Partie 1

Entrées/Sorties en console

1.1 Sortie en console

La fonction `print` permet l'affichage d'une valeur en console, quel qu'en soit le type.

On peut afficher un texte simple, ou composé de plusieurs parties, par concaténation de chaine :

```
clan = 'Vikings'
message = 'Fight Invaders'
print('We are the ' + clan + ' who say' + ' "'+ message + '!"')
# affiche
# We are the Vikings who say "Fight Invaders!"
```

Le problème de cette méthode de concaténation est que chaque partie composant la chaine doit être de type `str`. Cela est possible avec la fonction `str(variable`, mais il existe une méthode plus efficace : **formater la chaine**.

La méthode `str.format()` sur les chaines de caractères :

La methode de base :

```
clan = 'Vikings'
message = 'Fight Invaders'
print('We are the {} who say "{}!"'.format(clan, message))
# affiche
# We are the Vikings who say "Fight Invaders!"
```

On peut aussi préciser le nombre de caractères ou de chiffres attendus, le type de données, etc.

Exemple : affichage d'une table

```
for x in range(1, 11):
    print('{:2d} {:3d} {:4d}'.format(x, x*x, x*x*x))
# affichage :
 1   1    1
 2   4    8
 3   9   27
 4  16   64
 5  25  125
```

Cela permet d'aligner des chaînes dans une largeur de taille fixe.

1.2 Entrées au clavier : input

La fonction `input` met le programme en pause et attend que l'utilisateur entre une donnée.

```
nom = input('Quel est votre nom ? ')
print('Bonjour, {}'.format(nom))
```

Toutes les données saisies sont converties en chaîne. Il est donc necessaire de convertir les données numériques dans le type approprié avant de les utiliser :

```
prix = input('Combien coute ce camion ?')
prix = float(prix)
if prix < 10000:
  print('Ok, je le prends.')
```

Partie 2

Lire le contenu d'un fichier

2.1 Fichier txt

On utilise la fonction open de la librairie standart en python :

La fonction open crée un fichier s'il n'existe pas encore. Elle renvoie un objet de la classeTextIoWrapper. Par la suite, nous allons utiliser des méthodes de cette classe pour interagir avec le fichier. Il existe 2 méthode pour utiliser open :

- méthode 1 : fonctions open et `close`. On créé un objet qui sert à importer le fichier. Choisir un nom quelconque, pour l'exemple qui suit ce sera `file`. On utilise la fonction open avec pour paramètres, le nom du fichier (son chemin), suivi du caractère 'r' (pour `read`, ouverture du fichier en *lecture*).

On peut alors récuperer le contenu textuel du fichier à l'aide de la méthode `read`. Dans l'exemple ci-dessous, on place le texte du fichier dans la variable `texte`.

On peut traiter cette variable comme on le souhaite, l'afficher en entier (`print(texte)`), ou parcourir et traiter les caractères les uns après les autres (`for c in texte: ...`)

Penser alors à fermer le fichier de lecture : `file.close()`

```
file = open('exemple.txt','r')
texte = file.read()
file.close()
print(texte)
```

- méthode 2 : système d'instructions `with` + `open`

Le programme suivant est équivalent à celui de la méthode 1. Avec le mot clé `with`, les instructions de lecture du fichier sont mis dans un bloc, et donc indentées. A la sortie de ce bloc, le fichier est automatiquement *fermé*, ce qui fait l'economie de l'instruction `close`, et evite les oublis...

```
with open('exemple.txt', 'r') as file:
    texte = file.read()
print(texte)
```

2.2 Caractères speciaux

Certains caractères sont fort utiles lors de l'écriture de fichiers texte afin d'organiser les données. Le symbole ; est très utilisé comme séparateur de colonnes pour une matrice, on utilise également le passage à la ligne ou la tabulation. Comme ce ne sont pas des caractères « visibles », ils ont des codes :

- \ caractère d'echappement (par exemple pour écrire des guillemets dans une chaine)
- `\n` : passage à la ligne
- `\t` : tabulation, indique un passage à la colonne suivante dans le format tsv (Tabulation-separated values).

Lors de la **lecture d'un fichier**, on peut avoir besoin de supprimer ces caractères spéciaux :

2.3 Fichier csv

Le module csv permet de reduire d'une ligne l'import des données (choisir le bon *delimiter*) :

```
import csv
table = []
with open('ficher.csv', 'r') as file:
    reader = csv.reader(file, delimiter=',')
    for row in reader:
        table.append(row)
```

- Souvent, la première ligne ne concerne que l'en-tête (descripteurs).
- Le reste des données, numériques doivent être converties en *float*
- Supposons que les données contiennent les valeur de x pour la première colonne, et y la deuxième colonne :

```
x = []
y = []
for i in range(1,len(table[1:])):
    x.append(float(table[i][0]))
    y.append(float(table[i][1]))
```

Partie 3

Ecrire dans un fichier

- méthode 1 : fonctions `open` et `close`. On créé un objet qui sert à désigner le fichier. Choisir un nom quelconque, pour l'exemple qui suit ce sera `fileout`. On utilise la fonction `open` avec pour paramètres, le nom du fichier (son chemin), suivi du caractère 'w' (pour `write`, ouverture du fichier en *écriture*).

On peut alors placer du texte dans le fichier à l'aide de la méthode `write` : `fileout.write("...")`

Penser alors à fermer le fichier de lecture : `fileout.close()`

```
fileout = open('exemple.txt','w')
fileout.write("Je sais ta passion, et suis ravi de voir\nQue tous
    ses mouvements cèdent à ton devoir ;")
fileout.close()
```

On peut alors vérifier le fichier et son contenu en parcourant le dossier avec l'explorateur windows.

- méthode 2 : système d'instructions `with` + `open`

Le programme suivant est équivalent à celui de la méthode 1. Avec le mot clé `with`, les instructions de d'écriture dans le fichier sont mis dans un bloc, et donc indentées. A la sortie de ce bloc, le fichier est automatiquement *fermé*, ce qui fait l'economie de l'instruction `close`, et evite les oublis...

```
with open('exemple.txt','w') as fileout:
    fileout.write("Je sais ta passion, et suis ravi de voir\nQue
   tous ses mouvements cèdent à ton devoir ;")
```

Exemple avec une liste => fichier

```
animaux2 = ["poisson", "abeille", "chat"]
with open("zoo2.txt", "w") as filout:
    for animal in animaux2:
         filout.write(animal)
```

à la ligne 2 : ouverture du fichier `zoo2.txt` en mode écriture, avec le caractère `w pour write.`

Si nous ouvrons le fichier zoo2.txt avec un éditeur de texte, voici ce que nous obtenons :

`poissonabeillechat`

Ce n'est pas exactement le résultat attendu. Le script peut être amélioré en utilisant une écriture formatée, comme vu plus haut, à la dernière ligne du script :

```
         filout.write("{}\n".format(animal))
```

Ce qui donne maintenant, à l'ouverture du fichier :

```
poisson
abeille
chat
```

3.1 Tableau de données

- convertir les données en chaine de caractère formatée :

```
descripteurs = ['courant','tension','puissance']
datas = [[0.100,2.00,0.200],[0.200,4.00,0.400]]
tab = ""
for title in descripteurs:
    # ajout des descripteurs en tete du tableau
    tab = tab + title + ','
for i in range(len(datas)):
    # saut de ligne
    tab = tab + '\n'
    for j in range(len(datas[0])):
        tab = tab + datas[i][j] + ','
```

- exporter avec une extension *csv*

```
with open("fichier_datas.csv", "w") as filout:
    filout.write(tab)
```