L'utilisation de fichiers textes permet d'enregistrer des données du type chaines de caractères sur le disque dur sous la forme de lignes de texte.

1. Ouverture d'un fichier

La fonction open(*nom, option*) permet d'ouvrir un fichier en lecture et/ou en écriture.

- *fichier* = open('*nom du fichier*','r') ouvre un fichier existant en lecture (read).
- *fichier* = open('*nom du fichier*','r+') ouvre un fichier existant en lecture/écriture
- *fichier* = open('*nom du fichier*','w') ouvre un nouveau fichier en écriture (write/overwrite). Si un fichier existe déjà, il est écrasé.
- *fichier* = open('*nom du fichier*','a') ouvre un fichier en ajout (append). Si le fichier n'existe pas, il est créé.

fichier est le nom que vous donnez à l'objet fichier dans votre programme, '*nom du fichier*' est son nom sur le disque dur, incluant éventuellement un chemin d'accès, comme 'C:\\Users\\Pantxi\\Documents\\monfichier.txt'.

'w+' ouvre le fichier en écriture et lecture, 'a+' ouvre en ajout et lecture.

Les options 'rb', 'rb+', 'wb', 'wb+', 'ab', 'ab+' ouvrent le fichier en mode binaire.

Remarque : os.getcwd() vous donne le répertoire courant (après import os) dans lequel sont créés tous les fichiers si on ne précise pas un chemin.

2. Opérations sur un fichier ouvert

Les opérations sur les fichiers s'effectuent via un pointeur. L'ouverture d'un fichier en option read (r, r+, rb, rb+) ou write place le pointeur en début de fichier, l'ouverture en option append place le pointeur en fin de fichier.

- *fichier*.close() ferme le fichier.
- *fichier*.name renvoie le nom d'un fichier.
- *fichier*.read(*n*) renvoie *n* caractères. *fichier*.read() lit tout le fichier.
- *fichier*.readline() renvoie la ligne courante, renvoie une chaine vide si on est en fin de fichier.
- *fichier*.readlines() renvoie les lignes du pointeur à la fin dans une liste (un élément par ligne).
- *fichier*.write('texte') ou *fichier*.writelines(['texte1','texte2',…]) écrivent le texte à la fin du fichier. write et writelines() n'effectuent pas de retour à la ligne. Il faut inclure '\n' dans le texte pour aller à la ligne.
- *fichier*.seek(*n*) place le pointeur au *n*-ième octet du fichier.
  fichier.seek(0,2) place le pointeur à la fin du fichier.
  fichier.tell() renvoie la position du pointeur.

Le pointeur se déplace toujours à la fin de la chaîne de caractères traitée. Ainsi, des readline() consécutifs liront des lignes différentes.

Remarque : *fichier*.name est sans parenthèse car name est une variable (encapsulée dans l'objet fichier), les autres opérations sont des méthodes (des fonctions qui s'appliquent à des objets), avec parenthèses donc.

3. Gestion des erreurs

Les accès aux fichiers génèrent parfois des erreurs appelées exceptions (fichier inexistant, disque plein…). Afin que le programme ne s'interrompe pas, on doit gérer ces exceptions. On utilise la commande **try** … **except** et si besoin **else** et/ou **finally** comme dans l'exemple suivant.

```
try:
    fichier = open('blah.txt','r')
    print(fichier.readline())
    fichier.close()
except:
    print("Cette ligne s'exécute en cas d'exception ")
else:
    print("Celle-ci s'il n'y a pas d'exception")
finally:
    print("Celle-ci s'exécute toujours avant de quitter le bloc try")
```

Intérêt de finally: les instructions du bloc finally sont exécutées même s'il y a un return dans un bloc except ou else, contrairement à des lignes de code qui suivraient le bloc try.

On peut créer plusieurs blocs **except** en précisant le type d'exceptions :

```
except IOError:
except TypeError:
```

L'inconvénient de ce type de code est que s'il se produit une exception entre l'ouverture et la fermeture du fichier, celui-ci ne sera pas refermé. On doit rajouter un autre bloc **try** avec fichier.close() dans le bloc **finally**.
Une solution plus simple est d'ouvrir le fichier avec la commande **with.** Il sera automatiquement refermé lors de la sortie du bloc **with**, même en cas d'exception.

```
with open('blah.txt','r') as fichier:
    print(fichier.readline())
```

4. Un exemple de programme inutile

```
import numpy as np

def decpi():
    pi = str(np.pi)[2:]
    with open('decimalespi.txt','w+') as fichierpi:
        try:
            for i in range(len(pi)):
                fichierpi.write('La '+str(i+1)+'-ième décimale de pi est '+pi[i]+'\n')
            fichierpi.seek(0)
            n = int(input("Entrer n : "))
            for i in range(0,n-1):
                fichierpi.readline()
            print(fichierpi.readline())
        except:
            pass
```

5. Manipulation de chaines de caractères

Les commandes d'accès aux fichiers utilisent des chaines de caractères. Il faut donc être familiarisé avec ces objets.

Les chaines de caractères se notent entre apostrophes (' ') ou guillemets (" "). On utilisera des guillemets si la chaine contient des apostrophes et réciproquement. Les concaténations s'effectuent avec le signe +.

Exemple : chaine = "j'ai " + str(20) + " ans."

Une chaine peut être parcourue comme une liste : chaine[0] renvoie le premier caractère de chaine, chaine[1:] renvoie chaine privée du premier caractère, chaine[::2] renvoie la chaine obtenue en prenant un caractère sur deux…

La fonction len() renvoie la longueur d'une chaine de caractères.
Une chaine de caractères est itérable.

Exemple :

```
for i in "bonjour":
    print(i)
```

Les méthodes upper() et lower() convertissent une chaine en majuscules et minuscules. Les méthodes count(), find(), replace() et split() peuvent également être utiles.
Exemples :

```
In [1]: "bonjour".count("o")
Out[1]: 2

In [2]: "bonjour".find("jo")
Out[2]: 3

In [3]: "bonjour".replace("o","i")
Out[3]: 'binjiur'

In [4]: 'nom du fichier'.split()
Out[4]: ['nom', 'du', 'fichier']
```