TD1_corrige

August 26, 2026

1 TD1 : Parcours des listes

1.1 Exercice 1 :

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

[1]:
```
L=[]
for i in range(5):
    L.append(i)
L=L[:-1]
L=L+L[::-1]
```

[2]:
```
print(L)
```

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

[3]:
```
L=[]
for i in range(5):
    L.append(i)
L=L[:-1]
L=L+L[-1:-len(L)-1:-1]#autre possibilité
print(L)
```

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

[4]:
```
L=[]
for i in range(5):
    L.append(i)
L=L[:-1]
L=L+L[len(L)-1::-1]#autre possibilité !!!ne pas mettre -1 comme indice d'arrivée!
print(L)
```

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

1.2 Exercice 2 :

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

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

```
    L.append(i)
```

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

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

1.3 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.

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

[9]:
```
5
```

[1]:
```
def longueur(L):
    compteur = 0
    for _ in L:# notation _ qui implique que l'on ne se serre pas de la valeur␣
 ↪de l'élément
        compteur += 1
    return compteur
```

1.4 Exercice 4 :

Ecrire une fonction *parit* 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.

[3]:
```
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])
```

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

[6]:
```
def parite2(L):
    Lp=[i for i in L if i%2==0]
    Li=[i for i in L if i%2]
```

```
    return Lp,Li
parite2([0,1,2,3,4,5,6,7,8,9,10])
```

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

[5]:
```
def parité3(L):
    L_pair=[]
    L_impair=[]
    for i in L :
        (L_pair if i%2==0 else L_impair).append(i)
    return L_pair,L_impair
parité3([0,1,2,3,4,5,6,7,8,9,10])
```

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

1.5 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.

[2]:
```
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)
```

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

[3]:
```
def produit2(L,n):
    L_new=[]
    for i in range(len(L)):
        L_new.append(n*L[i])
    return L_new
produit2(L,2)
```

[3]: [0, 2, 4, 6, 8, 10, 12, 14, 16, 18]

[4]:
```
def produit3(L,n):
    i=0
    L_new=[]
    while i<len(L):
        L_new.append(L[i]*n)
        i=i+1
    return L
```

```
produit3(L,2)
```

[4]:
```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

[5]:
```
L=[0,1,2,3,4,5,6,7,8,9]
def produit4(L,n):
    return [n*i for i in L]
produit4(L,n=2)
```

[5]:
```
[0, 2, 4, 6, 8, 10, 12, 14, 16, 18]
```

1.6 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.

[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)
```

[18]:
```
25.0
```

[9]:
```
def distance2(nombres):
    moy=0
    i=0
    while i<len(L):
        moy=moy+L[i]
        i=i+1
    moy=moy/len(L)
    s=0
    i=0
    while i<len(L):
        s=s+abs(L[i]-moy)
        i=i+1
    return s
nombres=[0,1,2,3,4,5,6,7,8,9]
distance2(nombres)
```

[9]: 25.0

1.7 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)

[1]:
```
def f1(L,elt):#meilleure des méthodes
    return elt in L
def f2(L,elt):
    for i in L:
        if i==elt:
            return True
    return False
def f3(L,elt):#moins bien de travailler avec les indices (si on peut éviter)
    for i in range(len(L)):
        if L[i]==elt:
            return True
    return False
def f4(L, elt):
    i=0
    while i<len(L):
        if L[i]==elt:
            return True
        i=i+1
    return False
```

1.8 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).

[4]:
```
def f1(L,elt):
    for i in range(len(L)):
        if L[i]==elt:
            return i
    return False
def f2(L,elt):
    i=0
    while i<len(L):
        if L[i]==elt:
            return i
        i=i+1
```

```
    return False
def f3(L,elt):
    for i,val in enumerate(L):
        if val==elt:
            return i
    return False
def f4(L,elt):
    return L.index(elt) if elt in L else False
f4([1,2,3,4,5],5)
```

[4]: 4

[13]:
```
#on peut faire une liste des positions :
def liste(L,x):
    return [i for i in range(len(L)) if L[i]==x]
liste([0,1,2,3,4,5,6,78,90,1,2,3,4,5,6,78,9,78],78)
```

[13]: [7, 15, 17]

1.9 Exercice 9 :

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

[11]:
```
def occurence1(L,elt):#on prend la liste à l'envers
    i=len(L)-1
    while i>=0:
        if L[i]==elt:
            return i
        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=False
    for i in range(len(L)):
        if L[i]==elt:
            resultat=i
    return resultat
def occurence4(L,elt):
    resultat=False
    i=0
    while i<len(L):
        if L[i]==elt:
```

```
            resultat=i
        i=i+1
    return resultat
def occurence5(L,elt):
    liste=[]
    for i in range(len(L)):
        if L[i]==elt:
            liste.append(i)
    if len(liste)==0:
        return False
    else :
        return liste[-1]

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))
print(occurence4(L,elt))
print(occurence5(L,elt))
```

```
9
9
9
9
9
```

1.10 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.

[ ]:
```
def maximum(L):
    if not L:
        return None
    maxi,indice=L[0],0
    for i in range(1,len(L)):
        if L[i]>maxi:
            maxi,indice=L[i],i
    return maxi,indice
```

1.11 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.

[ ]:
```
def maximum1(L):
    if not L:
        return None
    maxi,indice=L[0],0
    for i in range(1,len(L)):
        if L[i]>=maxi:
            maxi,indice=L[i],i
    return maxi,indice
```

[9]:
```
def maximum2(L):
    if not L:
        return None
    maxi,indice=L[0],0
    for i,val in enumerate(L[1:],start=1):
        if val>=maxi:
            maxi,indice=val,i
    return maxi,indice
L=[0,1,2,3,4,5,6,7,8,9]
maximum2(L)
```

[9]:
```
(9, 9)
```

1.12 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.

[11]:
```
def liste_max(L):
    if not L:
        return None
    maxi=L[0]
    for i,val in enumerate(L[1:],start=1):
        if val>=maxi:
            maxi=val
    L_new=[]
    for i in range(len(L)):
        if L[i]==maxi:
            L_new.append(i)
    return maxi,L_new
```

[15]:
```
def liste_max(L):
    maxi,indice=L[0],0
    for i in range(1,len(L)):
        if L[i]>maxi:
            maxi,indice=L[i],i
    liste=[]
```

```
    for i in range(len(L)):
        if L[i]==maxi:
            liste.append(i)
    return maxi,liste
L=[0,1,2,3,4,5,6,7,8,9,9]
liste_max(L)
```

[15]: (9, [9, 10])

[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_indice))
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]
```

[2]:
```
def maximum4(L):#avec une seule boucle si on utilise une liste des indices dès␣
 ↪le départ qui est renouvellé à chaque nouveau max
    if not L:
        return None
    maxi = L[0]
    indices = [0]
    for i in range(len(L)):
        if L[i]>maxi:
            maxi,indices=L[i],[i]
        elif L[i]==maxi:
            indices.append(i)
    return maxi,indices
L=[0,3,3,3,3,3,9,9]
maximum4(L)
```

[2]: (9, [6, 7])

1.13 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.

[ ]:
```
def maxi_double(L):
    max1,max2=L[0],L[1]
    i1,i2=0,1
    if max1<max2:
        max1,max2=max2,max1
        i1,i2=i2,i1
    for i in range(2,len(L)):
        if L[i]>max1:
            max2,i2=max1,i1
            max1,i1=L[i],i
        elif L[i]>max2:
            max2,i2=L[i],i
    return max1,i1,max2,i2
```

[15]:
```
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(f"le max1 est {max1} et est à la position {indice1},le max2 est {max2}␣
 ↪et est à la position {indice2}")
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 position 8
```

[ ]:
```
def maxi_double(L):
    """Retourne ((max1, i1), (max2, i2)) avec max1 >= max2.
    En cas d'égalité, conserve les premiers indices rencontrés.
    """
    n = len(L)
    if n < 2:
        raise ValueError("La liste doit contenir au moins deux éléments.")

    # init avec les deux premiers
    max1, i1 = L[0], 0
```

```
    max2, i2 = L[1], 1
    if max2 > max1:
        max1, i1, max2, i2 = max2, i2, max1, i1  # garantit max1 >= max2

    # une seule passe
    for i, val in enumerate(L[2:], start=2):
        if val > max1:
            max2, i2 = max1, i1
            max1, i1 = val, i
        elif val > max2:
            max2, i2 = val, i

    return (max1, i1), (max2, i2)
```