TD6

February 17, 2020

1 Lire et écrire un fichier

Dans ce TD, nous allons voir comment ouvrir, lire et écrire un fichier. Ce TD requière que le fichier *fable.txt* soit placé dans un répertoire nommé *fichiers*.

1.1 Ouvrir un fichier (`open()`)

Pour ouvrir un fichier, on utilise la fonction `open()` en lui indiquant le chemin (relatif ou absolu) du fichier ainsi que le mode d'ouverture :

- `r` pour *read* : le fichier sera accessible en lecture seule
- `w` pour *write* : le fichier sera ouvert en écriture et le contenu sera écrasé.
- `a` pour *append* : le fichier sera ouvert en écriture. L'écriture se fera en fin de fichier et le contenu ne sera pas perdu
- `b` pour *binary* : cette option peut s'ajouter au précédente. Elle permet de spécifier que le fichier est un fichier *binaire*. Nous y reviendrons plus tard.

L'appel de la fonction se fait de la façon suivante :

```
In [59]: mon_fichier = open("fichiers/fable.txt","r") # ouverture en mode lecture seule
         type(mon_fichier)

Out[59]: _io.TextIOWrapper
```

La fonction `open()` retourne un objet de type `TextIOWrapper`. Même si nous ne regarderons pas en détail ce type d'objet, nous allons voir comment s'en servir. La fonction associée `read()` retourne le contenu du fichier sous forme d'un *gros* `str`.

```
In [60]: contenu = mon_fichier.read()
         type(contenu)

Out[60]: str
```

On peut donc utiliser tout ce que l'on sait sur les `str`.

```
In [61]: print(contenu)
```

```
Maître Corbeau, sur un arbre perché,
Tenait en son bec un fromage.
Maître Renard, par l'odeur alléché,
Lui tint à peu près ce langage :
Et bonjour, Monsieur du Corbeau,
Que vous êtes joli ! que vous me semblez beau !
```

1.2 Fermer un fichier (`close()`)

Pour fermer un fichier ouvert, on utilise la fonction associée `close()` sur l'objet de type `TextIOWrapper`.

```
In [62]: mon_fichier.close()
```

1.3 Ecrire des `str` dans un fichier (`write()`)

Pour écrire dans un fichier, il faut tout d'abord l'ouvrir. On peut ouvrir un fichier existant, mais aussi ouvrir un fichier qui n'existe pas encore. Dans ce cas il sera créé.

```
In [63]: mon_fichier = open("fichiers/nouveau.txt","w") # création du fichier nouveau.txt
```

On peut alors ajouter écrire du texte dans le fichier sous forme de `str` avec la fonction associée `write()` :

```
In [64]: mon_fichier.write(contenu)
         mon_fichier.write("Sans mentir, si votre ramage")
         mon_fichier.write("Se rapporte à votre plumage,")
         mon_fichier.write("Vous êtes le Phénix des hôtes de ces bois.")

Out[64]: 42
```

La fonction `write()` renvoie le nombre de caractères ajoutés. Ici le `42`correspond à la dernière commande `write()`.

Il ne reste plus qu'à fermer le fichier.

```
In [65]: mon_fichier.close()
```

Vous pouvez vérifier que dans le répertoire fichiers, le fichier *nouveau.txt* a été créé et qu'il contient le texte précédent. Vous pouvez utiliser la commande shell `cat` pour afficher le contenu du fichier.

1.4 Fonctions associées aux `str`

Jusqu'à présent, nous n'avons pas vraiment regardé les fonctions associées aux `str`. La lecture et l'écriture de `str` dans un fichier est l'occasion de revenir sur plusieurs fonction qui peuvent être utiles. Nous ne serons pas exhaustif. N'hésitez pas à chercher sur internet…

Avant d'aller plius loin, Nous rappelons que les chaînes de caractères sont des listes. Vous pouvez donc utiliser toutes les méthodes que nous avons vues dans le TD 5.

1.4.1 Fonctions simples :

```
In [77]: texte = "  mon TEXTE  "
         texte.lower() # met tout en minuscule

Out[77]: '  mon texte  '

In [78]: texte.upper() # met tout en majuscule

Out[78]: '  MON TEXTE  '

In [79]: texte.capitalize() #  met une majuscule en début de phrase et le reste en minuscule

Out[79]: '  mon texte  '

In [80]: texte.strip() # retire les expaces en début et fin de chaîne

Out[80]: 'mon TEXTE'

In [87]: texte.find("TEXTE") # cherche une chaîne de caractères
                             # et renvoie l'index du début de la chaîne (ici 6).
         texte[6]

Out[87]: 'T'

In [70]: texte = "La la la la la !!!"
         texte.replace("la","ho") # remplace une chaîne par une autre

Out[70]: 'La ho ho ho ho !!!'

In [71]: texte = "La la la la la !!!"
         texte.replace("la","ho",2) # remplace une chaîne par une autre,
                                    # un nombre de fois spécifié

Out[71]: 'La ho ho la la !!!'
```

1.4.2 Fonction associée `format()`

Cette fonction est très puissante. Elle permet de créer facilement des chaînes de caractères dynamique. Lors de la création de la chaîne de caractère, on place des *labels* entre {}qui seront remplacés par des valeurs spécifiées dans la fonction `format()`. Ok, regardons un exemple, ce sera plus parlant :

```
In [72]: texte = "Je m'appelle {prenom} et j'ai {age} ans." # deux labels {prenom} et {age} sont
         print(texte) # on peut afficher la chaîne précédente.

Je m'appelle {prenom} et j'ai {age} ans.


In [73]: texte.format(prenom="Thomas",age=20) # la fonction format remplace ici les balises par
```

```
Out[73]: "Je m'appelle Thomas et j'ai 20 ans."
```

Notez qu'ici la variable `texte` *n'est pas modifiée :*

```
In [74]: print(texte)

Je m'appelle {prenom} et j'ai {age} ans.
```

Si l'on souhaite modifié la variable `texte` de façon définitive, on peut écrire :

```
In [75]: texte = texte.format(prenom="Thomas",age=20)
         print(texte)

Je m'appelle Thomas et j'ai 20 ans.
```

1.5 Exercice 1 : Tableaux périodiques

Atomes = [["Fer",26],["Ag",47],["Ca",20],["Al",13],["Ne",10],["O",8],["Au",79]]

1) Ecrire un programme qui parcourt la liste précédente et affiche pour chaque élément : "L'élément XXX a pour numéro atomique YYY.".
2) Modifier ce programme pour que le texte affiché soit maintenant sauvegardé dans un fichier.

1.6 Problème 1 : Fichier codé

Récupérer le fichier *code.txt* et placer un sous répertoire *fichiers* dans votre répertoire de travail.

Ce fichier est codé. Il va falloir le décoder. Le code est le suivant : - les chiffres 0,1,2,3,4,5,6,7,8,9 replacent respectivement a,c,e,i,l,n,o,r,s,t - Chaque caractère (espace compris) a été échangé avec son voisin, exemple : *"Le train arrive."* -> *"eLt arnia rrvi.e"*

1) Ouvrir le fichier et afficher le texte qu'il contient
2) Décoder le code

```
In [ ]:
```

1.7 Ecrire des objets dans un fichier (`pickle`)

Il est également possible d'enregistrer des *objets* comme des listes dans des fichiers et de les récupérer plus tard. Pour cela, nous allons utiliser la librairie `pickle`.

```
In [16]: import pickle
```

Comme précédemment, on ouvre en écriture (`w`) le fichier que l'on veut créer en ajoutant l'option `b` pour préciser que le fichier sera au format bianaire.

Le fichier ne sera donc pas lisible par un humain, moais l'ordinateur pour y mettre des informations supplémaires pour y stocker des objets.

Une fois le fichier ouvert, on utilise la fonction associée `pickle.dump(objet,fichier)` pour ajouter un objet dans le fichier. Il est possible d'ajouter plusieurs objets. Il ne reste plus qu'à fermer le fichier.

```
In [17]: fichier = open("fichiers/data.bin","wb")

         Atomes = [["Fer",26],["Ag",47],["Ca",20],["Al",13],["Ne",10],["O",8],["Au",79]]
         Nombre = [1,2,3,4]

         pickle.dump(Atomes,fichier)
         pickle.dump(Nombre,fichier)

         fichier.close()
```

Pour récupérer plus tard, ce que nous avons mis dans le fichier, il faut réouvrir le fichier avec les options `rb`, puis charger un à un les objets sauvegardés.

```
In [18]: fichier = open("fichiers/data.bin","rb")

         Atomes = pickle.load(fichier)
         Nombre = pickle.load(fichier)

         print(Atomes)
         print(Nombre)
```

```
[['Fer', 26], ['Ag', 47], ['Ca', 20], ['Al', 13], ['Ne', 10], ['O', 8], ['Au', 79]]
[1, 2, 3, 4]
```

Noter que **vous devez savoir ce qu'il y a dans le fichier**. S'il y a deux objets et que vous en chargez trois, il y aura une erreur:

```
In [20]: Toto = pickle.load(fichier)
```

```
         ---------------------------------------------------------------------------

         EOFError                                  Traceback (most recent call last)

         <ipython-input-20-fb57e1700de5> in <module>
    ----> 1 Toto = pickle.load(fichier)


         EOFError: Ran out of input
```

N'oubliez pas de fermer le fichier.

```
In [21]: fichier.close()
```

1.8 Exercice 2 : PIB par pays

Le fichier *PIB.bin* contient une liste d'élément. Chaque élément est constitué du nom d'un pays, de son PIB par habitant et de son nombre d'habitants.

1) Charger le fichier, récuperer la liste.
2) Afficher à l'aide d'un nuage de point, le PIB par habitant en fonction du nombre d'habitants.
3) Calculer le PIB total de chaque pays. Quel Pays a le PIB total le plus important ?

```
In [ ]:
```

1.9 Problème 2 : Chûte libre

Un avions lâche une caisse de matériel d'une altitude H et une vitesse initiale horizontale $\vec{v_0}$. Nous allons étudier la trajetoire de la caisse.

Si l'on néglige les frottements, la trajectoire s'obtient à partir du principe fondamentalre de la dynamique. Ici il n'y a que le poids $\vec{p}$ qui agit donc :

- $a_x = 0$
- $a_z = -g$
- $v_x = v_0$
- $v_z = -g\,t$
- $x = v_0\,t$
- $z = -1/2g\,t^2 + H$

On prendra $H = 10000\,\text{m}$, $g = 9.81\,\text{m.s}^{-2}$ et $v_0 = 100\,\text{m.s}^{-1}$

1) Tracer la trajectoire jusqu'au sol, c'est à dire z en fonction de x.
2) Cette chûte a été enregistrée par une caméra. Le fichier *chute.bin* contient la trajectoire enregistrée sous forme de deux listes : la première correspond à x, la seconde à z. Tracer un même schéma la trajectoire enregistrée et celle calculée précedemment.
3) D'où provient la différence observée ?

```
In [ ]:
```