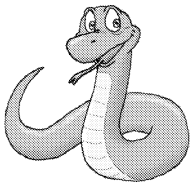
IPT : Cours 10



Lecture / Ecriture dans un fichier avec Python (2h)

MPSI : Prytanée National Militaire

Pascal Delahaye

13 janvier 2017

Même si très souvent les données d'un programme sont fournies par l'utilisateur par l'intermédiaire du clavier et les sorties sont affichées à l'écran, la communication entre un programme et l'extérieur peut aussi se faire sous la forme de lecture et d'édition de fichiers. Ce cours a pour objectif de présenter comment un programme peut aller rechercher ou sauvegarder des informations dans un fichier.

1 Mise en place d'un environnement de travail adapté

Lorsqu'un fichier est appelé ou est créé par un programme, celui-ci doit savoir où aller le chercher et où l'enregistrer. Ce paramétrage s'effectue dans le menu "outils - préférences - répertoire de travail global" de Spyder.
Pour les manipulations qui vont suivre nous allons choisir de mettre les fichiers manipulés et le fichier contenant les programmes Python dans le même répertoire.

Pour cela :

1. Il faut créer un répertoire de travail dans notre espace personnel : "python"
2. Il faut préciser dans les préférences de Spyder que désormais, le *répertoire de travail global* est "python"
   On pourra pour cela, modifier les options : "executer" et "répertoire de travail"
3. Il faut sauvegarder les fichiers textes que l'on souhaite lire dans ce répertoire.

Manipulations à effectuer :

1. Créer un répertoire "python" dans votre espace personnel ou sur le bureau de l'ordinateur
2. Créer un fichier txt contenant les différents paragraphes :

   "toto va à la plage" - "titi va se faire manger" - "tutu aime la danse"

   Enregistrer-le sous le nom "toto.txt" dans le répertoire "python" précédent
3. Paramétrez correctement le répertoire de travail global de Spyder

2 Lecture d'un fichier

2.1 La fonction `open()`

La fonction `open()` permet "d'importer" sous python un fichier en lecture (et/ou en écriture...).
Elle a pour effet de créer une copie de ce fichier dans la mémoire vive de l'ordinateur afin de le lire.

La structure de cette fonction est la suivante : `f = open('nom_du_fichier.txt', 'r')`

- `f` est le nom de la copie du fichier que nous sommes en train de créer et sur laquelle nous allons travailler.
- `'nom_du_fichier.txt'` est le nom du fichier que nous souhaitons importer
- `'r'` est l'argument qui indique que la copie que nous créons est destinée à être lue ('r' comme 'read')

Manipulations à effectuer :

1. Créer à l'aide de la fonction `open()` une copie de travail (en lecture) du fichier "toto.txt" créé précédemment. Vous appellerez "toto" cette copie.

Remarque 1. Lorsqu'on souhaite lire un fichier bitmap, on utilise l'argument `'rb'` au lieu de `'r'` (vu en 2ème année!).

2.2 Lecture du fichier importé

La lecture du fichier `f` se fait grâce aux *méthodes* : `f.readline()` et `f.readlines()`.
Pour comprendre leur fonctionnement, il faut imaginer qu'un curseur est placé en début de fichier lors de son importation.

1. La méthode `f.readline()` a pour effet de stocker dans une chaîne de caractères le premier paragraphe du texte contenu dans le fichier `f` et de placer le curseur au début du paragraphe suivant. Ainsi, la prochaine utilisation `f.readline()` permet de récupérer le second paragraphe et ainsi de suite... Lorsque le curseur arrive à la fin du fichier, la commande `f.readline()` renvoie la chaîne de caractères vide `''`.

2. La méthode `f.readlines()` a quant à elle pour effet de stocker dans une liste les différents paragraphes du texte contenu dans le fichier `f`. Le premier paragraphe étant stocké sous la forme d'une chaine de caractères dans la première composante, le deuxième dans la deuxième... etc... Comme la fonction `f.readline()`, elle ne porte que sur la partie du texte qui suit le curseur.

Manipulations à effectuer :

1. Taper l'instruction : `>>> C1 = toto.readline()`
   Vérifier le contenu de la variable C1 dans l'explorateur de variable.
2. Taper l'instruction : `>>> C2 = toto.readline()`
   Vérifier le contenu de la variable C2 dans l'explorateur de variable.
3. Taper l'instruction : `>>> C3 = toto.readline()`
   Vérifier le contenu de la variable C3 dans l'explorateur de variable.
4. Taper l'instruction : `>>> C4 = toto.readline()`
   Vérifier le contenu de la variable C4 dans l'explorateur de variable.

Remarque 2. On constate que le retour à la ligne et la tabulation sont codés respectivement sous la forme `\n` et `\t`

Manipulations à effectuer :

1. Taper l'instruction : `>>> L = toto.readlines()`
   Vérifier le contenu de la variable L en tapant l'instruction `>>> L`
   Interpréter le résultat obtenu...

2. Créer une nouvelle copie du fichier "toto.txt" à l'aide de la fonction `open()`
   Retaper l'instruction : `>>> L = toto.readlines()`
   Vérifier le contenu de la variable L en tapant l'instruction `>>> L`
   On peut alors récupérer les différents paragraphes du texte en tapant `L[0]`, `L[1]`, et `L[2]`.

2.3 2 méthodes de traitement d'une chaîne de caractères

La méthode `C.strip()` :

Elle permet d'éliminer les espaces et les fins de paragraphe en début et en fin d'une chaine de caractères `C`.

`Exemple 1.` Nettoyer le début et la fin de la phrase : " toto, 23, rouge \n"

La méthode `C.split('x')` :

Elle permet de décomposer une chaine de caractères `C` sous la forme d'une liste de chaines de caractères dont les composantes correspondent aux différentes parties de la chaîne `C` séparées par le séparateur `x`.

Les séparateurs usuellement utilisés sont : \t . , ; et :

`Exemple 2.` Décomposez sous la forme d'une liste la chaîne de caractères suivante : "toto, 23, rouge".
Vous prendrez tour à tour les séparateurs "," et "o"

2.4 Application : La lecture par python d'un fichier excel

Pour convertir un fichier excel en un fichier de données lisible par python, le fichier doit être enregistré au format `*.csv`.

Manipulations à effectuer :

1. Créer un fichier excel ou openoffice contenant la ligne les données suivantes :

toto	23	rouge
titi	12	vert
tutu	57	rose

2. Enregistrer ce fichier au format csv sous le nom : `toto.csv`
3. Ouvrir ce fichier à l'aide du blocnote pour voir ce qu'il contient.
4. Concevoir un programme python permettant de calculer la somme des nombres de la deuxième colonne.

2.5 Fermeture d'un fichier en lecture

L'ouverture d'un fichier `f` en lecture mobilise de la mémoire vive. Si le fichier n'a plus besoin d'être lu, il faut impérativement libérer cette mémoire en utilisant la méthode

```
f.close()
```

2.6 Exercice

Commencer par créer un fichier "premiers.text" dans lequel on stocke les deux paragraphes suivants :

```
2, 3, 5, 7, 11, 13, 17, 19,
23, 29, 31, 37, 41, 43
```

Concevoir un programmme python permettant de calculer la somme de tous les nombres contenus dans ce fichier.
Vous n'oublierez pas de fermer le fichier en lecture à la fin du programme.

La conversion d'une chaine de caractères en un nombre se fait à l'aide des fonctions : `eval()`, `int()` ou `float()`

3 Ecriture dans un fichier

3.1 Ecriture dans un nouveau fichier

• CREATION :

La fonction `open()` permet également de créer sous python un fichier en écriture.
Elle a pour effet de créer un fichier une copie de travail dans la mémoire vive (RAM).

La structure de cette fonction est la suivante : `s = open('nom_du_fichier.txt', 'w')`

- `s` est le nom de la copie du fichier en écriture que nous sommes en train de créer.
- `'nom_du_fichier.txt'` est le nom que portera le fichier dans le répertoire de travail global.
- `'w'` est l'argument qui indique que nous créons une copie en écriture ('w' comme 'write')

Manipulations à effectuer :

1. Créer un fichier en écriture nommé "resultats.txt" et dont la copie python est nommés `s`.
   Vérifiez que ce fichier a bien été créé dans votre répertoire de travail global et vérifiez son contenu.

Remarque 3. Lorsqu'on souhaite écrire un fichier bitmap, on utilise l'argument `'wb'` au lieu de `'w'` (vu en deuxième année !).

• ECRITURE : L'écriture dans un fichier se fait alors grâce à la méthode : `f.write()`

Manipulations à effectuer :

1. taper l'instruction : `s.write('voyons si ça marche?')`
   Vérifier si le fichier "resultats.txt" contient la phrase précédente.

2. taper l'instruction : `s.close()`
   Vérifier si le fichier "resultats.txt" contient la phrase précédente.

   là encore, la fermeture d'un fichier en écriture est indispensable car elle permet d'enregistrer dans le fichier présent sur le disque dur, les informations inscrites dans la copie python...

`Exemple 3.` Fonction créant un fichier contenant tous les nombres pairs allant de 2 à 2N, en revenant à la ligne après chaque multiple de 10 :

```
def pair(N):
    f = open('pairs.txt','w')                    # Création du fichier en écriture
    for i in range(0,N+1):
        x = 2*i
        if x%10 == 0 : f.write(str(2*i) + '\n') # retour à la ligne si x est pair
        else : f.write(str(2*i) + ' - ')        # Ecriture des nombres pairs dans le fichier
    f.close()                        # Enregistrement des données dans le fichiers "pairs.txt"
```

Vous pouvez maintenant ouvrir le fichier "pairs.txt" qui vient d'être créé et vérifier qu'il contient bien les nombres pairs souhaités.

Remarque 4.

1. Bien retenir que l'instruction `f.write()` n'inscrit que des chaînes de caractères.
   Il faut donc convertir la valeur `2*i` en chaîne de caractères grâce à l'instruction `str()`.
2. Vous constaterez que l'ouverture en écriture d'un fichier :
   — soit crée un nouveau fichier lorsqu'il n'existait pas,
   — soit écrase automatiquement le fichier contenant un nom identique.

3.2 Ajout dans un fichier existant

• CREATION :

La structure de cette fonction est la suivante : `s = open('nom_du_fichier.txt', 'a')`

- `s` est le nom de la copie du fichier auquel nous souhaitons ajouter des éléments.
- `'nom_du_fichier.txt'` est le nom du fichier auquel on souhaite ajouter des éléments.
- `'a'` est l'argument qui indique que nous créons une copie en ajout ('a' comme 'append')

Manipulations à effectuer :

1. Ouvrir en ajout le fichier "resultats.txt" : on nommera `s` la copie python.

Remarque 5. Lorsqu'on souhaite ajouter des informations à un fichier bitmap, on utilise l'argument `'ab'` au lieu de `'a'` (vu en deuxième année!).

• ECRITURE en AJOUT :

L'écriture dans un fichier se fait de nouveau grâce à la méthode : `f.write()`
Mais cette fois, le curseur d'écriture est placé à la fin du fichier afin de ne pas écraser les informations déjà contenues dans le fichier.

Manipulations à effectuer :

1. taper l'instruction : `s.write("Où va se rajouter cette phrase?")`
   Vérifier si le fichier "resultats.txt" contient la phrase précédente.

2. taper l'instruction : `s.close()`
   Vérifier si le fichier "resultats.txt" contient la phrase précédente.
   Est-elle placé dans le fichier comme un nouveau paragraphe ?

`Exemple 4.` Pour ajouter un nouveau nombre premier à la liste des nombres premiers précédents :

```
f = open('premiers.txt','a')   # Création en ajout de l'objet représentant le fichier
f.write(', 47')                # Ajout de ', 47' au contenu du fichier
f.close()                      # Enregistrement des résultats
```

Vérification :
On peut, bien sûr, ouvrir le fichier "premiers.txt" et vérifier que 47 y a bien été ajouté.
Comment vérifier l'efficacité de la manipulation précédente sans ouvir le fichier "premiers.txt" ?

4 Exemple : Elimination des "e" dans un texte

1) Procédure d'élimination des "e" dans une chaine de caractères :

Les chaînes de caractères sont des itérables immuables. Nous sommes donc obligés dans le programme suivant, de créer une nouvelle chaîne de caractères pour stocker notre résultat.

```
def modvoy(chaine):
    n = len(chaine)
    L = ""
    for k in range(0,n-1) :
        if chaine[k] != 'e':
            L = L + chaine[k]
    return L
```

2) Procédure récupérant un texte dans un fichier avant d'en éliminer les "e" et d'enregistrer le résultat dans un autre fichier :

```
def destroy(nom_du_fichier) :
    f = open(nom_du_fichier,'r')        # Ouverture du fichier à traiter
    s = open('resultat.txt','w')        # Création du fichier de sortie
    texte = f.readlines()               # Récupération des données sous la forme d'une liste
    for l in texte :                     # Parcours des données
        L = modvoy(l)            # Elimination des voyelles de la ligne Lk
        s.write(L + '\n')               # Enregistrement du résultat dans le fichier de sortie
    f.close()
    s.close()
```

Vérifiez que le fichier "resultat.txt" contient bien le résultat attendu.

5 Exercices

Exercice : 1
Déterminer une fonction `compte(T)` permettant de calculer le nombre de caractères contenus dans un fichier texte T.

Exercice : 2
Déterminer une fonction `fusion(i,j,F)` qui permet de fusionner les colonnes C_i et C_j d'un fichier excel F.