• Il est possible de lire et d'écrire dans des fichiers en utilisant Python, c'est l'un des points fort de ce langage.

• Lors d'une expérience dans un laboratoire de physique, les résultats fournis par une carte d'acquisition sont enregistrés par le logiciel d'étude dans un fichier texte (ou dans un tableur).

Il est possible d'ouvrir ce fichier en Python, d'en lire les lignes afin de mettre les résultats dans des listes et de tracer des graphiques.

• Dans beaucoup d'exemples, nous allons utiliser des fichiers "texte" avec l'extension .txt. On pourra aussi manipuler des fichiers avec l'extension .csv (comma separated values), ce format est le format universel pour manipuler des bases de données, il est utilisé dans des applications comme Microsoft Excel, Numbers, le tableur Google.

1 Découverte des commandes

1.1 Lecture d'un un fichier

• À l'aide d'un éditeur de texte, créez un fichier que vous enregistrerez dans votre répertoire de travail sous le nom *zoo.txt* avec le contenu suivant :

vache
tigre
loup
fourmi
girafe
singe

Tester ensuite le code suivant directement dans la console Python.

```
>>>fic = open("zoo.txt", "r")
# fic est le nom sous lequel j'ai choisi d'ouvrir le fichier
# si le fichier n'est pas trouvé, il faudra préciser le chemin d'accès complet
# l'option "r" (read) va permettre de lire le fichier (et pas de le modifier)
# le fichier est ouvert, à présent on veut récupérer les informations

>>> fic.readlines()
['vache\n', 'tigre\n', 'loup\n', 'fourmi\n', 'girafe\n', 'singe\n']
# cette commande permet de récupérer toutes les lignes de notre fichier sous la forme d'une liste de
    chaines de caractères
# \n est un retour à la ligne

>>>fic.close()
# pour fermer le fichier une fois l'utilisation terminée
```

Voici comment afficher les informations obtenues, essayer les lignes de code suivantes :

```
fic = open("zoo.txt", "r")
lignes = fic.readlines()
for i in lignes:
    print(i)

fic.close()
```

Par défaut l'instruction *print*() affiche quelque chose puis revient à la ligne. Ce retour à la ligne se cumule avec celui de fin de ligne (\n), c'est pour cela qu'une ligne est sautée.

• À la place de *.readlines*(), vous pouvez utiliser la méthode *.read*() qui lit tout le contenu du fichier et renvoie une chaine de caractères. Il existe aussi la méthode *.readline*() qui renvoie une seule ligne sous la forme d'une chaine de caractères, à chaque nouvel appel de *.readline*() la ligne suivante sera lue. Tester ces commandes.

• Voici une dernière technique simple et élégante pour parcourir notre fichier. C'est la méthode que nous allons privilégier dans toute la suite.

```
fic = open("zoo.txt", "r")
for l in fic: # permet de parcourir les lignes du fichier
    print(l)
```

1.2 Mise en forme des données

• Un travail de mise en forme est à faire avant d'exploiter les données. On considère le fichier texte suivant :

3.2, 4.5, 7.2
4.2, 4.9, 3.2
4.2, 3.3, 0.9
1.2, 4.3, 9.7

On souhaite, à titre d'exemple, récupérer dans une liste le deuxième nombre de chaque ligne. Testez-vous-même les commandes suivantes qui permettent de faire ceci en Python :

```
>>>fic = open("données.txt", "r") # le nom de mon fichier
>>>L = fic.readlines()
>>>L
['3.2, 4.5, 7.2\n', '4.2, 4.9, 3.2\n', '4.2, 3.3, 0.9\n', '1.2, 4.3, 9.7\n']
# on a bien une liste de chaines de caractères

>>>l = L[0] # on travaille d'abord sur la première ligne
>>>l
'3.2, 4.5, 7.2\n'

>>>l = l.strip()
>>>l
'3.2, 4.5, 7.2'
# la méthode .strip() permet de supprimer les retours à la ligne, les tabulations et les blancs placés au
    début et à la fin de la chaine de caractères

>>>l = l.split(",")
>>>l
['3.2', ' 4.5', ' 7.2']
# la méthode .split() permet de découper une chaine de caractères et de placer le résultat dans une liste
# ici le découpage s'est fait selon le caractère "," mais on pouvait utiliser un autre délimiteur
# par défaut c'est l'espace qui est le caractère selon lequel on délimite

>>>l = [float(i) for i in l]
>>>l
[3.2, 4.5, 7.2]
# on convertit les chaines de caractères en nombres flottants

>>>l[1]
4.5
# pour récupérer le deuxième nombre comme voulu
```

Il reste à faire cela à chaque ligne à l'aide d'une boucle pour récupérer la liste voulue. Écrivez le programme qui réalise ceci.

1.3 Écriture sur un fichier

• Écrire sur un fichier se fait aussi très bien en Python. Tester les commandes suivantes :

```
fic = open("essai.txt", "w")
# l'option w (write) permet d'ouvrir le fichier en mode écriture
# vous pouvez aussi préciser le dossier dans lequel vous voulez sauvegarder votre fichier

for i in range(10):
    fic.write(str(i ** 2) + "\n")
# on écrit par exemple les carrés (que l'on convertit en chaine de caractères) avec un retour à la ligne
# l'opération + entre chaines de caractères effectue une concaténation

fic.close() # on ferme le fichier pour enregistrer les modifications
```

Vous pouvez à présent vérifier le contenu de votre fichier.

• Si le fichier est fermé puis à nouveau ouvert en écriture ultérieurement tout ce qui était écrit auparavant est perdu. Pour écrire à nouveau dans un fichier déjà fermé, on utilise l'ouverture en mode "ajout", c'est-à-dire avec l'option "a". Tester vous-même ce procédé.

2 Premiers exercices

2.1 Exploitation de données

Le fichier *Notes.txt* contient des notes à un devoir fictif. Ouvrez-le pour en voir le contenu et la façon dont les données sont organisées.

1. (a) Écrire un programme qui permet de récupérer la liste des notes.
   (b) En déduire la moyenne obtenue à ce devoir.
   (c) Tracer l'histogramme des notes.
2. Écrire un programme qui permet d'afficher le nom de l'étudiant ayant eu la meilleure note et le nom de l'étudiant ayant eu la moins bonne note. On effectuera une recherche d'un maximum et d'un minimum même si on peut voir en regardant le fichier que la note minimale est 0 et la note maximale est 20.
3. Les étudiants ayant obtenu 10 ou plus sont admis à cet examen. Écrire un programme qui renvoie la liste des étudiants admis.
4. (a) Reprendre la liste des notes obtenues à la question 1. Écrire une fonction $rang(L, note)$ qui prend en paramètre cette liste L que l'on suppose triée dans l'ordre décroissant et la note examinée et qui renvoie le classement correspondant à cette note. On pourra utiliser méthode *.sort*() qui permet de trier une liste et la méthode *.reverse*() qui renverse la liste.
   (b) À l'aide la fonction précédente former le dictionnaire où les clés sont les noms des étudiants et les valeurs sont des couples formés de la note et du rang.

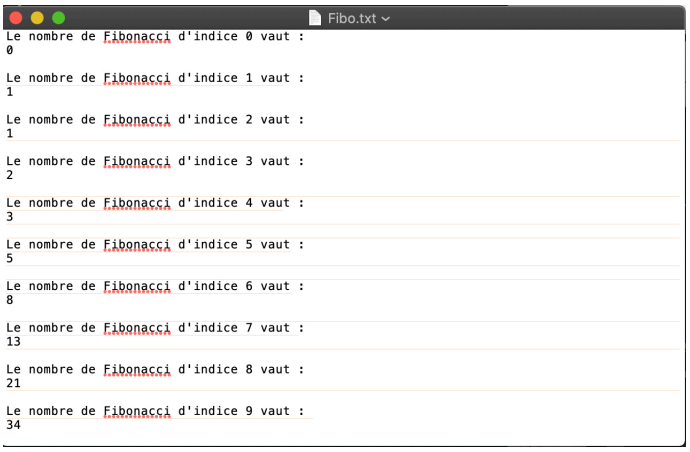
2.2 Tracé d'un graphique

Tracer la courbe représentant y en fonction de x à partir du fichier "Courbe.txt". Vous aurez besoin de la méthode *.replace*() qui permet de remplacer un caractère par un autre, comme dans l'exemple suivant :

```
>>>ch = "Le chat monte sur la table"
>>>ch = ch.replace("e", "z")
>>>ch
'Lz chat montz sur la tablz'
```

2.3 Écriture des nombres de Fibonacci

Écrire un programme qui permet de créer un fichier texte contenant les 10000 premiers nombres de Fibonacci. La forme attendue pour le début de ce fichier est celle-ci.



2.4 Statistiques sur les décimales de π

Le fichier *pi_100000.txt* contient environ 100000 décimales de π.

1. Écrire une fonction qui place toutes ces décimales dans une liste d'entiers.
2. Tracer l'histogramme correspondant. Que remarque t-on ?
3. Écrire un programme qui permet de vérifier si votre date de naissance (au format suivant, par exemple : 220703) apparait dans les 100000 premières décimales de π.

3 Le jeu du pendu

Écrire un programme permettant de simuler le jeu du pendu, dans lequel on doit deviner un mot en proposant des lettres. Quelques consignes :

- Le mot sera choisi au hasard dans le fichier "Mots.txt".
- Votre programme va interagir avec l'utilisateur en lui demandant une lettre.
- On affichera à chaque fois les lettres trouvées et les lettres cachées seront sous forme d'étoiles.
- On laissera 8 essais au joueur.