TD1 : Parcours des listes

Exercice 1 :

Prévoire la liste L retournée à la fin de ce programme ?

In [4]:
```
L=[]
for i in range(5):
    L.append(i)
L=L[:-1]
L=L+L[::-1]#L=L+L[len(L)-1::-1]   possible aussi
```

In [5]:
```
print(L)
```

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

Exercice 2 :

Prévoire la liste L retournée à la fin de ce programme ?

In [6]:
```
L=[]
for i in range(5,-1,-2):
    L.append(i)
```

In [7]:
```
print(L)
```

```
[5, 3, 1]
```

Exercice 3 :

Ecrire une fonction *longueur* qui prend en paramètre une chaîne de caractères ou une liste et renvoie la longueur de cette chaîne ou de cette liste. Il est interdit d'utiliser la fonction len.

In [9]:
```
def longueur(liste):
    compteur=0
    for i in liste:
        compteur=compteur+1
    return compteur
chaine="abcde"
longueur(chaine)
```

Out[9]: 5

Exercice 4 :

Ecrire une fonction *parité* qui prend en argument une liste d'entiers et renvoie un couple de deux listes : la première contient les nombres pairs et la seconde les nombres impairs.

In [10]:
```
def parité(L):
    L_pair=[]
    L_impair=[]
    for i in L :
        if i%2==0:
            L_pair.append(i)
        else :
            L_impair.append(i)
    return L_pair,L_impair
parité([0,1,2,3,4,5,6,7,8,9,10])
```

Out[10]:
```
([0, 2, 4, 6, 8, 10], [1, 3, 5, 7, 9])
```

In [12]:
```
def parite2(L):
    Lp=[i for i in L if i%2==0]
    Li=[i for i in L if i%2!=0]
    return Lp,Li
parite2([0,1,2,3,4,5,6,7,8,9,10])
```

Out[12]:
```
([0, 2, 4, 6, 8, 10], [1, 3, 5, 7, 9])
```

Exercice 5 :

Ecrire une fonction *produit* qui prend en paramètres un entier naturel n et une liste de nombre nombres L et renvoie une nouvelle liste L_{new} obtenue en multipliant chaque élément de la liste L nombre par n.

In [13]:
```
def produit(L,n):
    L_new=[]
    for i in L :
        L_new.append(n*i)
    return L_new
L=[0,1,2,3,4,5,6,7,8,9]
n=2
produit(L,n)
```

Out[13]:
```
[0, 2, 4, 6, 8, 10, 12, 14, 16, 18]
```

In [ ]:
```
def produit2(L,n):
    L_new=[]
    for i in range(len(L)):
        L_new.append(n*L[i])
    return L_new
def produit3(L,n):
    return [n*i for i in L]

```

Exercice 6 :

Ecrire une fonction $distance$ qui prend en argument une liste de nombres appelée $nombres$ et qui renvoie la somme de la valeur absolue des écarts de chaque élément de nombres avec la moyenne de nombres . En python, la fonction abs renvoie la valeur absolue du nombre donné en paramètre. La fonction $mean$ n'est pas autorisée.

In [18]:
```
def distance(nombres):
    somme=0
    for i in nombres:
        somme=somme+i
    moyenne=somme/len(nombres)
    somme_ecart=0
    for i in nombres:
        somme_ecart=somme_ecart+abs(i-moyenne)
    return somme_ecart
nombres=[0,1,2,3,4,5,6,7,8,9]
distance(nombres)
```

Out[18]: 25.0

Exercice 7 :

On considère une liste L d'entiers.

- Ecrire une fonction qui prend L en argument et un entier elt et qui renvoie $True$ si elt est dans L et $False$ dans le cas contraire (en utilisant une boucle while)
- Ecrire une fonction qui prend L en argument et un entier elt et qui renvoie $True$ si elt est dans L et $False$ dans le cas contraire (en utilisant une boucle for)

In [26]:
```python
def f1(L,elt):
    i=len(L)-1
    while i>=0:
        if L[i]==elt:
            return True
        else :
            i=i-1
    return False

def f1_bis(L,elt):
    i=0
    while i<len(L):
        if L[i]==elt:
            return True
        else :
            i=i+1
    return False

def f2(L,elt):
    for i in L:
        if i==elt:
            return True
    return False
L=[0,1,2,3,4,5,6,7,8,9]
elt=9
f1_bis(L,elt)
```

Out[26]: True

In [1]:
```python
def f3(L,elt):
    return elt in L
L=[0,1,2,3,4,5,6,7,8,9]
elt=9
f3(L,elt)
```

Out[1]: True

Exercice 8 :

On considère une liste L d'entiers.

- Ecrire une fonction qui prend L en argument et un entier elt et qui renvoie l'indice de l'élément recherché si elt est dans L et $False$ dans le cas contraire (en utilisant une boucle while).
- Ecrire une fonction qui prend L en argument et un entier elt et qui renvoie l'indice de l'élément recherché si elt est dans L et $False$ dans le cas contraire (en utilisant une boucle for).

In [2]:
```python
def f1(L,elt):
    i=len(L)-1
    while i>=0:
        if L[i]==elt:
            return i
        else :
            i=i-1
    return False

def f1_bis(L,elt):
    i=0
    while i<len(L):
        if L[i]==elt:
            return i
        else :
            i=i+1
    return False

def f2(L,elt):
    for i in range(len(L)):
        if L[i]==elt:
            return i
    return False
L=[0,1,2,3,4,5,6,7,8,9]
elt=9
print(f1(L,elt))
print(f1_bis(L,elt))
print(f2(L,elt))
```

```
9
9
9
```

Exercice 9 :

On considère une liste L d'entiers. Chercher l'indice de la dernière occurrence de l'élément elt dans L.

In [3]:
```python
def occurence1(L,elt):
    i=len(L)-1
    while i>=0:
        if L[i]==elt:
            return i
        else :
            i=i-1
    return False

def occurence2(L,elt):
    for i in range(len(L)-1,-1,-1):
        if L[i]==elt:
            return i
    return False
def occurence3(L,elt):
    resultat=None
    for i in range(len(L)):
        if L[i]==elt:
            resultat=i
    if resultat!=None:
        return resultat
    else :
        return False
L=[0,1,2,3,4,5,6,7,8,9]
elt=9
print(occurence1(L,elt))
print(occurence2(L,elt))
print(occurence3(L,elt))
```

```
9
9
9
```

Exercice 10 :

On souhaite trouver le maximum d'une liste L. Si la liste est non vide, on suppose alors que le maximum est le 1e élément de la liste, puis on parcourt L et chaque fois que l'on rencontre un élément plus grand que le maximum provisoire, on dit que c'est ne nouveau maximum provisoire.

Proposer une fonction prenant une liste en argument et permettant de décrire la situation ci-dessus. Le maximum et sa position seront renvoyés.

In [9]:
```python
def maximum(L):
    indice=0
    maxi=L[0]
    for i in range(1,len(L)):
        if L[i]>maxi:
            maxi=L[i]
            indice=i
    print("le maximum est {0} et est à la position {1}".format(maxi

L=[0,1,2,3,4,5,6,7,8,9]
maximum(L)
```

```
le maximum est 9 et est à la position 9
```

Exercice 11 :

Ecrire une fonction qui prend en paramètre une liste de nombres et renvoie le maximum de ces nombres avec son indice dans la liste. Si plusieurs éléments sont égaux au maximum, la fonction renvoie l'indice le plus grand parmi les indices de ces éléments.

In [3]:
```
def maximum2(L):
    indice=0
    maxi=L[0]
    for i in range(1,len(L)):
        if L[i]>=maxi:
            maxi=L[i]
            indice=i
    print("le maximum est {0} et est à la position {1}".format(maxi
L=[0,1,2,3,4,5,6,7,8,9,9]
maximum2(L)
```

```
le maximum est 9 et est à la position 10
```

Exercice 12 :

Ecrire une fonction qui prend en paramètre une liste de nombres non vide et renvoie le maximum de ces nombres avec la liste des indices de tous les éléments égaux au maximum.

In [12]:
```
def maximum3(L):
    indice=0
    maxi=L[0]
    for i in range(1,len(L)):
        if L[i]>maxi :
            maxi=L[i]
            indice=i
    L_indice=[]
    L_indice.append(indice)
    for i in range(indice+1,len(L)):
        if L[i]==maxi:
            L_indice.append(i)
    print("le maximum est {0} et est à la position {1}".format(maxi
L=[0,1,2,3,4,5,6,7,8,9,9]
maximum3(L)
```

```
le maximum est 9 et est à la position [9, 10]
```

Exercice 13 :

On cherche maintenant à trouver les deux plus grands éléments d'une liste L de longueur $n > 2$ en appliquant la méthode suivante :

- les deux premiers éléments de L constituent le couple recherché,
- puis on parcourt la liste pour remplacer éventuellement l'un de ces deux éléments.

Proposer un programme permettant de retourner l'index de ces deux maximas ainsi que leur valeur respective.

In [19]:
```
def maxi_double(L):
    max1,max2=L[0],L[1]
    indice1,indice2=0,1
    if max1<max2 :
        max1,max2=max2,max1
        indice1,indice2=indice2,indice1
    for i in range(2,len(L)):
        if L[i]>max1:
            max2,indice2=max1,indice1
            max1,indice1=L[i],i
        elif L[i]>max2:
            max2,indice2=L[i],i
    print("le max1 est {0} et est à la position {1}, \
le max2 est {2} et est à la position {3}".format(max1,indice1,max2,
L=[0,1,2,3,4,5,6,7,8,9]
maxi_double(L)
```

```
le max1 est 9 et est à la position 9, le max2 est 8 et est à la posi
tion 8
```

Exercice 14 :

On donne la fonction mystère suivante :

In [19]:
```
def mystere(liste_1,liste_2):
    liste=[]
    i,j=0,0
    while i<len(liste_1) and j<len(liste_2):
        if liste_1[i]<liste_2[j]:
            liste.append(liste_1[i])
            i=i+1
        else :
            liste.append(liste_2[j])
            j=j+1
    return liste

```

On appelle cette fonction avec l'instruction mystere([2,4,6,8],[1,3,5,7,9])

Quel est le résultat envoyé ?

Cette fonction permet d'obtenir une liste triée à partir des éléments de deux listes elles-mêmes déjà triées (à un élément près...)

In [22]:
```
mystere([2,4,6,8],[1,3,5,7,9])
```

Out[22]:
```
[1, 2, 3, 4, 5, 6, 7, 8]
```

In [2]:
```
#on pourra remarquer qu'il manque le dernier élément
def mystere(liste_1,liste_2):
    liste=[]
    i,j=0,0
    while i<len(liste_1) and j<len(liste_2):
        if liste_1[i]<liste_2[j]:
            liste.append(liste_1[i])
            i=i+1
        else :
            liste.append(liste_2[j])
            j=j+1
    if i==len(liste_1):
        return liste+liste_2[j:]
    else :
        return liste+liste_1[i:]
mystere([2,4,6,8],[1,3,5,7,9])
```

Out[2]:
```
[1, 2, 3, 4, 5, 6, 7, 8, 9]
```

In [ ]:
```

```