Les fichiers

1 Introduction

Un fichier peut être vu comme un ensemble de bits stockés dans la mémoire pour une utilisation future et qui sont interprétés selon le type de fichier.

Il existe généralement deux types de fichiers :

- les fichiers textes qui contiennent une suite lisible de caractères.
- les fichiers binaires qui contiennent généralement une suite de bits, non lisible directement, destinés à être interprétés comme autre chose que des caractères de texte comme d'images, de sons, d'exécutables, ...

On s'intéresse à la manipulation des fichiers texte qui peuvent être utilisés pour sauvegarder les résultats d'un programme pour une utilisation future.

2 Ouverture et fermeture de fichier

Tout fichier doit être ouvert avant de pouvoir accéder à son contenu en lecture et écriture.

La fonction $open(chemin\_fichier, mode)$ renvoie un objet fichier en utilisant deux arguments :

- Le premier est une chaîne de caractères représentant le chemin du fichier qui peut être :
  - ***relatif*** : basé sur le répertoire courant (généralement là où se trouve le programme en exécution).
  - ***absolu*** : emplacement dans la structure de système de fichiers, depuis la racine.
- Le deuxième argument est un caractère décrivant la façon dont le fichier est utilisé. Ce mode peut être :
  - ***'r'*** : (*read*) en lecture seulement (valeur par défaut);
  - ***'w'*** : (*write*) en écriture seulement en écrasant l'existant. Si le fichier n'existe pas, il sera créé;
  - ***'a'*** : (*append*) en écriture seulement à la fin du fichier. Si le fichier n'existe pas, il sera créé;
  - ***'x'*** : en création exclusive, pour prévenir l'écrasement ou l'ajout.

Pour fermer le fichier afin d'exploiter sa mise à jour et libérer les ressources système qu'il utilise on doit appeler la fonction *close*().

2.1 Exemple

[1]:
```
# En utilisant un chemin relatif, le fichier se trouve dans le même répertoire␣
↪du programme

mon_fichier = open('todo.txt', 'r')
mon_fichier.close()
```

[2]:
```
# En utilisant un chemin absolu que l'on peut obtenir en utilisant la fonction␣
↪getcwd() du module os

import os
print(os.getcwd())
```

```
/Users/yussuf
```

[3]:
```
mon_fichier = open('/Users/yussuf/todo.txt', 'r')
mon_fichier.close()
```

3 Méthodes des objets fichiers

3.1 Lecture

Pour lire le contenu d'un fichier, on utilise la fonction *read*(*taille*) : elle renvoie *taille* octets de données sous la forme d'une chaîne de caractères. Quand le paramètre *taille* est omis, le contenu entier du fichier est lu et donné.

3.1.1 Exemple

[4]:
```
# Ouverture

mon_fichier = open('todo.txt', 'r')

# Lecture du contenu entier
contenu = mon_fichier.read()

mon_fichier.close()

print(contenu)
```

```
Acheter du pain
Arroser les plantes
Payer la facture d'eau et d'électricité
```

Pour lire une seule ligne du fichier on utilise la fonction *readline*() qui, à chaque appel, renvoie la ligne suivante. Lorsque la fin du fichier est atteinte, la fonction renvoie une chaîne de caractères vide ''.

3.1.2 Exemple

[5]:
```
mon_fichier = open('todo.txt', 'r')

ligne1 = mon_fichier.readline()
ligne2 = mon_fichier.readline()
ligne3 = mon_fichier.readline()
ligne4 = mon_fichier.readline()


mon_fichier.close()

print("Première ligne : ", ligne1)
print("Deuxième : ", ligne2)
print("Troisième : ", ligne3)
print("Quatrième : ", ligne4)
```

```
Première ligne :  Acheter du pain

Deuxième :  Arroser les plantes

Troisième :  Payer la facture d'eau et d'électricité

Quatrième :
```

Pour construire une liste avec toutes les lignes d'un fichier, il suffit d'utiliser la fonction *readlines*(). Ainsi, on peut itérer sur les lignes à travers cette liste.

3.1.3 Exemple

[6]:
```
mon_fichier = open('todo.txt', 'r')

lignes = mon_fichier.readlines()

mon_fichier.close()

for ligne in lignes:
    print(ligne)
```

```
Acheter du pain

Arroser les plantes

Payer la facture d'eau et d'électricité
```

3.2 Écriture

La fonction *write*(*contenu*) écrit la chaîne de caractères *contenu* dans le fichier et renvoie le nombre de caractères écrits.

3.2.1 Exemple

[7]:
```
mon_fichier = open('todo.txt', 'a')

mon_fichier.write('Réserver le billet de train')

mon_fichier.close()
```

[8]:
```
mon_fichier = open('todo.txt', 'r')

contenu = mon_fichier.read()

mon_fichier.close()

print(contenu)
```

```
Acheter du pain
Arroser les plantes
Payer la facture d'eau et d'électricité
Réserver le billet de train
```

Pour écrire plusieurs lignes on utilise la fonction *writelines*(*liste_lignes*) qui prend la liste de lignes à écrire marquer de $\boldsymbol{\backslash n}$ à la fin.

[9]:
```
mon_fichier = open('todo.txt', 'w')

mon_fichier.writelines(['Regarder le documentaire Planète Terre\n', 'Acheter du␣
 ↪lait\n', 'Imprimer le document X\n'])

mon_fichier.close()
```

[10]:
```
mon_fichier = open('todo.txt', 'r')

contenu = mon_fichier.read()

mon_fichier.close()

print(contenu)
```

```
Regarder le documentaire Planète Terre
Acheter du lait
Imprimer le document X
```

Dans le cas d'une grande liste de lignes sans **\n** et pour faciliter, on peut recourir à la méthode *join* en utilisant **\n** comme séparateur :

[11]:
```
liste_lignes = ['Regarder le documentaire Planète Terre', 'Acheter du lait',␣
↪'Imprimer le document X']

mon_fichier = open('todo.txt', 'w')

mon_fichier.write('\n'.join(liste_lignes))

mon_fichier.close()
```

[12]:
```
mon_fichier = open('todo.txt', 'r')

contenu = mon_fichier.read()

mon_fichier.close()

print(contenu)
```

```
Regarder le documentaire Planète Terre
Acheter du lait
Imprimer le document X
```

4 TP

Exercie 1

1. Créer une fonction $fusionner(f1, f2)$ qui permet de fusionner deux fichiers $f1$ et $f2$.
2. Créer une fonction $copier(fichier, chemin\_absolu)$ qui permet de copier un fichier dans un répertoire en utilisant un chemin absolu.

Exercice 2

1. Écrire une fonction $nombre\_lignes(fichier)$ qui renvoie le nombre de lignes dans un fichier.
2. Écrire une fonction $nombre\_mots(fichier)$ qui renvoie le nombre de mots dans un fichier.
3. Écrire une fonction $nombre\_caractères(fichier)$ qui renvoie le nombre de caractères dans un fichier.
4. Écrire une fonction $taille(fichier)$ qui renvoie la taille en *bit* d'un fichier.

Exercice 3

1. Créer un fichier *log.txt* pour sauvegarder les tâches effectuées en incluant la date et le temps d'ajout.
2. Écrire une fonction $task\_done(description)$ qui prend la description d'une tâche effectuée et l'ajoute à la fin du fichier *log*. La ligne ajoutée doit être de la forme $date\ \ temps\ \ description$. Pour cela on peut utiliser la fonction *now* du module *datetime* qui renvoie la date et le temps actuels.

Exemple

[13]:
```
from datetime import datetime

print(f'{datetime.now()} description\n')
```

```
2025-05-02 18:25:38.723990 description
```

4. Écrire une fonction $task(date)$ qui renvoie les tâches effectuées en une date donnée sous la forme $année - mois - jour$.