

Cette évaluation est composée d'exercices indépendants. Les réponses sont à donner sur feuille de copie :

1- MINIMUM ET MAXIMUM :

- 1- Écrire le code python d'une fonction *minMax()* qui prend en paramètre une liste de nombres ℓ **non vide**, et qui renvoie la plus petite et la plus grande valeur de ℓ ainsi que leur index, sous la forme d'un dictionnaire à quatre clés min, max, iMin et iMax.

Par exemple, l'exécution de cette fonction avec la liste : $\ell = [0, 1, 4, 2, -2, 9, 3, 1, 7, 1]$

donne dans la console :

```
>>> minMax(l)
{'min': -2, 'max': 9, 'iMin': 4, 'iMax': 5}
```

```
def minMax(l) :
    iMin = 0
    iMax = 0
    for i in range(1, len(l)) :
        if l[i] < l[iMin] : iMin = i
        if l[i] > l[iMax] : iMax = i
    dic = {}
    dic['min'] = l[iMin]
    dic['max'] = l[iMax]
    dic['iMin'] = iMin
    dic['iMax'] = iMax
    return dic
```

CORRIGE

- 2- Quelle est la classe de complexité de ce script ? Justifier.

On réalise dans ce script un parcours simple de la liste. Le nombre d'opérations élémentaires est donc proportionnel à la taille n de la liste. La complexité est ainsi de classe $\mathcal{O}(n)$.

2- VERIFICATION :

- 1- Écrire le code python d'une fonction *verification()* qui prend en paramètre une liste de nombres ℓ , et qui renvoie *True* si la liste est triée, *False* dans le cas contraire.

On donne ci-dessous 4 exemples différents d'exécution de cette fonction dans la console :

```
>>> verification([1,8,8,10,12])
True

>>> verification([1,8,-7,10,12])
False
```

```
>>> verification([1])
True

>>> verification([])
True
```

```
def verification(l):
    n = len(l)
    if n < 2 : return True
    retour = True
    for i in range(n-1) :
        if l[i] > l[i+1] : retour = False
    return retour
```

CORRIGE

2- Quelle est la classe de complexité de ce script ? Justifier.

On réalise dans ce script un parcours simple de la liste. Le nombre d'opérations élémentaires est donc proportionnel à la taille n de la liste. La complexité est ainsi de classe $\mathcal{O}(n)$.

3- TRI PAR SELECTION :

Vous disposez d'une fonction *echange()* dont le code est donné ci-contre.

```
def echange(liste , i , j) :
    n = len(liste)-1
    if i < 0 or i > n : return False
    if j < 0 or j > n : return False
    tmp = liste[i]
    liste[i] = liste[j]
    liste[j] = tmp
    return True
```

1- Ecrire sur feuille de copie, le code python d'une fonction *tri()* qui prend en paramètre une liste ℓ et renvoie cette liste triée en utilisant le principe du tri par sélection. L'exécution de cette fonction donne par exemple :

```
>>> tri([0, 1, 4, 2, -2, 9, 3, 1, 7, 1])
[-2, 0, 1, 1, 1, 2, 3, 4, 7, 9]
```

```
def tri(liste) :
    n = len(liste)
    for i in range(n-1) :
        jMin = i
        for j in range(i+1,n) :
            if liste[j] < liste[jMin]:
                jMin = j
        echange(liste,i,jMin)
    return liste
```

CORRIGE

3- Quelle est la classe de complexité de ce script ? Justifier.

On réalise dans ce script un parcours de la liste dans lequel à chaque itération, on refait un parcours sur les éléments situés à droite. Le nombre d'opérations élémentaires est donc proportionnel à n^2 , n étant la taille de la liste. La complexité est ainsi de classe $\mathcal{O}(n^2)$.

2- On donne la liste $\ell = [0, 1, 4, 2, -2, 9, 3, 1, 7, 1]$. Lorsque cette liste est triée en utilisant l'algorithme de tri par sélection que vous avez écrit, 9 échanges sont réalisés. Ecrire sur feuille de copie, l'état de cette liste après chaque échange.

$l = [0, 1, 4, 2, -2, 9, 3, 1, 7, 1]$

```
[ -2, 1, 4, 2, 0, 9, 3, 1, 7, 1]
[ -2, 0, 4, 2, 1, 9, 3, 1, 7, 1]
[ -2, 0, 1, 2, 4, 9, 3, 1, 7, 1]
[ -2, 0, 1, 1, 4, 9, 3, 2, 7, 1]
[ -2, 0, 1, 1, 1, 9, 3, 2, 7, 4]
[ -2, 0, 1, 1, 1, 2, 3, 9, 7, 4]
[ -2, 0, 1, 1, 1, 2, 3, 9, 7, 4]
[ -2, 0, 1, 1, 1, 2, 3, 4, 7, 9]
[ -2, 0, 1, 1, 1, 2, 3, 4, 7, 9]
```

4- LIGNES DE COMMANDE :

L'affichage sur Windows du contenu d'un dossier créé sur la racine du disque dur, donne :

nsi	27/09/2024 18:15	Dossier de fichiers	
test	27/09/2024 18:16	Dossier de fichiers	
fichier.txt	27/09/2024 18:15	Document texte	1 Ko
image.png	27/09/2024 18:15	Fichier PNG	8 Ko
lorem.txt	27/09/2024 18:15	Document texte	6 Ko

On utilise à présent un terminal de commande. En exécutant la commande `pwd`, on obtient :

```
$ pwd
/c/Users/eval
```

En exécutant la commande `ls`, on obtient :

```
$ ls
fichier.txt image.png lorem.txt nsi/ test/
```

1- Quelle commande doit-on exécuter pour obtenir :

```
./ ../ fichier.txt image.png lorem.txt nsi/ test/
```

`ls -a`

2- Dans cet affichage, que représente `../` et `./` ?

`../` représente le dossier parent et `./`, le dossier courant

3- On exécute la commande `$ mkdir vacances`. Que renvoie ensuite l'exécution de la commande `ls` ?

`fichier.txt image.png lorem.txt nsi/ test/ vacances/`

4- On exécute les commandes `$ mv image.png vacances` et `$ cd vacances`. Que renvoie ensuite l'exécution de la commande `ls` ?

`image.png`

5- On exécute les commandes `$ cd ..` et `$ cp -r vacances autres`. Que renvoie ensuite l'exécution de la commande `ls` ?

`/autres fichier.txt image.png lorem.txt nsi/ test/ vacances/`

6- Quelle commande doit-on exécuter pour afficher le contenu de `fichier.txt` dont le contenu est affiché ci-dessous ?

```
Ceci est un fichier.  
Le terminal, c'est facile !
```

`cat fichier.txt`

7- On exécute `$ grep "Le" fichier.txt`. Qu'obtient-on dans le terminal ?

`Le terminal, c'est facile !`

8- On exécute `$ ls -a > liste.txt`. Que se passe-t-il ?

Le fichier `liste.txt` est créé avec comme contenu

`./ ./ /autres fichier.txt image.png lorem.txt nsi/ test/ vacances/`