TP 03

Structure séquentielle par boucles imbriquées

Savoirs et compétences :

- ❒ *Th. 2 : Algorithmes opérant sur une structure séquentielle par boucles imbriquées.*

Proposition de corrigé

Activité 1 –

Question 1 *Écrire une fonction* `Nb_Ventes(ventes:dict)` *qui prend en entrée un dictionnaire et renvoie le nombre total de ventes dans la société.*

```
def Nb_Ventes(ventes):
    """Renvoie le nombre total de ventes dans la societe"""
    S=0
    for elt in ventes.items():
        S+=elt[1]
    return S
```

Question 2 *Écrire une fonction* `Nom_vendeur(ventes:dict)` *qui prend en entrée un dictionnaire et renvoie le nom du vendeur ayant réalisé le plus de ventes. Si plusieurs vendeurs sont ex-aequo, la fonction devra retourner le nom de l'un d'entre eux seulement.*

```
def Nom_vendeur(ventes):
    """Renvoie le nombre total de ventes dans la societe"""
    max=0
    nom=''
    for elt in ventes.items():
        if elt[1]>max:
            max=elt[1]
            nom=elt[0]
    return nom
```

Activité 2 – Recherche dans un tableau

Question 3 *On appelle distance minimale, la distance entre deux éléments les plus proches (éventuellement égaux) et d'indices distincts. Écrire une fonction* `distance_min(L)` *qui renvoie la distance minimale de* `L`.

```
def distance_min(L): # On cherche min |Ti-tj| pour i<j
    n=len(L)
    min=abs(L[1]-L[0])
    for i in range(n):
        for j in range(i+1,n):
            if abs(L[j]-L[i])<min:
                min=abs(L[j]-L[i])
    return(min)
```

Question 4 *Écrire une fonction* `indices_distance_min2(L)` *qui renvoie un couple d'indices réalisant la distance minimale*

```
def indices_distance_min2(L): # On cherche min |Ti-tj| pour i<j
    n=len(L)
    min=abs(L[1]-L[0])
    p,q=0,1
    for i in range(n):
        for j in range(i+1,n):
            if abs(L[j]-L[i])<min:
                min=abs(L[j]-L[i])
                p,q=i,j
    return p,q
```

Question 5 *Écrire une fonction* `indices_distance_min3(L)` *qui, étant donnée une liste* `L`*, réalise ces opérations et renvoie un couple solution*

```
 def indices_distance_min3(L):
    D={}
    n=len(L)
    for i in range(n-1):
        m=abs(L[i+1]-L[i])
        j=i+1
        for k in range(i+1,n):
            if abs(L[k]-L[i])<m:
                m=abs(L[k]-L[i])
                j=k
        D[str(i)]=[j,m]
    p,q=0,D["0"][0]
    min=D["0"][1]
    for elt in D.items():
    # elt de la forme ["3",[4,min {|Lk-L3|, k>4}]
        if elt[1][1]<min:
            p,q=int(elt[0]),elt[1][0]
            min=elt[1][1]
    return(p,q)
```

Activité 3 – Recherche d'un mot dans un texte

Question 6 *Écrire une fonction* `est_ici(texte,motif,i)` *qui, étant données deux chaines de caractères* `texte`, `motif` *et un indice* `i`*, renvoie* `True` *ou* `False` *selon que* `motif` *est ou n'est pas dans* `texte` *au rang* `i`*. On utilisera une boucle* `while`*.*

```
def est_ici(texte, motif, i):
    p=len(motif)
    j=0
    while j<=p-1 and motif[j]==texte[i+j]:
            j=j+1
    return(j==p)
```

Question 7 *Écrire une fonction* `est_sous_mot(texte,motif)` *qui renvoie* `True` *ou* `False` *selon que* `motif` *est dans* `texte` *ou pas. On utilisera une boucle* `while`*.*

```
def est_sous_mot(texte, motif):
    n,p=len(texte), len(motif)
    i=0
    while i<=n-p and not est_ici(texte, motif, i):
        i=i+1
    return(i<=n-p)
```

Question 8 *Écrire une fonction* `position_sous_mot(texte,motif)` *qui renvoie la liste de toutes les occurences de l'indice de position de la première lettre du mot* `motif` *dans* `texte`*. On utilisera une boucle* `for`*.*

```
def position_sous_mot(texte, motif):
    n,p=len(texte), len(motif)
    L=[]
    for i in range(n-p+1):
        if est_ici(texte, motif, i):
            L.append(i)
    return(L)
```

Activité 4 – Tri à bulles

Question 9 *Appliquer l'algorithme de tri à Bulles «à la main» au tableau ci-dessous :*

2	1	6	9	8	4

Question 10 *Ecrire une fonction* `est_trie(T)` *qui renvoie* `True` *ou* `False` *selon que le tableau* `T` *est trié ou pas.*

```
def est_trie(T):
    """Teste si T est trié ou pas"""
    test=True
    n=len(T)
    i=0
    while i<=n-2 and test:
        test=(T[i]<=T[i+1])
        i=i+1
    return(test)
```

Question 11 *Écrire une fonction* `TriBulles(T)` *triant le tableau* `T` *par l'algorithme de tri à bulles*

```
def TriBulles(T):
    """tri du tableau T avec le tri àbulles"""
    n=len(T)
    for i in range(1,n): # numéro du parcours
        for k in range(n-i):
            if T[k]>T[k+1]:
                T[k],T[k+1]=T[k+1],T[k]
        print(T)
    return(T)
```

Question 12 *Vérifier que la fonction ci-dessus est correcte en utilisant la fonction* `EstTrie` *sur un tel tableau de nombres aléatoires.*

Question 13 *Pour un tableau T de longueur n, calculer le nombre de comparaisons $C(n)$ faites dans la fonction* `TriBulles(T)` *et vérifier que la suite* $\left(\frac{C(n)}{n^2}\right)$ *a une limite finie donc est bornée (on dit alors que la* ***complexité est quadratique****);*

```
'''
La première boucle k exécute n-1 opérations. Pour chaque k, il y a n-k boucles i et une ↙
    comparaison pour chacune. Au total, il y a une complexité de Sum_{k=1...n-1\}.Sum_{i=0..n-↙
    k} 1 = Sum_{k=1...n-1} (n-k) = Sum_{k=1...n-1} k = n(n-1)/2 = O(n**2): on obtient une ↙
    complexité quadratique.
'''
```

Question 14 *Ecrire une fonction améliorée* `TriBulles2(T)` *qui sort de la fonction dès que le tableau a été parcouru sans faire d'échange (auquel cas, il est trié et donc inutile de continuer).*

```
def TriBulles2(T):
    """tri du tableau T avec le tri àbulles"""
    n=len(T)
```

```
    i=1
    echange=True
    while i<=n-1 and echange:
        echange=False
        for k in range(n-i):
            if T[k]>T[k+1]:
                T[k],T[k+1]=T[k+1],T[k]
                echange=True
        i+=1
        print(T)
    return(T)
```

Question 15 *Écrire une fonction* `TriCocktail(T)` *qui étant donné un tableau* `T` *le renvoie trié selon la méthode du tri cocktail.*

```
def TriCocktail(T):
    n=len(T)
    for k in range(1,(n-1)//2):
        for i in range(k-1,n-k): # Parcours des cases k-1 àn-k-1
            if T[i]>T[i+1]:
                T[i],T[i+1]=T[i+1],T[i]
        print(T)
        for i in range(n-k-1,k-2,-1): # Parcours des cases n-k-1 àk-1
            if T[i]>T[i+1]:
                T[i],T[i+1]=T[i+1],T[i]
        print(T)
    return(T)
```

Question 16 *Justifier en quoi cette variante peut être intéressante selon le type de tableau à trier mais vérifier néanmoins que la complexité reste quadratique.*

```
'''
Cette fonction est àprivilégier lorsque le tableau contient des éléments minima en fin de ↙
    tableau, notamment lorsque le tableau est décroissant par exemple.
# Néanmoins, dans le pire des cas, le nombre de comparaison est la somme sur k variant de 1 à(↙
    n-1)//2 de (n-k-1)-(k-1)+1+(n-k-1)-(k-1)+1=2(n-2k+1). Cela donne encore une complexité ↙
    quadratique car équivalent àun terme de la forme a.n**2.
```