exercici

Volem fer una funció que llegeixi el fitxer de comandes i guardi tota la informació utilitzant una llista en què cada element guardi la informació d'un client en una llista on el primer element sigui el nom del client i el segon element una altra llista amb el codi de totes les comandes del client, tal com es mostra a continuació:

comandes.txt

```
client1 101 102 103
client2 201
client3 301 302
client4 401 402 403 404
client5 501 502 503 504
client6 601
client7 701 702 703
client8 801 802 803 804
client9 901
```



```
[['client1', ['101', '102', '103']],
 ['client2', ['201']],
 ['client3', ['301', '302']],
 ['client4', ['401', '402', '403', '404']],
 ['client5', ['501', '502', '503', '504']],
 ['client6', ['601']],
 ['client7', ['701', '702', '703']],
 ['client8', ['801', '802', '803', '804']],
 ['client9', ['901']]]
```

Llistes de llistes

```
llista = ['a', [1, 2, 3], 'b', [4, 5, 6]]
print (llista[0])
-> a
print (llista[1])
-> [1, 2, 3]
print (llista[1][0])
-> 1
llista.append(['c', [7, 8, 9]])
print (llista)
-> ['a', [1, 2, 3], 'b', [4, 5, 6], ['c', [7, 8, 9]]]
```

Un element d'una llista pot ser una altra llista

➤ Com podem accedir al 9?


```
print (llista[4][1][2])
```

Exercici

Volem fer una funció que llegeixi el fitxer de comandes i guardi tota la informació utilitzant una llista en què cada element guardi la informació d'un client en una llista on el primer element sigui el nom del client i el segon element una altra llista amb el codi de totes les comandes del client, tal com es mostra a continuació:

comandes.txt

```
client1 101 102 103
client2 201
client3 301 302
client4 401 402 403 404
client5 501 502 503 504
client6 601
client7 701 702 703
client8 801 802 803 804
client9 901
```



```
[['client1', ['101', '102', '103']],
 ['client2', ['201']],
 ['client3', ['301', '302']],
 ['client4', ['401', '402', '403', '404']],
 ['client5', ['501', '502', '503', '504']],
 ['client6', ['601']],
 ['client7', ['701', '702', '703']],
 ['client8', ['801', '802', '803', '804']],
 ['client9', ['901']]]
```


Ejemplo de uso

Volem fer una funció que rebi com a paràmetre una llista de comandes com la que hem generat a l'exercici anterior i ens retorni una llista només amb els noms de tots els clients.

comandes

```
[['client1', ['101', '102', '103']],  
 ['client2', ['201']],  
 ['client3', ['301', '302']],  
 ['client4', ['401', '402', '403', '404']],  
 ['client5', ['501', '502', '503', '504']],  
 ['client6', ['601']],  
 ['client7', ['701', '702', '703']],  
 ['client8', ['801', '802', '803', '804']],  
 ['client9', ['901']]]
```

noms

```
['client1',  
 'client2',  
 'client3',  
 'client4',  
 'client5',  
 'client6',  
 'client7',  
 'client8',  
 'client9']
```

Primera versió

```
def nomsComandes(comandes):  
    noms = []  
    for c in comandes:  
        noms.append(c[0])  
    return noms
```

Construcció de llistes

Provem aquest codi...

```
llista = [x**2 for x in list(range(10))]  
print (llista)
```

➤ Quin és el contingut de llista després de l'execució?

-> [0, 1, 4, 9, 16, 25, 36, 49, 64, 81]

opcional

[<expressió> for <element> in <llista> *{if <condició>}*]

- Construeix una nova llista aplicant l'expressió a cada element de la llista
- Opcionalment es pot posar una condició que s'ha de complir per posar l'element resultant a la llista

```
llista = [x**2 for x in list(range(10)) if x%2 == 0]  
print (llista)  
-> [0, 4, 16, 36, 64]
```

Ejemplo de uso

Volem fer una funció que rebi com a paràmetre una llista de comandes com la que hem generat a l'exercici anterior i ens retorni una llista només amb els noms de tots els clients.

comandes

```
[['client1', ['101', '102', '103']],  
 ['client2', ['201']],  
 ['client3', ['301', '302']],  
 ['client4', ['401', '402', '403', '404']],  
 ['client5', ['501', '502', '503', '504']],  
 ['client6', ['601']],  
 ['client7', ['701', '702', '703']],  
 ['client8', ['801', '802', '803', '804']],  
 ['client9', ['901']]]
```

noms

```
['client1',  
 'client2',  
 'client3',  
 'client4',  
 'client5',  
 'client6',  
 'client7',  
 'client8',  
 'client9']
```

Segona versió (utilitzant la construcció de llistes)

```
def nomsComandes(comandes):  
    noms = [c[0] for c in comandes]  
    return noms
```

Primera versió

```
def nomsComandes(comandes):  
    noms = []  
    for c in comandes:  
        noms.append(c[0])  
    return noms
```

Exercici

Volem fer una funció que permeti afegir comandes a la llista de comandes obtinguda amb l'exercici anterior. La funció rebrà com a paràmetres la llista de comandes, el nom del client i una llista amb les comandes que es volen afegir. Si el client ja està a la llista, afegirà les noves comandes a les ja existents del client. Si no existeix, afegirà el client i les seves comandes com un nou element a la llista de comandes.

comandes

```
[['client1', ['101', '102', '103']],  
 ['client2', ['201']],  
 ['client3', ['301', '302']],  
 ...  
 ['client9', ['901']]]
```

```
def modificaClient(comandes, nomClient, novesComandes)
```

Operacions sobre llistes

Cerca en una llista

```
llista = list(range(10))
x = int(input())
if x in llista:
    print(x, 'està a la llista')
else:
    print(x, 'no esta a la llista')
```

Recordem: input retorna un string. Ho hem de convertir a int

<valor> in <llista>: retorna true si valor està a la llista

Concatenació de llistes

```
llista1 = list(range(5))
llista2 = list(range(5,10))
llista3 = llista1 + llista2
print (llista3)
-> [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Retorna una nova llista afegint els elements de llista2 al final de llista1

Eliminació d'elements de la llista

```
llista = list(range(10))
llista[0:2] = []
-> [2, 3, 4, 5, 6, 7, 8, 9]
llista[2] = []
-> [2, 3, [], 5, 6, 7, 8, 9]
```

llista[ini:final] = [] elimina elements entre la posició ini i final - 1

Compte! No elimina el valor. El substitueix. Per eliminar: llista[2:3] = []

Operacions sobre llistes

Mètodes de les llistes

```
llista = ['a', 'b', 'c']  
llista.append('d')  
-> ['a', 'b', 'c', 'd']  
posició = llista.index('b')  
print(posició)  
-> 1  
llista.remove('b')  
-> ['a', 'c', 'd']  
llista.pop(2)  
-> ['a', 'c']
```

Afegeix un element al final de la llista

Retorna l'índex d'un element de la llista
Si l'element no existeix genera un error d'execució

Elimina la primera aparició d'un element de la llista
Si l'element no existeix genera un error d'execució

Elimina l'element que ocupa la posició que es
passa com a paràmetre. Retorna l'element
eliminat. Si índex fora de rang genera un error
d'execució.

Altres mètodes de les llistes:

<https://docs.python.org/3.7/tutorial/datastructures.html#more-on-lists>

Funcions sobre llistes

```
llista = list(range(10))  
print(len(llista), sum(llista), max(llista), min(llista))
```

Exercici

Volem fer una funció que permeti afegir comandes a la llista de comandes obtinguda amb l'exercici anterior. La funció rebrà com a paràmetres la llista de comandes, el nom del client i una llista amb les comandes que es volen afegir. Si el client ja està a la llista, afegirà les noves comandes a les ja existents del client. Si no existeix, afegirà el client i les seves comandes com un nou element a la llista de comandes.

comandes

```
[['client1', ['101', '102', '103']],  
 ['client2', ['201']],  
 ['client3', ['301', '302']],  
 ...  
 ['client9', ['901']]]
```

```
def modificaClient(comandes, nomClient, novesComandes)
```

Exercici

Volem fer una funció que elimini comandes a la llista de comandes que hem utilitzat en els exercicis anteriors. La funció rebrà com a paràmetres la llista de comandes, el nom del client i el codi de la comanda que es vol eliminar. Si el client o la comanda no existeixen s'ha de mostrar un missatge d'error.

comandes

```
[['client1', ['101', '102', '103']],  
 ['client2', ['201']],  
 ['client3', ['301', '302']],  
 ...  
 ['client9', ['901']]]
```

```
def eliminaComanda(comandes, nomClient, codiComanda)
```


Exercici

1. Volem fer una funció que donada una llista de comandes ens retorni una altra llista que contingui per cada client el nº de comandes que ha fet.

comandes

```
[['client1', ['101', '102', '103']],  
 ['client2', ['201']],  
 ['client3', ['301', '302']],  
 ...  
 ['client9', ['901']]]
```

nComadesClient

```
[['client1', 3],  
 ['client2', 1],  
 ['client3', 2],  
 ...  
 ['client9', 1]]
```

2. Volem fer una funció que donada una llista de comandes ens retorni el número total de comandes fetes per tots els clients
3. Volem fer una funció que donada una llista de comandes ens retorni el nom dels clients que han fet més d'una comanda.

Exercici

Què passa si fem el següent i perquè:

```
l1=[0]*5
l2=[0 for x in range(5)]
l3=[0,0,0,0,0]
import sys
print(sys.getsizeof(l1))
print(sys.getsizeof(l2))
print(sys.getsizeof(l3))
```

```
l4=[10,5,7,8,9]
l5=l4[1:4]
print(l5)
l5[1]=23
print(l4)
print(l5)
```

```
a=[1,2]
b=[3,4]
c=[5,6]
l1=[a,b]
l2=l1
l3=list(l1)
print(l1)
print(l2)
print(l3)
```

```
l1.append(c)
print(l1)
print(l2)
print(l3)
```

```
l2[0][1]=55
print(l1)
print(l2)
print(l3)
print(a)
```

**descarregueu codi a
Caronte:
provesS23.py**

TIPUS SEQÜENCIALS A PYTHON

Python té 3 tipus de dades anomenades seqüencials:

- **list**
- **tuple**
- **str**

Totes elles:

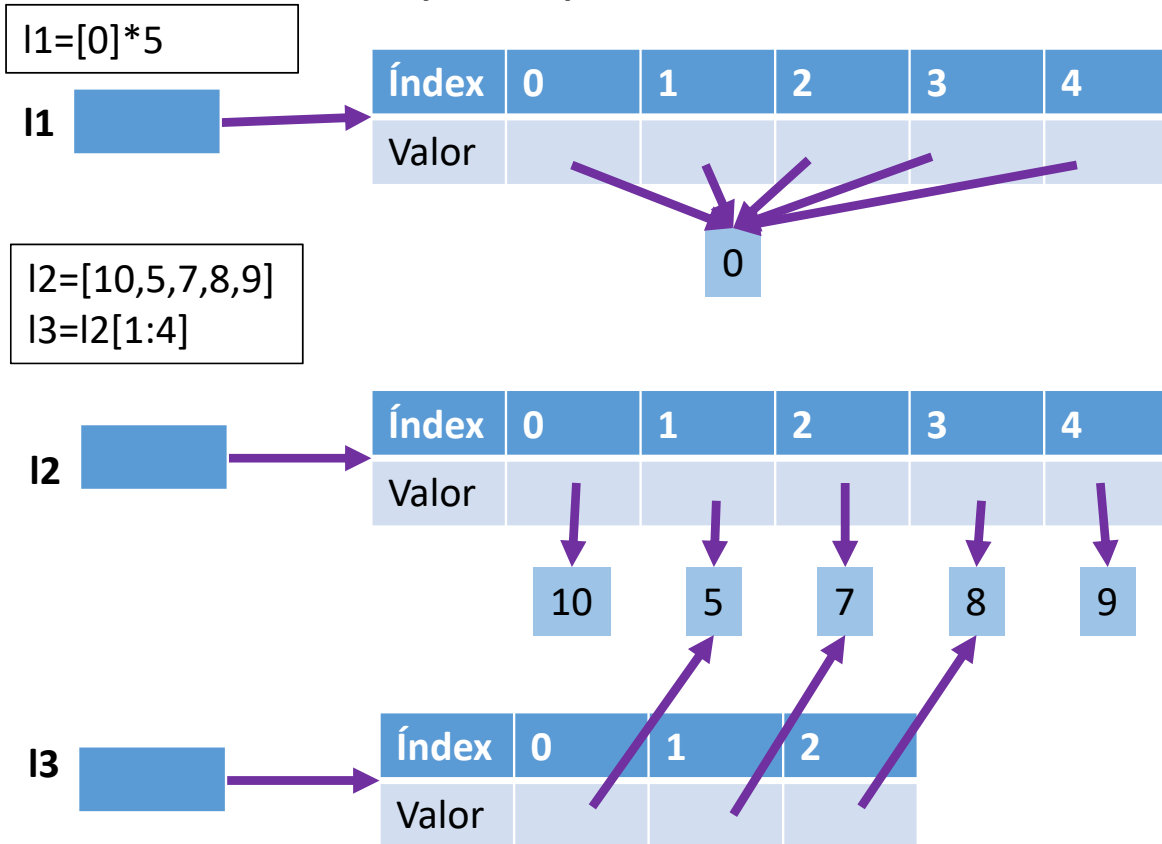
- Suporten indexació per accedir a elements de la seqüència utilitzant la notació `s[i]`
- Es basen en el concepte de baix nivell **ARRAY**

És important conèixer el seu baix nivell per:

- Escriure programes eficients
- Evitar errors i comportaments estranys al vostre codi.

LLISTES COM ARRAY REFERENCIAL

Tenir llistes i tuples que són estructures referencials fa que tinguem:



Un element referenciat per diferents posicions d'una mateixa llista

Nota: Tots els int del -5 al 256 estan fixes a una determinada posició

Dues llistes poden referenciar als mateixos elements.

LLISTES COM ARRAY REFERENCIAL

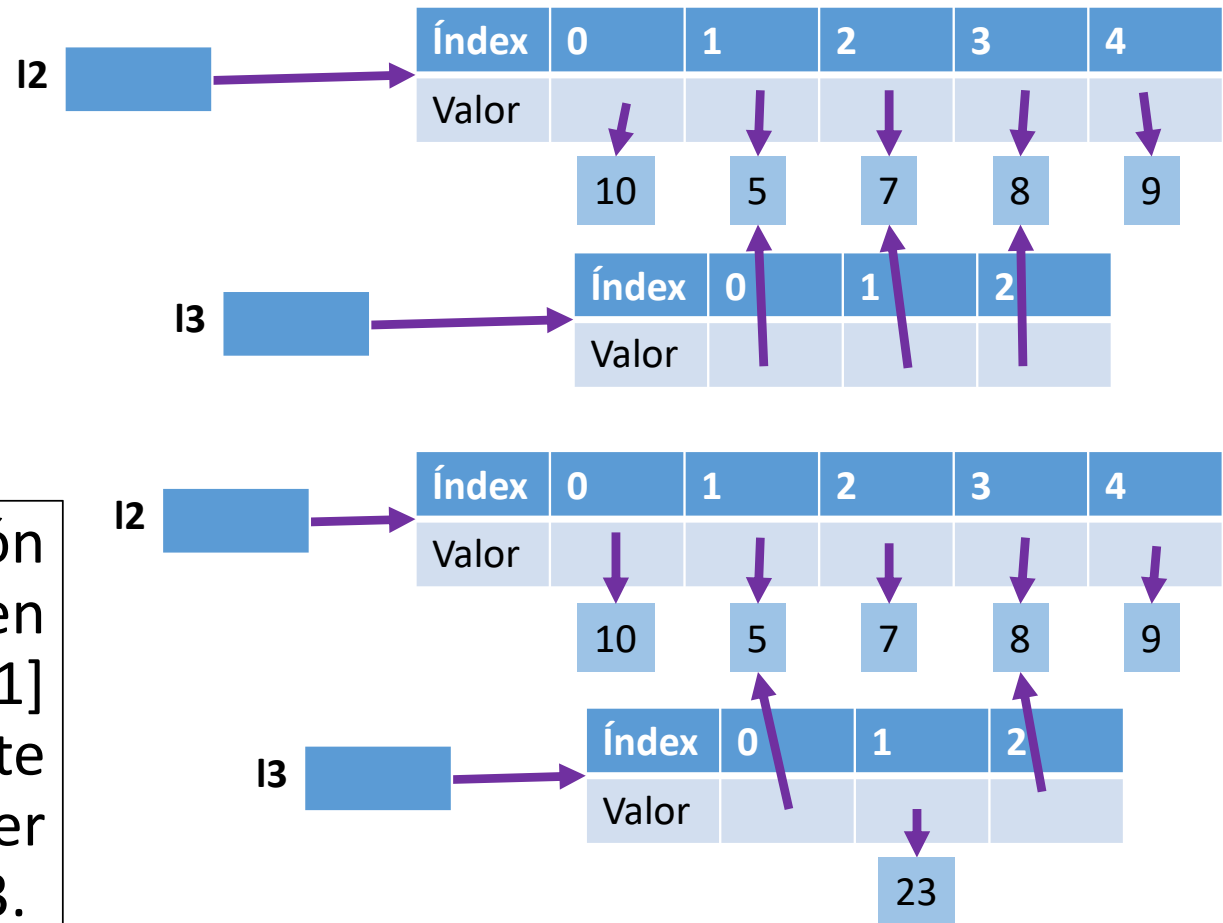
Què passava realment al fer:

```
I2=[10,5,7,8,9]  
I3=I2[1:4]
```

I si fèiem després:

```
I3[1]=23
```

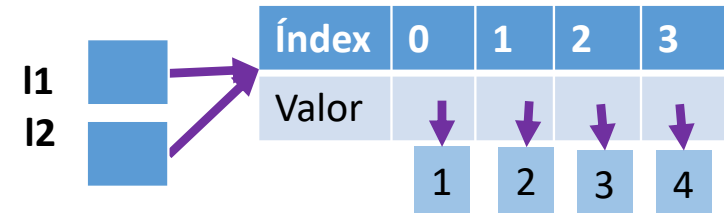
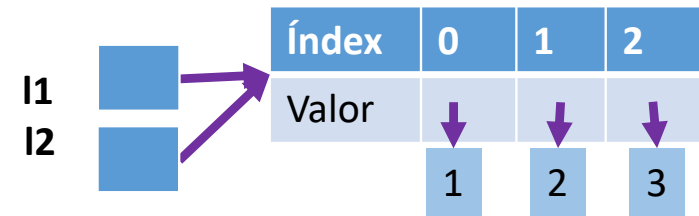
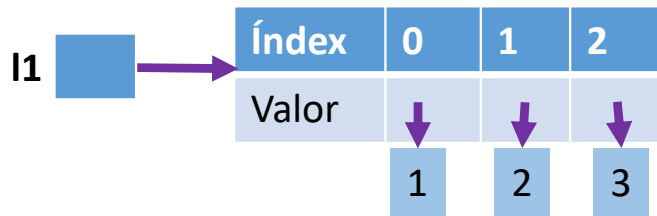
Com els integers són immutables no es poden canviar i per tant fer I3[1] significa crear un objecte nou que serà apuntat per la posició 1 de la llista I3.



LLISTES COM ARRAY REFERENCIAL

Tornant al codi inicial:

```
l1=[1,2,3]
l2=l1
print(l1)
print(l2)
l2.append(4)
print(l1)
print(l2)
```



Què passava realment?

Per què?

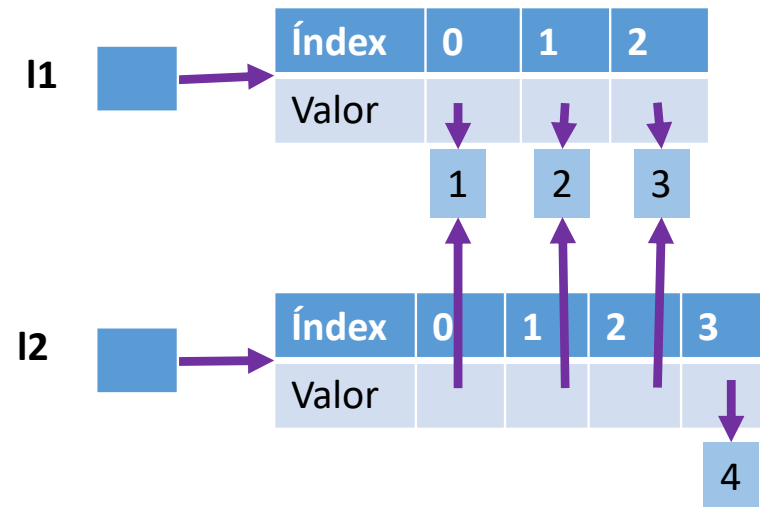
En aquest cas no tenim dues llistes sinó només una apuntada per dos identificadors: **l1** i **l2**. Si modifiquem un també modificarem l'altre.

LLISTES COM ARRAY REFERENCIAL

I quan hem executat el següent codi:

```
l1=[1,2,3]
l2=list(l1)
print(l1)
print(l2)
l2.append(4)
print(l1)
print(l2)
```

Què ha fet?

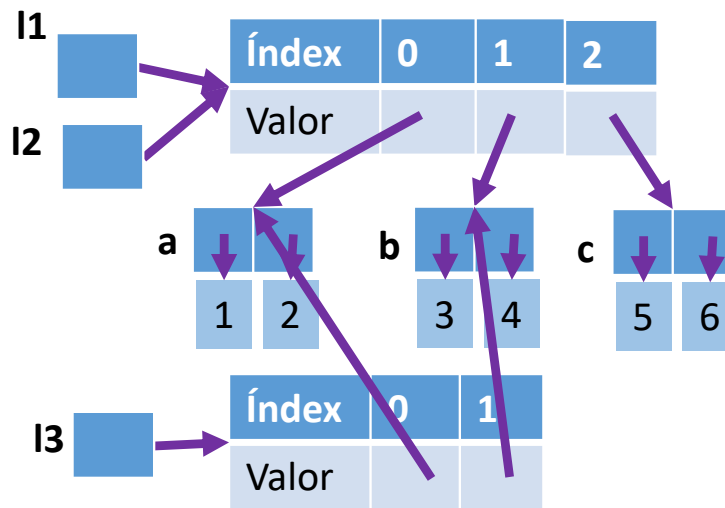


l2 és una **shallow copy** de l1 perquè comparteixen els valors. Però com els enters són immutables això no és un problema.

LLISTES COM ARRAY REFERENCIAL

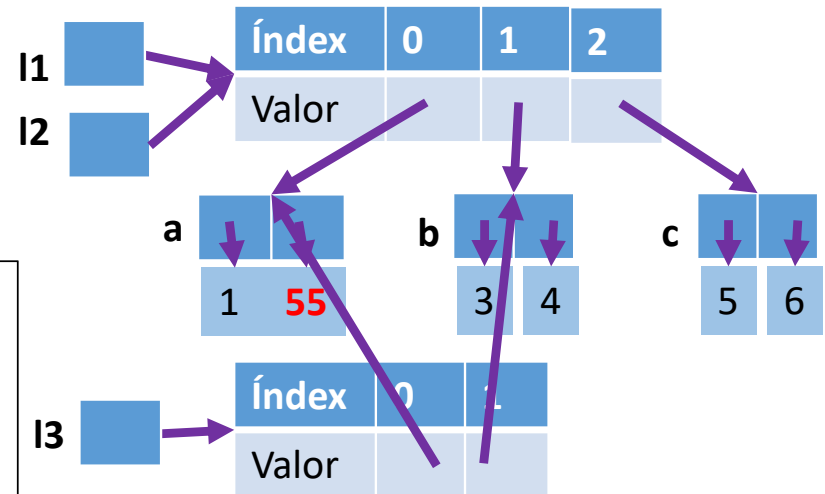
Però quan hem canviat els integers per llistes. Què ha passat?

```
a=[1,2]
b=[3,4]
c=[5,6]
l1=[a,b]
l2=l1
l3=list(l1)
print(l1)
print(l2)
print(l3)
l1.append(c)
print(l1)
print(l2)
print(l3)
```



I a l'afegir el següent canvi què ha passat?

```
l2[0][1]=55
print(l1)
print(l2)
print(l3)
print(a)
```

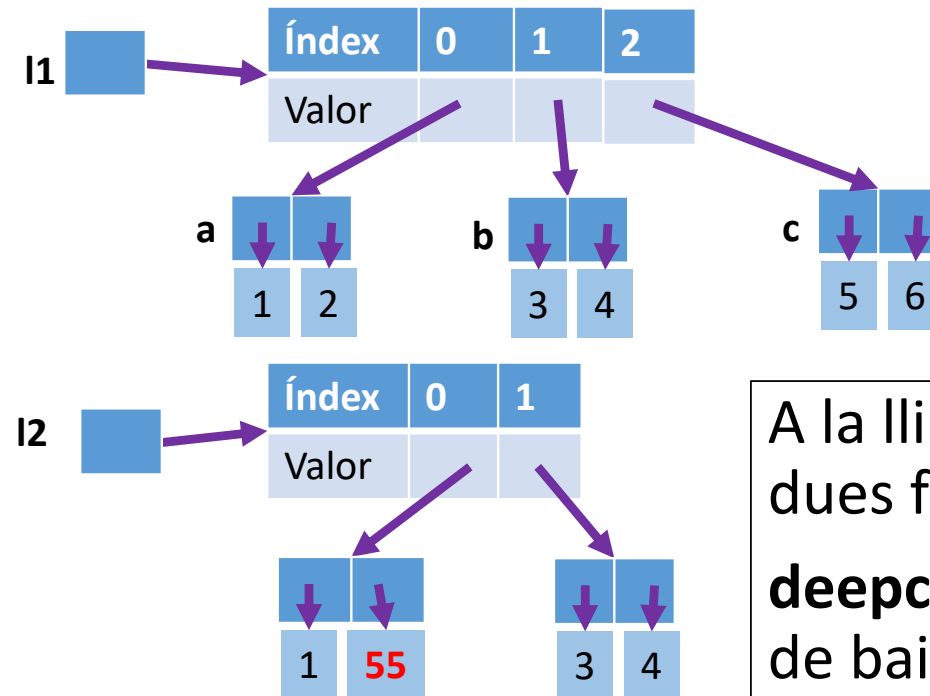


Les llistes no són inmutables, al modificar una queda modificada a les tres llistes i a la variable.

LLISTES COM ARRAY REFERENCIAL

Solució: **deepcopy**

```
import copy
a=[1,2]
b=[3,4]
c=[5,6]
l1=[a,b]
l2=deepcopy(l1)
print(l1)
print(l2)
l1.append(c)
print(l1)
print(l2)
l2[0][1]=55
print(l1)
print(l2)
print(a)
```



I ara? Què ha passat?

A la llibreria **copy** tenim dues funcions:

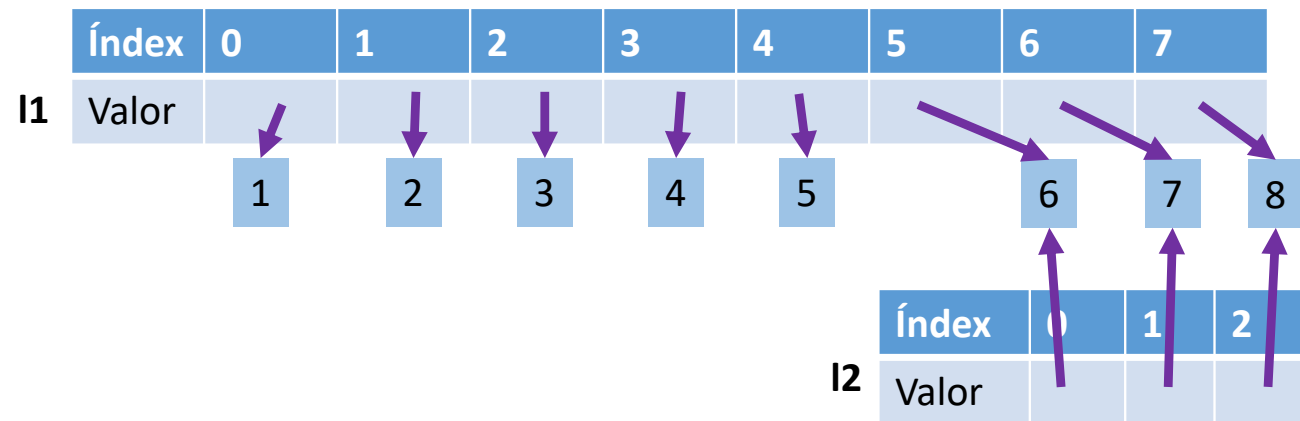
deepcopy: Fa una còpia de baix nivell.

copy: Fa una shallow copy. (còpia superficial)

LLISTES COM ARRAY REFERENCIAL: Curiositats

extend: no copia els elements sinó que fa referència als ja existents.

```
l1=[1,2,3,4,5]  
l2=[6,7,8]  
l1.extend(l2)  
print(l1)  
print(l2)
```



Diccionaris (1)

Recordem...

Una **taula Hash** és un contenidor associatiu (tipus Diccionari) de parelles <clau,valor> que permet recuperar una valor a partir de la seva clau que és única.

- Hem de poder trobar la clau ràpidament: $O(1)$
- Hem de poder inserir de forma eficient una parella clau – valors: $O(1)$
- Hem de poder esborrar una parella clau – valors (la complexitat és menys important)

```
class Hash
{
public:
    Hash();
    ~Hash();
    string& operator[](const string& clau);
    void insert(const string& clau, const string& valor);
    bool find(const string& clau, string& valor) const;
    bool esborra(const string& clau);
    ...
private:
    std::vector<std::pair<string, string>> m_diccionari;
    ...
};
```

Llibreria estàndar C++:

- `std::unordered_map`

Python:

- Diccionaris

Diccionaris

Creació de diccionaris

- Un diccionari és un conjunt de parelles (clau, valor)

```
notes = {}  
notes = dict()
```

Creació d'un diccionari buit

Inicialització del diccionari

```
notes = {'joan': 5, 'anna': 4, 'marta' : 10, 'jordi' : 8}  
notes = dict([('joan', 5), ('anna', 4), ('marta', 10), ('jordi', 8) ])
```

- L'ordre dels elements no està definit

```
print (notes)
```

Accés als diccionaris

- L'accés al valor dels elements es fa a partir de la clau.

```
print (notes['joan'])  
notes['joan'] = 8  
notes['maria'] = 25
```

- Si la clau no està al diccionari genera un error en temps d'execució

- Si la clau ja està al diccionari modifica el valor
- Si la clau no està al diccionari l'afegeix

```
print (notes)
```

Diccionaris

Cerca en diccionaris

```
if 'joan' in notes:  
    print (notes['joan'])
```

<clau> in <diccionari>

- Retorna true si hi ha algun valor al diccionari amb aquesta clau.

```
if 'anna' not in notes:  
    notes['anna'] = 5
```

Eliminació en diccionaris

```
if 'joan' in notes:  
    del (notes['joan'])  
print (notes)
```

del (diccionari[<clau>])

- Elimina el valor corresponent a la clau del diccionari
- Si la clau no existeix al diccionari genera una excepció

Exercici

Abans vam fer unes funcions per llegir d'un fitxer i modificar i eliminar les comandes dels clients d'una empresa guardades en una llista on a cada element de la llista el primer valor era el nom del client i el segon valor la llista de les seves comandes:

```
def llegirFitxer(nomFitxer):  
    fitxer = open(nomFitxer, "rt")  
    comandes = []  
    for linia in fitxer:  
        paraules = linia.split()  
        comandes.append([paraules[0], paraules[1:]])  
    return comandes
```

comandes

```
[['client1', ['101', '102', '103']],  
 ['client2', ['201']],  
 ['client3', ['301', '302']],  
 ...  
 ['client9', ['901']]]
```

Exercici

Abans vam fer unes funcions per llegir d'un fitxer i modificar i eliminar les comandes dels clients d'una empresa guardades en una llista on a cada element de la llista el primer valor era el nom del client i el segon valor la llista de les seves comandes:

```
def modificaClient(comandes, nomClient, novesComandes):  
    noms = [c[0] for c in comandes]  
    if nomClient in noms:  
        posicio = noms.index(nomClient)  
        comandes[posicio][1] = comandes[posicio][1] + novesComandes  
    else:  
        comandes.append([nomClient, novesComandes] )
```

```
def eliminaComanda(comandes, nomClient, codiComanda):  
    noms = [c[0] for c in comandes]  
    if nomClient in noms:  
        posicio = noms.index(nomClient)  
        if codiComanda in comandes[posicio][1]:  
            comandes[posicio][1].remove(codiComanda)  
        else:  
            print ('Error. Comanda no existent')  
    else:  
        print ('Error. Client no existent')
```

comandes

```
[[ 'client1', ['101', '102', '103']],  
 [ 'client2', ['201']],  
 [ 'client3', ['301', '302']],  
 ...  
 [ 'client9', ['901']]]
```


Exercici

Volem **modificar** aquestes funcions per fer que les comandes es guardin en un **diccionari** on la **clau** sigui el **nom del client** i el valor la llista amb els codis de les comandes

comandes

```
[['client1', ['101', '102', '103']],  
 ['client2', ['201']],  
 ['client3', ['301', '302']],  
 ...  
 ['client9', ['901']]  
]
```

comandes

```
{'client1': ['101', '102', '103'],  
 'client2': ['201'],  
 'client3': ['301', '302'],  
 ...  
 'client9': ['901']  
}
```

Diccionaris

Recorregut

```
print [notes.keys()]
```

```
for nom in notes.keys():  
    print (nom)
```

```
llista = list(notes.keys())
```

- Retorna un iterable amb totes les claus del diccionari en ordre d'inserció:
-> `dict_keys(['joan', 'anna', 'marta', 'jordi'])`
- Aquest iterable es pot utilitzar per recórrer totes les claus del diccionari
- Si el volem convertir en una llista per manipular-lo:
`list(notes.keys())`

```
print [notes.values()]
```

```
for nota in notes.values():  
    print (nota)
```

- Retorna un iterable amb tots els valors del diccionari en ordre d'inserció:
-> `dict_values([5, 4, 10, 8])`

Diccionaris

Recorregut

```
print [notes.items()]
```

```
for n in notes.items():  
    print (n)
```

- Retorna una llista amb les parelles (clau, valor) del diccionari en ordre d'inserció:
[('marta', 10), ('jordi', 8), ('joan', 8), ('anna', 4)]
- Aquesta llista es pot utilitzar per recórrer tots els valors del diccionari

```
for n in notes.items():  
    print [(n[0], n[1])]
```

Cada parella (clau, valor) és el que s'anomena una **tupla**.

- Les tuples són similars a les llistes i s'accedeixen igual (per posició de l'element) però són objectes *immutable* (no es pot canviar el seu valor)

```
for [k,v in notes.items():]  
    print (k, v)
```

Les tuples es poden desempaquetar (*unpack*) i utilitzar variables diferents per accedir a cada element de la tupla

```
for k,v in [sorted(notes.items())]:  
    print (k, v)
```

Retorna la llista d'elements del diccionari ordenada per la clau

Més informació sobre recorreguts de seqüències i diccionaris:

<https://docs.python.org/3.7/tutorial/datastructures.html#looping-techniques>

Exercici

A la sessió anterior vam fer una funció que donada una llista de comandes com la de l'exercici anterior retornava una altra llista que on per cada client tenim el nº de comandes que ha fet.

```
def nComandes(comandes):  
    nComandesClient = [[c[0], len(c[1])] for c in comandes]  
    return nComandesClient
```

comandes

```
{'client1': ['101', '102', '103'],  
 'client2': ['201'],  
 'client3': ['301', '302'],  
 ...  
 'client9': ['901']  
}
```

Volem **modificar** aquesta funció perquè treballi amb el **diccionari de comandes** que hem creat a l'exercici anterior, enlloc de treballar amb la llista. **Assegureu-vos** que la **llista** resultant queda **ordenada** per nom de client

comandes

```
[['client1', ['101', '102', '103']],  
 ['client2', ['201']],  
 ['client3', ['301', '302']],  
 ...  
 ['client9', ['901']]  
]
```

comandes

```
{'client1': ['101', '102', '103'],  
 'client2': ['201'],  
 'client3': ['301', '302'],  
 ...  
 'client9': ['901']  
}
```