Chapitre 1 : Parcours des listes

I- Le vocabulaire

- Une liste est irétable ce qui signifie que chaque élément de cette dernière peut être lue
- Une liste est indiçable ce qui signifie que chaque élément possède un indice

II- Présentation des listes

a) initialisation ou création d'une liste vide

In [1]:
```
L=[]
```

b)Création d'une liste par extension (avec des éléments souhaités)

L=[elt_1,elt_2,elt_3,..,elt_n]# cette liste contient n éléments=> len(L)=n

Liste	$L = [elt_1, elt_2, \ldots\ldots, elt_{n-1}, elt_n]$
Index	$0 \quad 1 \quad \ldots\ldots \quad n-2 \quad n-1$
Index	$-len(L) \quad -len(L)+1 \quad -2 \quad -1$

c)Obtenir un élément d'une liste

In [2]:
```
L=[5,14,9,15]
L[0]#spécifie la valeur du 1e élément (ici 5)
L[1]#spécifie la valeur du 2e élément (ici 14)
L[-1]#spécifie le dernier élément tout comme L[len(L)-1]
# x in L renvoie True si x est dans L (False sinon)
```

Out[2]: 15

d)Obtenir une sous liste L1 à partir de L

L1=L[i:j:pas]

- i indice de début
- j indice de fin exclu
- pas est l'intervelle entre deux valeurs Rq : Par défaut pas=1 si on écrit L[i:j]

e)Modifier un ou des éléments d'une liste

- L[i]=a affecte la valeur a à L[i]
- L[i:j+1]=L2 affecte la liste L2 dans L entre l'indice i et j inclus

d) Opérations sur les listes

- la méthode append : L.append(x) rajoute x à la fin de L
- Concaténation de deux listes L1 et L2 : L1+L2
- L1*3 revient à L1+L1+L1

In [2]:
```
#on peut évoquer la méthode extend qui permet de concaténr en place
L1=[0,1]
L2=[2,3]
print(id(L1))
L1.extend(L2)
print(L1,id(L1))
```

```
102861960
[0, 1, 2, 3] 102861960
```

III- Parcours des listes à l'aide de boucles

On distingue deux types de boucles :

a) Les boucles non conditionnelles "for":

Ces boucles (bornées) permettent d'itérer une liste un nombre fini de fois

In [3]:
```
somme=0#initialisation
L=[0,1,2,3,4,5]
for i in L:#itération sur les élément de L
    somme=somme+i#typage dynamique=> on obtient la somme des élémen
print(somme)
```

```
15
```

On utilise souvent la fontion $range(début, finexclue, pas)$, on peut aussi écrire range(fin) qui impose un pas de 1 et un début à 0 par défaut

In [4]:
```
L=[]
for i in range(0,5,1):
    L.append(i)
print(L)
L=[]
for i in range(5):#itération sur les entiers [0,5[
    L.append(i)
print(L)
```

```
[0, 1, 2, 3, 4]
[0, 1, 2, 3, 4]
```

b) Les boucles condiotionnelles "while" :

La boucle while est utilisée pour répéter une séquence d'instruction tant qu'une condition est vérifiée. Il est donc essentiel d'atteindre cette condition pour stopper la boucle

In [3]:
```
somme=0#initialisation
L=[0,1,2,3,4,5]
i=0#initialisation
while i<len(L):
    somme=somme+L[i]
    i=i+1#essentiel pour parcourir la liste et stopper la boucle
print(somme)

```

```
15
```

Rq : ici i est un compteur initialisé à 0, qui permet par exemple de connaître le nombre de passages dans la boucle while

On peut donc générer des listes :

- Par extension (amnuellement) L=[0,1,2,3,4]
- Par conversion L=list(range(5))
- par compréhension : liste=[i**2 for i in range(5) if i%2==0]
- Par ajout successif avec une boucle for et la méthode append

In [ ]:
```

```