TP n°1 : Fichiers textes en Python

Instruction générale : hormis pour les exercices corrigés collectivement, vous ferez valider votre travail par l'enseignant.

1 Ouverture et fermeture

1.1 Méthode `open close`

La commande `fichier = open(nom_fichier, mode, encodage)` en Python permet d'ouvrir un fichier. Une fois les opérations effectuées sur celui-ci, il faut utiliser la commande `fichier.close` pour le fermer. En effet, si le fichier reste ouvert, il peut être alors inutilisable pour d'autres utilisateurs ou applications en fonction des droits de celui-ci.

`nom_fichier :` le nom du fichier avec son éventuel chemin s'il n'est pas dans le même dossier que le fichier Python exécuté et son extension. Par exemple, dans notre cas `fichier.txt`.

`mode :` (optionnel) il existe quatre options :

`"r"`	pour `read`, il s'agit d'un mode de lecture – on obtient une erreur si le fichier n'existe pas.
`"w"`	pour `write`, il s'agit d'un mode de d'écriture – si un fichier du même nom existe déjà, il est écrasé, s'il n'existe pas, il est crée.
`"a"`	pour `append`, il s'agit d'un mode d'ajout – si le fichier n'existe pas, il est crée.
`"x"`	il s'agit d'un mode de création – si le fichier existe déjà, on obtient une erreur.

`encodage :` (optionnel) permet de spécifier l'encodage dans lequel on souhaite encoder ou décoder le fichier.

Le nom du fichier et le mode d'ouverture doivent écrit entre guillemets. Il existe en réalité plus d'options mais nous nous limiterons à celles-ci.

Exemple : `fichier = open("fichier.txt", "w", encoding = "utf-8")` ouvre le fichier `fichier.csv` en mode écriture (en UTF8) si celui-ci existe et le crée sinon. La commande `fichier.close` permet de le fermer.

1.2 Méthode `with open`

Cette méthode a l'avantage de fermer automatiquement le fichier lorsque la commande prend fin, il est inutile d'utiliser `close`.

```
with open(nom_fichier, mode, encodage) as fichier :
    bloc d'instructions
```

1.3 Gestion des erreurs avec `try except`

Le mode de lecture seule `r` peut provoquer une erreur si le fichier que l'on tente d'ouvrir n'existe pas. Les commandes `try except` permettent de tenter d'exécuter un bloc d'instructions et de gérer des exceptions si cela n'est pas possible.

```
try :
      bloc d'instructions

except :
      bloc d'instructions
```

Par exemple

```
try :
      with open("fichier.txt", "w", encoding = "utf-8") as fichier :
           bloc d'instructions

except FileNotFoundError :
      print("Le fichier n'a pas été trouvé")

except :
      print("Erreur inconnue")
```

va tenter d'ouvrir le fichier `fichier.txt` puis d'effectuer les instructions suivantes. Si le fichier n'est pas trouvable (erreur FileNotFoundError) le texte géré par la première exception est affiché. S'il s'agit d'une autre erreur, c'est la deuxième exception qui interviendra en affichant son texte.

Il est possible d'enchaîner plusieurs exceptions (`except`) en précisant ou non avec le nom d'une erreur Python (ZeroDivisionError, FileNotFoundError, etc). Il existe même d'autres commandes telles que `else` (qui s'exécute si aucune exception n'a été relevée) et `finally` (qui vient à la fin et s'exécute toujours).

2 Écriture

La commande `write(contenu)` permet d'écrire du contenu dans le fichier ouvert. Par exemple, le script

```
with open("fichier.txt", "w", encoding = "utf-8") as fichier :
      fichier.write("exemple")
```

va ouvrir ou créer le fichier `exemple.txt` en mode écriture, y écrire le texte « exemple » puis refermer le fichier.

Exercice 1. [Rio ne répond plus] Ouvrir en mode écriture le fichier `oss117.txt` puis y écrire le texte suivant avec une ligne pour chaque tiret.

– Une dictature c'est quand les gens sont communistes, déjà. Qu'ils ont froid, avec des chapeaux gris et des chaussures à fermeture éclair. C'est ça, une dictature, Dolorès.
– D'accord. Et comment vous appelez un pays qui a comme président un militaire avec les pleins pouvoirs, une police secrète, une seule chaîne de télévision et dont toute l'information est contrôlée par l'État ?
– J'appelle ça la France, mademoiselle. Et pas n'importe laquelle ; la France du général de Gaulle.

Exercice 2. Ouvrir un fichier texte au nom de cet exercice et écrire sur les mille premières lignes le texte « ligne {numéro de la ligne} ».

Exercice 3. Ouvrir dix fichiers textes aux noms `test_{numéro du fichier}.txt` et écrire dans chacun d'entre eux le texte « test {numéro du fichier} ».

3 Lecture

Tant que le fichier est ouvert, on peut lire son contenu grâce aux commandes suivantes.

`read(n)`	lit les `n` premiers caractères du fichier (sans argument, la totalité) ; stockage dans une chaîne de caractère.
`readline()`	ne lit qu'une ligne à la fois ; stockage dans une chaîne de caractère.
`readlines()`	lit la totalité du fichier (toutes les lignes restantes si les méthodes `read` et `readline` ont été utilisées au préalable) ; stockage dans une liste.

Par exemple, le script suivant ré-ouvre le fichier `exemple.txt` et en lit les trois premiers caractères donc « exe ».

```
fichier = open("exemple.txt", "r")
print(fichier.read(3))
fichier.close
```

Exercice 4. [Perdu de recherche] Écrire un programme prenant en entrée le nom d'un fichier texte puis un mot et donnant en sortie les numéros de ligne où apparaissent le mot dans le texte ; s'il n'y apparaît pas, le programme devra en avertir l'utilisateur. Il devra aussi le faire si l'utilisateur entre un nom de fichier incorrect ou introuvable. On pourra tester le programme sur le fichier téléchargeable test.txt dans le dossier correspondant à ce TP.

Exercice 5. [Le grand remplacement] Écrire un programme prenant en entrée le nom d'un fichier texte puis deux mots et remplaçant le premier par le second. Si le premier mot n'y apparaît pas, si l'utilisateur entre un nom de fichier incorrect ou introuvable, le programme devra en avertir l'utilisateur. On pourra tester le programme sur le fichier utilisé dans l'exercice précédent.