Dépôt BLIH: PYjour01
Répertoire de rendu : c'est précisée pour chaque exos.

Vous avez besoin du module pyrser :

Installez le package :

```
$> sudo pip install pyrser
```

si vous avez un python 2.x et 3.x sur votre machine faites un **pip3**. Suivant la configuration vous pouvez aussi avoir à rajouter le flags --pre.

Regardez la documentation ici : http://pythonhosted.org/pyrser.

LISEZ LE TUTORIAL 1 : A Guided JSON PARSER !

Si vous appréciez le paquet mettez un « I Use This » sur https://www.openhub.net/p/pyrser

ou suivez son activité sur https://github.com/LionelAuroux/pyrser (une petite étoile ou un watch).

pythonBasic

Dans un sous répertoire **pythonBasic**

Faire un module « bidon.py », contenant :

1. -une classe Bidon :
    1. contenant une variable de classe zaz aillant pour valeur « je suis un pro du python ».
    2. se construisant comme dans les cas suivant :

```
iopi$ python3.3
Python 3.3.x
[GCC x.x.x] on linux
Type …
>>> import bidon
>>> a = bidon.Bidon(«cool»)
>>> vars(a)
{'txt' : 'cool', 'num': 42}
>>> a = bidon.Bidon(«heu», 666)
>>> a.num
666
>>> a = bidon.Bidon(«keywords var», 1024, le=4, nain=5, a=6, chausette=7)
>>> print(a.le+a.nain+a.a+a.chausette)
22
```

2. -contenant une fonction var2listsort.

```
iopi$ python3.3
Python 3.3.x
[GCC x.x.x] on linux
Type …
>>> import bidon
>>> print(bidon.var2listsort(3, 1, 7, 2))
[1, 2, 3, 7]
>>> print(bidon.var2listsort(40.3, 40.0, 39.8, 38.7, 39.9, 40.1, 38.80,
38.69))
[38.69, 38.7, 38.8, 39.8, 39.9, 40.0, 40.1, 40.3]
```

Pour réaliser ses exercices, référez vous à la documentation python :

http://docs.python.org/3/tutorial/index.html
http://docs.python.org/3/library/

plus particulièrement sur :

class
setattr
default parameter

*var
**kwvar
dict
list

ParserPseudoIni

Dans un sous répertoire **parserPseudoIni**

Faire un module « parserPseudoIni.py » capable de lire un fichier comme décrit dans la BNF suivante.

```
Ini = [ Section+ eof ]

Section = [ '[' id ']' ClefValeur+ ]

ClefValeur = [ id '=' valeur ]

valeur = [ id | num | string ]
```

Cette BNF est une base de travail. Elle doit être retouchée car elle n'est pas fonctionnelle dans l'état.
Pyrser utilise une syntaxe de BNF très légèrement différente de celle du précédent jour.
Basiquement, '::=' est transformé en simple '=' et il n'y a pas de terminateur ';' mais seulement un groupe '[]'.

Ce script doit construire un arbre en mémoire tel que :

```
import sys
from parserPseudoIni import *

ini = parserPseudoIni()
res = ini.parse_file(sys.argv[1])

for section_name, section_values in res.sections.items():
    for key, value in section_values.items():
        print("variable [%s] %s = %s" % (section_name, key, value))
```

Affiche :

```
$>python3.3 testini.py test.ini
variable [log] file = "/var/log/pifou"
variable [log] LEVEL = ALL
variable [session] debug = 1
variable [session] command = "ls -l"
$>
```

Avec le fichier suivant en entrée :

```
[session]
debug=1
command = "ls -l"

[log]
LEVEL = ALL
file = "/var/log/pifou"
```

ParserCSV

Dans un sous répertoire **parserCSV**

Faire un module python « parserCSV.py » contenant la grammaire capable de lire un fichier CSV (fichier de sortie possible d'excel) tel que celui-ci :

```
iopi$> cat -e starter.csv
A;B;C;D$
E;F;G$
```

Commencez par cet exemple simple. Dans tous les exemples et dans la correction les données à parser séparé par des ';' sont des identifiants (règle id).
Puis faites en sorte de pouvoir parser des choses plus complexes celui-ci :

```
iopi$> cat -e test.csv
test1;test2;test3;test4$
test5;test6;test7;test8;$
test9;;test11;;$
;;;test3;;coucouiopi$>
```

Le script doit stocker en mémoire tel que si nous écrivons le script suivant dans un fichier « test.py » :

```
import sys
from parserCSV import *

csv = parserCSV()
res = csv.parse_file(sys.argv[1])
for line in res.lines:
    for word in line:
        print(word)
print(res.lines[3][3])
```

Et que nous l'exécutions par exemple via la commande :

```
iopi$>python3.3 testcsv.py test.csv |cat -e
test1$
test2$
test3$
test4$
test5$
test6$
test7$
test8$
$
test9$
$
test11$
$
$
$
$
$
test3$
$
coucou$
test3$
iopi$>
```

Le but de l'exercice est de vous faire trouver l'algorithme de «parsing de liste », comprendre la différence entre un 'séparateur' et un 'terminateur', et finalement voir que 'eof' ne se traite pas de n'importe quel manière.