fiche_python2

September 10, 2019

1 Les listes par compréhension, la méthode `format` et la lecture/écriture de fichiers.

1.1 Énumération

1. Comment afficher `[1,2,3,4,5,6,7,8,9]` ? et la même chose jusqu'à 90 ?
2. Comment afficher `[1,3,5,7,9]` ? et la même chose jusqu'à 90 ?
3. Comment afficher `[1,4,9,16,25,36,49,64]` ?
4. Comment afficher `[1,2,4,8,16,32,64,128,256,512,1024]` ?
5. Comment obtenir la liste des caractères d'une chaîne de caractère ?

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

1.2 Format

`format` est une méthode permettant d'afficher des variables dans une chaîne de caractère.

[165]:
```
a = 1
b = 'une'
c = True

' Voici {} exemple pour agrémenter {} chaîne. {} or False ?'.format(a,b,c)
```

[165]:
```
' Voici 1 exemple pour agrémenter une chaîne. True or False ?'
```

Amusez-vous avec cet exemple

1.3 Créer et écrire dans un fichier

On est souvent amené à créer et modifier des fichiers. Cela peut se faire très simplement en Python

[183]:
```
fic = open('Mon_1er_Fichier.txt', 'r+')
```

`r` pour *read* et le `+` pour ajouter les droits d'écriture (*write*).
On aurait pu faire le contraire : `w+`
Ou seulement `w` comme on le comprendra plus bas.
On peut se demander quel est le type de cet objet `fic`:

[184]:
```
type(fic)
```

[184]:
```
_io.TextIOWrapper
```

Comme chez McDo...
Que signifie IO ?

[185]:
```
fic.write("J'écris dans un fichier.\nJe suis heureux(se)")
```

[185]:
```
44
```

Pourquoi 44 ?
Regardons `fic` :

[186]:
```
fic
```

[186]:
```
<_io.TextIOWrapper name='Mon_1er_Fichier.txt' mode='r+' encoding='UTF-8'>
```

[187]:
```
fic.write(' et je continue')
```

[187]:
```
15
```

Essayons alors de lire le fichier avec la méthode `read` :

[171]:
```
contenu = fic.read()
```

[172]:
```
print(contenu)
```

Rien ! En fait, après chaque écriture, le curseur de lecture est en fin de fichier et il faut le remettre au début avec un autre `open`

[188]:
```
fic = open('Mon_1er_Fichier.txt', 'r')
```

[189]:
```
fic.read()
```

[189]:
```
"J'écris dans un fichier.\nJe suis heureux(se) et je continue"
```

[190]:
```
fic.read()
```

[190]:
```
''
```

Le curseur est à nouveau à la fin...
Mais si seul la création du fichier nous intéresse, l'essentiel est fait : un fichier `Mon_1er_Fichier.txt` a été créé dans le répertoire courant et on peut voir son contenu en l'ouvrant.
À la fin, il ne faut pas oublier de fermer le fichier avec `close`:

[192]:
```
fic.close()
```

[193]:
```
fic.read()
```

```
␣
↪---------------------------------------------------------------------------

       ValueError                                Traceback (most recent call␣
↪last)

       <ipython-input-193-4c3760645c61> in <module>
   ----> 1 fic.read()


       ValueError: I/O operation on closed file.
```

Il y a une méthode plus pythonesque de procéder avec `with` qui fermera le fichier automatiquement:

[194]:
```
with open('Mon_1er_Fichier.txt', 'r+') as fic:
    fic.write('Ligne 1\nLigne2\Ligne3')
```

[195]:
```
fic.read()
```

```
␣
↪---------------------------------------------------------------------------

       ValueError                                Traceback (most recent call␣
↪last)

       <ipython-input-195-4c3760645c61> in <module>
   ----> 1 fic.read()


       ValueError: I/O operation on closed file.
```

1.4 Un exercice bilan

Créez un fichier qui contiendra 1000 lignes, chacune contenant le mot ligne et le numéro de la ligne:

```python
Ligne 1
Ligne 2
Ligne 3
.
.
.
Ligne 1000
```

Trois lignes suffisent...

[ ]: