TP 4 INFO — Manipulation de fichiers texte

Dans ce TP, nous allons étudier comment manipuler des fichiers texte avec Python. Ces possibilités sont très utiles notamment pour importer des données afin de pouvoir les manipuler avec Python ou pour exporter des résultats obtenus à partir d'un code en Python (afin de les sauvegarder).

Pour commencer, récupérez l'ensemble des fichiers texte que nous utiliserons dans la rubrique informatique du site se trouvant à l'adresse `http://vonbuhren.free.fr`, puis placez ces derniers au même emplacement que votre fichier Python.

Partie I Lire un fichier texte

I.A - Les commandes pour ouvrir, lire et fermer un fichier texte

Pour pouvoir accéder à un fichier texte, nous devons commencer par l'ouvrir en utilisant la fonction `open` en Python. Par exemple, pour ouvrir un fichier intitulé `exemple.txt`, on utilise l'instruction suivante.

```
fichier = open('exemple.txt', mode='r')
```

L'option `mode` indique à Python qu'on souhaite ouvrir le fichier en mode lecture (`'r'` pour *read* en anglais) : avec ce mode nous ne pourrons pas modifier le fichier, mais nous pourrons récupérer les informations qu'il contient. Une fois qu'on a fini de travailler avec le fichier, il est impératif de le refermer avec la commande `fichier.close()`.

```
fichier = open('exemple.txt', mode='r')
# INSTRUCTIONS POUR LIRE LE FICHIER (VOIR CI-DESSOUS)
fichier.close()
```

Une fois le fichier ouvert, on peut le lire de différentes manières. Pour illustrer ces différentes possibilités, nous supposerons dans la suite que le fichier texte `exemple.txt` contient les trois lignes suivantes.

```
ligne_0
ligne_1
ligne_2
```

- La commande `fichier.read()` permet de lire la totalité du fichier : le résultat est une chaîne de caractères que l'on peut donc stocker et manipuler. Il faut éviter d'utiliser cette méthode lorsqu'on manipule un fichier volumineux.

```
>>> fichier = open('exemple.txt', mode='r')
>>> fichier.read()
'ligne_0\nligne_1\nligne_2'
>>> fichier.close()
```

- Il est possible de spécifier un entier en paramètre en utilisant la commande `fichier.read(n)` : le résultat est une chaîne de caractères composée des n caractères suivants du fichier. Le curseur de lecture est alors placé juste après le n-ième caractère en attendant une prochaine instruction de lecture.

```
>>> fichier = open('exemple.txt', mode='r')
>>> fichier.read(3)
'lig'
>>> fichier.read(2)
'ne'
>>> fichier.close()
```

- La commande `fichier.readline()` permet de lire la prochaine ligne du fichier : le résultat est une chaîne de caractères que l'on peut donc stocker et manipuler. Après lecture d'une ligne, un curseur (invisible pour l'utilisateur) est placé au début de la ligne suivante en attendant son éventuelle lecture.

```
>>> fichier = open('exemple.txt', mode='r')
>>> fichier.readline()
'ligne_0\n'
>>> fichier.readline()
'ligne_1\n'
>>> fichier.close()
```

- La commande `fichier.readlines()` renvoie une liste dont les éléments sont les différentes lignes du fichier. Il faut éviter d'utiliser cette méthode lorsqu'on manipule un fichier volumineux.

```
>>> fichier = open('exemple.txt', mode='r')
>>> fichier.readlines()
['ligne_0\n', 'ligne_1\n', 'ligne_2']
>>> fichier.close()
```

- L'objet `fichier` est **itérable** : on peut le parcourir en utilisant une boucle for comme sur l'exemple suivant.

```
>>> fichier = open('exemple.txt', mode='r')
>>> for ligne in fichier:
...         print(ligne)
ligne_0

ligne_1

ligne_2
>>> fichier.close()
```

Remarque 1 : Le symbole `\n` que l'on peut observer dans les exemples précédents permet d'indiquer les retours à la ligne dans le fichier texte.

I.B - Quelques décimales de π

Dans cette sous-partie, on travaille avec le fichier `pi.txt` qui contient un certain nombre des premières décimales du nombre π.

Question 1 : Lire le fichier `pi.txt` et sauvegarder son contenu dans une chaîne de caractères.

Question 2 : Combien de décimales du nombre π contient le fichier `pi.txt` ?

Question 3 : Déterminer la liste `nb_occ` contenant dix éléments de sorte que l'élément d'indice `k` soit le nombre d'occurrences du chiffre `k` dans les décimales de π présentes dans le fichier `pi.txt`.

Question 4 : La date d'aujourd'hui au format JJMMAA apparaît-elle dans les décimales de π présentes dans le fichier `pi.txt` ?

I.C - Une liste de nombres

Dans cette sous-partie, on travaille avec le fichier `nombres.txt` qui contient un entier naturel par ligne.

Question 5 : Combien de nombres se trouve dans le fichier `nombres.txt` ?

Question 6 : Fabriquer une liste d'entiers contenant les nombres du fichier `nombres.txt`.

Dans la suite, nous allons utiliser la fonction ci-dessous.

```
def mystere(n:int) -> bool:
    res = True
    if n <= 1:
        res = False
    elif n >= 3:
        d = 2
        while d**2 <= n and res:
            if n % d == 0:
                res = False
            d += 1
    return(res)
```

Question 7 : Expliquer ce que fait la fonction `mystere` définie ci-dessus.

Question 8 : Parmi les nombres présents dans le fichier `nombres.txt`, combien sont des nombres premiers ?

Partie II Écrire des données dans un fichier

II.A - Les commandes pour ouvrir, écrire et fermer un fichier texte

Pour écrire dans un fichier texte, on doit l'ouvrir en mode écriture (`'w'` pour *write* en anglais). Si le le fichier ouvert en mode écriture existait préalablement, son contenu est effacé, sinon le fichier est créé automatiquement.
Une fois qu'on a fini de travailler avec le fichier, il est impératif de le refermer avec la commande `fichier.close()`.

```
fichier = open('exemple.txt', mode='w')
# INSTRUCTIONS POUR ÉCRIRE DANS LE FICHIER (VOIR CI-DESSOUS)
fichier.close()
```

Une fois ouvert, on utilise la commande `fichier.write` pour écrire dans le fichier.

```
fichier = open('exemple.txt', mode='w')
fichier.write('Voici une première ligne.\nVoici une seconde ligne.')
fichier.write('Voici la suite de la seconde ligne.')
fichier.close()
```

En ouvrant le fichier `exemple.txt` avec un éditeur de texte, son contenu sera le suivant.

```
Voici une première ligne.
Voici une seconde ligne.Voici la suite de la seconde ligne.
```

Remarque 2 : Si on souhaite ajouter des données à la fin dans un fichier texte préexistant (sans effacer son contenu), on doit ouvrir l'ouvrir en mode écriture avec l'option `'a'` (pour *append* en anglais).

II.B - L'algorithme de l'amitié

Dans cette sous-partie, on travaille avec le fichier `bbt213.txt` qui contient le script de l'épisode 13 de la saison 2 de la série « The Big Bang Theory » intitulé « L'algorithme de l'amitié » .

Question 9 : Combien de questions sont posées par les personnages durant cet épisode ?

Rappelons que chaque ligne du fichier texte est considérée comme une chaîne de caractères, donc on peut leur appliquer les méthodes usuelles pour ce type d'objets. Une des plus utiles est la méthode `split` qui permet de séparer une chaîne de caractères en une liste des sous-chaînes qui étaient séparées par une chaîne passée en paramètre.

```
>>> ligne = 'Archimède : Eureka !'
>>> ligne.split(:)
['Archimède ', ' Eureka !']
```

Vous pouvez également utiliser la méthode `strip` qui permet de supprimer les espaces se trouvant au début et à la fin d'une chaîne de caractères.

```
>>> mot = '  test   '
>>> mot.strip()
'test'
```

Question 10 : Écrire une fonction `repliques(perso:str) -> None` prenant en argument le nom d'un personnage `perso` et qui génère un fichier texte portant le nom du personnage (avec l'extension `.txt`) et contenant toutes ses répliques dans cet épisode.