Université Moulay Ismail
Faculté des sciences de Meknès
Département d'Informatique

Année universitaire :2023/2024
Filière : MIP/Digitalisation
Module : INFO2

TD N°4 : Les chaînes de caractères, les fichiers et la complexité algorithmiques

(Corrigé)

Exercice 1 :

Ecrire un programme en Python qui renvoie la longueur d'une liste donnée sans utiliser la méthode len().

Exercice 1 (corrigé) :

```python
# coding: utf-8
def lenChaine(Ch):
    # initialisation de la longueur de la chaine
    l = 0
    # parcour les éléments de la chaine
    for x in Ch:
        # incrémentation de la longueur
        l = l + 1
    return l

# Exemple
Chaine = "Bonjour tout le monde..."
print("la longueur de la chaine est : " , lenChaine(Chaine))
```

Exercice 2 :

Ecrire un programme en Python, qui demande à l'utilisateur de saisir un texte et qui détermine la liste des mots commençant par une majuscule.
Exemple. Si texte="Python is more power than Java"
Le programme renvoi l'ensemble : ["Python", "Java"]

Exercice 2 (corrigé) :

```python
texte=input("Saisir un texte")
newList=[]
#divise une chaîne de caractères en sous-chaînes,
res=texte.split()
for w in res :
    #isupper() permet de tester si un caractère donné est une lettre majuscule
    if(w[0].isupper()):
        #ajoute un élément à la fin d'une liste existante
        newList.append(w)

print(newList)
```

Exercice 3 :

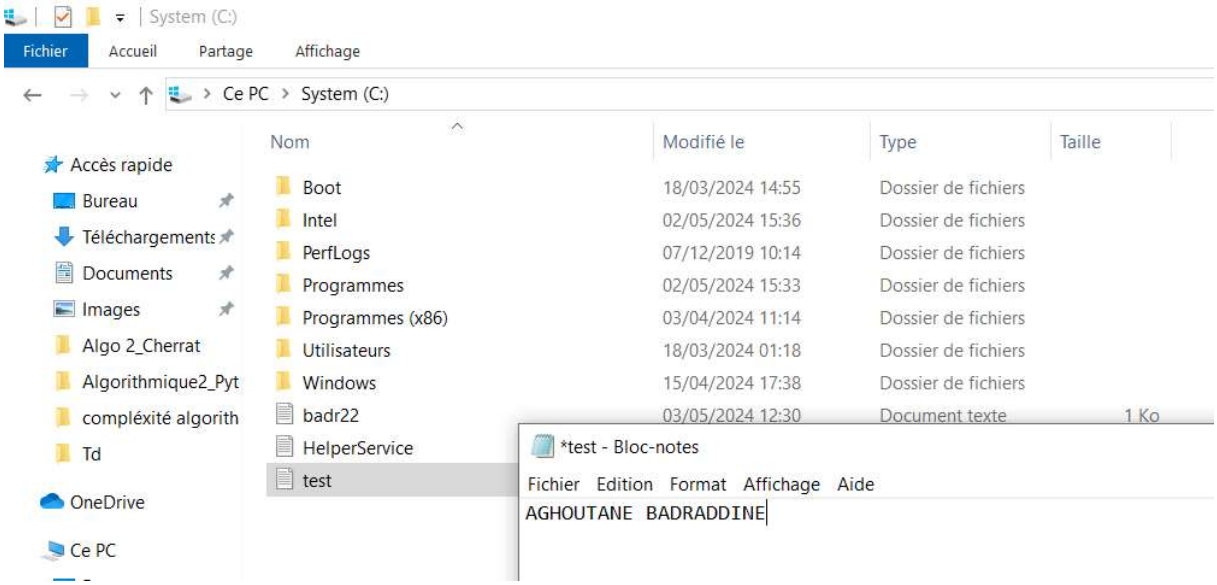
Ecrire un algorithme permettant de créer et de remplir un fichier par N caractère. Les cases sont lues à partir du clavier.
Le nom du fichier sera spécifié par l'utilisateur puis lire et afficher le contenu du fichier.

Exercice 3 (corrigé) :

Algorithme lecture_écriture
Variables
fich : Fichier caractère
carac : caractère
N : entier
Nom_reel : chaine de caractères
Début
Lire (nom_reel)
fich←Ouvrir (nom_reel, ecriture)
Lire(N)
Pour i = 1 à N faire
Lire (carac)
Ecrire (fich, carac)
FinPour
Fermer (fich)
Ouvrir (fich, lecture)
TantQue (non FDF (fich)) faire
Lire (fich, carac)
Ecrire (carac)
FinTantQue
Fermer (fich)
Fin

Programme Python :

```
7  #nom_reel="C:\test.txt"
8  nom_reel = input("Saisir le nom du fichier")
9  f=open(nom_reel,"w")
10 N=int(input("Donnez le nombre de caractère à saisir"))
11 for i in range(N):
12     print("Saisir un caractère")
13     carac=input()
14     f.write(carac)
15 f.close()
16 f=open(nom_reel, "r")
17 for i in f :
18     print(i, end="")
19 f.close()
```



Exercice 4 :

Ecrire un algorithme permettant de :

- Créer et remplir un fichier « fnotes » qui contient les notes de 30 étudiants.
- Copier les notes dans un tableau Tnote.
- Trier le tableau Tnote dans l'ordre croissant.
- Copier les notes triées du tableau vers le fichier fnotes.

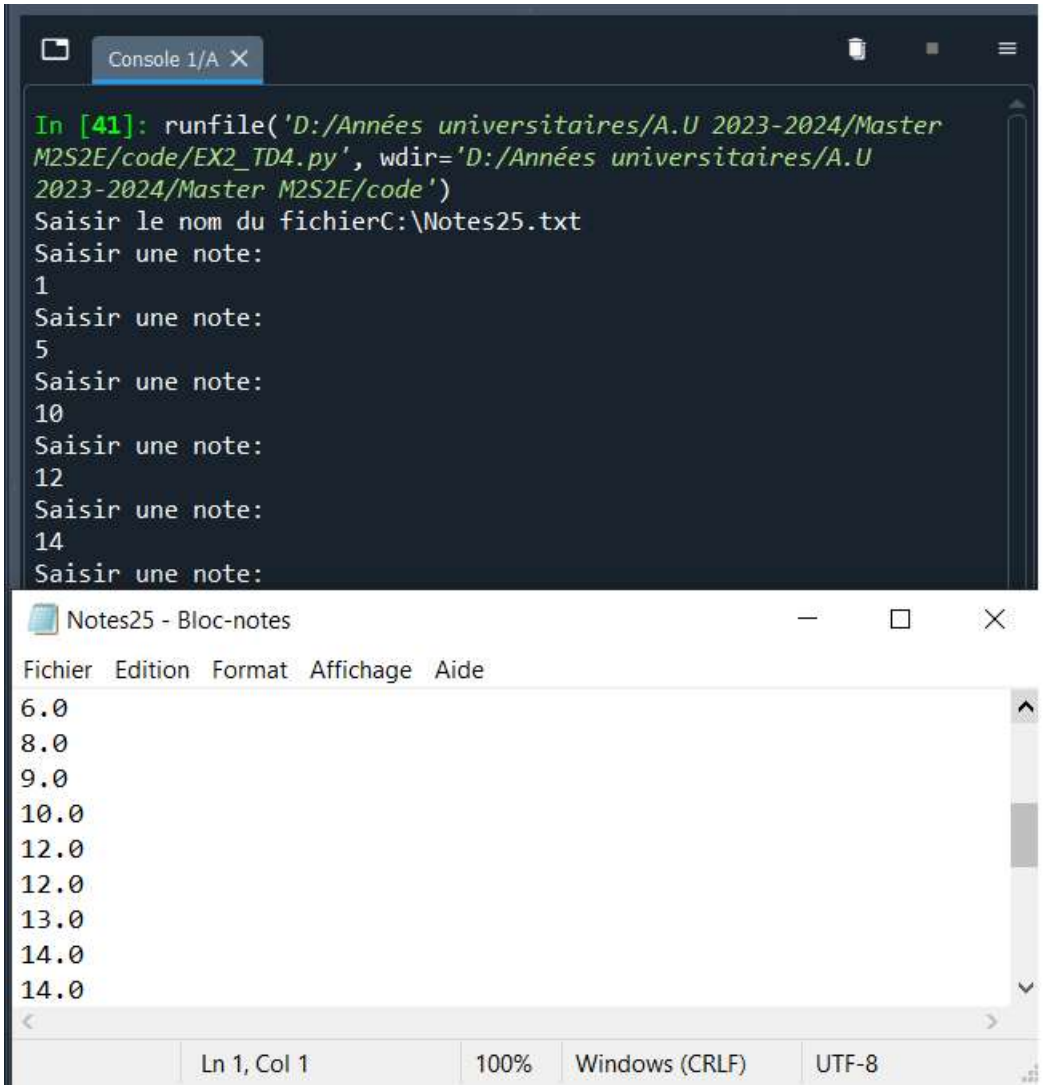
Exercice 4 (corrigé) :

```
Algorithme Notes
    Variables
        fnotes : Fichier réel
        Tableau Tnote [30] : réel
        x, note : réel
        nom_reel : chaine de caractère
        échange : booléen
        i : entier
    Début
        (* Création du fichier fnotes *)
        Lire (nom_reel)
        fnotes←Ouvrir (nom_reel, ecriture)
        Pour i =1 à 30 faire
            Ecrire("Entrer une note : ")
            Lire(note)
            Ecrire(fnotes,note)
        FinPour
        Fermer(fnotes)

        (* Copie du fichier fnotes dans le tableau Tnote *)
        Ouvrir(fnotes,lecture)
        Pour i ← 1 à 30 faire
            Lire(fnotes,note)
```

```
            Tnote[i]← note
        FinPour
        Fermer(fnotes)
        (* Tri du tableau Tnote *)
        Pour i ←0 à 28 faire
            Posmini ←i
            Pour j←i+1 à 29 faire
                Si(Tnote[i]< Tnote[posmini]) Alors
                    Posmini ←j
                Finsi
            FinPourj
            Temp← Tnote[posmini]
            Tnote[posmini]← Tnote[i]
            Tnote[i]←Temp
        FinPouri
        (* Copie du tableau Tnote dans le fichier fnotes *)
        Ouvrir(fnotes,E)
        Pour i =1 à 30
            Ecrire(fnotes,Tnote[i])
        FinPour
        Fermer(fnotes)
Fin
```

```python
# Création du fichier des notes
fnotes = input("Saisir le nom du fichier")
f=open(fnotes,"w")
for i in range(30):
    print("Saisir une note: ")
    note=input()
    f.write(note+"\n")
f.close()
# Copie du fichier fnotes dans le tableau Tnote
Tnotes=[]
f=open(fnotes, "r")
for i in range(30) :
    note=float(f.readline())
    Tnotes.append(note)
f.close()
# Tri du tableau Tnote
for i in range(0,29) :
    posmini=i
    for j in range(i+1,30) :
        if (Tnotes[j]<Tnotes[posmini]) :
            posmini=j
    temp=Tnotes[posmini]
    Tnotes[posmini]=Tnotes[i]
    Tnotes[i]=temp
# Copie du tableau Tnote dans le fichier fnotes
f=open(fnotes, "w")
for i in range(30):
    f.write(str(Tnotes[i]))
    f.write("\n")
f.close()
```



Exercice 5 :

S ←0
Pour i ←1 à n
  S ← s + i
FinPour

- Calculer la complexité de cet algorithme. **O(2 + n * 4)**

Exercice 5 (corrigé):

S ←0 [O(1)]
Pour i ←1 à n faire [O(2)]
S ← S + i [O(2)]
FinPour

$\sum_1^n O(2+2)$=O(4n)

O(1) + O(4n) = **O(1+4n)**

Exercice 6 :

Considérons le programme python suivant :

```
1.  a, b = 3, 6
2.  c = a + b
3.  print(c)
```

Calculer la complexité de ce programme python ?

Exercice 6 (corrigé) :

La ligne 1 comporte 2 affectations ;

La ligne 2 comporte 1 affectation et une opération élémentaire (l'addition) ;

La ligne 3 comporte une instruction d'affichage (aucune affectation ni opération élémentaire (on peut ne pas la compter)).

On dit alors ici que la complexité (le coût) du programme est égal à 4 (ou 5). La complexité est *constante*.

Exercice 7 :

Soit la fonction python suivante :

```
1.  def fct(n):
2.      s = 0
3.      for i in range(1,n+1):
4.          s += i
5.
6.      return s
7.
8.  print( fct(10) )
```

Exercice 7 (corrigé) :

Quelle est la complexité de la fonction *fct*(n) ?

On voit qu'il y a une première affectation (s = 0).

Ensuite, il y a n affectations pour la variable i ainsi que n opérations ($s + i$) et n autres affectations (pour s). Ainsi, au total, il y a $3n+1$ opérations élémentaires, qui correspond à la complexité de la fonction.

Après, la fonction retourne une valeur (return s)

On dit ici que la complexité est *linéaire* car $C(n) = 3n + 2$, fonction donnant la complexité, est une fonction linéaire.

On dit alors que la complexité est en O(n) : cela signifie qu'elle est quasi-proportionnelle à n.