Tools Python

INITIATION PYTHON, PREMIÈRE APPROCHE ET PARSING

Réalisé par : Aiguier Maxime

SOMMAIRE

INTRODUCTION

Objectif

- Apprendre les bases du parsing en Python
- S'entrainer sur un exercice type

Languages

- Python

INITIATION AU PYTHON

Les variables

Première chose importante à savoir à propos du Python: il s'agit d'un language non-typé, c'est à dire qu'il n'est pas nécessaire de spécifier le type d'une variable lors de sa déclaration, python se chargera tout seul de déduire son type.

```
# ------ variables ------
varString = "toto"
varNumber = 42
varBool = False
varList = ["toto", "tata", "tonton"]
varTuple = (2, "toto")
varDict = {"surname": "Jean", "name": "Dupont"}
```

Nous allons ici parler de trois types de variables qui peuvent vous sembler inconnus: la Liste, le Tuple et le Dictionnaire.

- **Liste**: Une liste est ce qui s'apparente le plus à une liste chainée en C (ensemble de valeurs qui se suivent et se référencent entre elles). Elle est moins gourmande en mémoire et plus flexible que les tableaux standards.
- **Tuple**: Celui-ci est identique à une liste à l'exception que ses valeurs ne peuvent pas être modifiées. Il est encore moins gourmand en mémoire qu'une liste et est donc employé lorsque l'on sait que les valeurs qu'il contient ne changeront pas.
- **Dictionnaire**: Un dictionnaire agit de la même façon que ce que l'on appelle dans la Programmation Orientée Object (POO) un Object. C'est un ensemble de « clé-valeur ». Nous l'aborderons plus en détail tout à l'heure.

L'affichage

En python, la fonction «print » nous permet d'afficher des variables, des strings, des retours de fonctions,…

Le paramètre « end » permet de définir le caractère de fin d'affichage. De base, ce dernier est un « \n » , ce qui équivaut à un saut à la ligne.

```
# ------ display ------
surname = "Jean"
name = "Dupont"
age = 42
print(surname + " " + name) # Jean Dupont
print("%s à %d ans" % (surname, age)) # Jean à 42 ans

print("test")
print("test 2")
# test
# test 2

print("test", end=", ")
print("test 2")
# test, test 2
```

Les boucles

Les boucles permettent d'effectuer une actions jusqu'à ce que celle-ci soit interrompus (via une méthode break ou un return), ou alors jusqu'à ce que sa condition de boucle ne soit plus valide.

En Python, l'indentation est **PRI-MOR-DIABLE**. Un code mal indenté est un code non

```
# ------ boucles ------
i = 0
while (i < 5):
    print(i, end=" ")
    i += 1
# 0 1 2 3 4

varList = ["toto", "tata", "tonton"]
for value in varList:
    print(value, end=" ")
# toto tata tonton

for index, value in enumerate(varList):
    print("index: %d | value: %s" % (index, value))
# index: 0 | value: toto
# index: 1 | value: tata
# index: 2 | value: tonton
```

fonctionnel. Ainsi, les boucles (et les conditions) prennent « : » à la fin de celles-ci. De plus, La fonction enumerate permet de récupérer à la fois l'index ET la valeur d'une liste.

Les listes

Voici quelques syntaxes utiles pour manipuler des listes.

```
# ------ list ------
varList = ["toto", "tata", "tonton"]
print(len(varList)) # 3
print(varList[1]) # tata
print(varList[-1]) # tonton

varList = [0] * 5
print(varList) # [0, 0, 0, 0, 0]

varList = [x for x in range(5)]
print(varList) # [0, 1, 2, 3, 4]

varList = []
varList.append("toto")
varList.append("tata")
print(varList) # ["toto", "tata"]
varList.extend(["tonton", "maman"])
print(varList) # ["toto", "tata", "tonton", "maman"]
varList = ["titi"] + varList
print(varList) # ["titi", "toto", "tata", "tonton", "maman"]
varList.insert(4, "kikou")
print(varList) # ["titi", "toto", "tata", "tonton", "kikou", "maman"]

newList = varList[1:4]
print(newList) # ["toto", "tata", "tonton"]
newList = varList[2:]
print(newList) # ["tata", "tonton", "kikou", "maman"]

del varList[:3]
print(varList) # [tonton", "kikou", "maman"]
popedValue = varList.pop(1)
print(popedValue) # kikou
print(varList) # ["tonton", "maman"]
```

Les dictionnaires

De même pour les dictionnaires.

```
# ------ dict ------
varDict = {"key1": "valeur1", "key2": "valeur2"}
print(varDict["key1"]) # valeur1

print(list(varDict.values())) # ["valeur1", "valeur2"]
print(list(varDict.keys())) # ["key1", "key2"]
print(list(varDict.items())) # [("key1": "valeur1"),
                                ("key2": "valeur2")]
```

Leçon de Parsing

Le but final de cette introduction au Python est de savoir parser le plus rapidement et le plus efficacement possible un fichier contenant les informations voulues. Nous allons donc partir d'un fichier contenant des informations diverses sur des utilisateurs et nous souhaitons les récupérer sous forme d'une liste de dictionnaires.

Tout d'abord, il nous faut ouvrir et lire un fichier. Voici comment procéder:
Vous noterez que la première ligne contient le nom de l'information correspondante.

```
# ------ parsing ------
fo = open("toto.txt")
readData = fo.read()
print(readData)
# surname,name,age,address,city,country
# Jean,Dupont,42,3 rue Dupont,Vesoul,France
# Anne,Dupont,32,3 rue Dupont,Vesoul,France
# Léos,Julien,20,18 rue Tucpa,Montpellier,France
# Aiguier,Maxime,12,8 avenue de la gare,Montpellier,France
```

Nous allons ensuite utiliser la fonction « split » (je vous laisserez vous renseigner sur sa compagne « join ») afin de parser le bloc de texte et d'en récupérer une liste. Pour cela, il suffit de lui fournir en paramètre le « délimiteur » avec lequel l'on souhaite parser le texte.

```
users = readData.split('\n')
print(users)
# [
#   'surname,name,age,address,city,country',
#   'Jean,Dupont,42,3 rue Dupont,Vesoul,France',
#   'Anne,Dupont,32,3 rue Dupont,Vesoul,France',
#   'Léos,Julien,20,18 rue Tucpa,Montpellier,France',
#   'Aiguier,Maxime,12,8 avenue de la gare,Montpellier,France'
# ]
```

Initialisons maintenant la liste sensée contenir les dictionnaires d'utilisateurs. Parsons également la première ligne afin de stocker à part les noms des informations.

```
usersParsed = []
userKeys = users.pop(0).split(', ')
print(userKeys)
# ['surname', 'name', 'age', 'address', 'city', 'country']
```

Vient ensuite le bout de code qui va nous permettre de transformer le texte en liste de dictionnaire. Prenez bien le temps de le lire et de le comprendre car il peut être assez complexe à première vue.

Affichons le résultat et voila !

```
for user in users:
    user = user.split(',')
    newUser = {}
    for index, data in enumerate(user):
        newUser[userKeys[index]] = data
    usersParsed.append(newUser)
```

```
print(usersParsed)
# [
#   {
#       'surname': 'Jean',
#       'name': 'Dupont',
#       'age': '42',
#       'address': '3 rue Dupont',
#       'city': 'Vesoul',
#       'country': 'France'
# }, {
#       'surname': 'Anne',
#       'name': 'Dupont',
#       'age': '32',
#       'address': '3 rue Dupont',
#       'city': 'Vesoul',
#       'country': 'France'
# }, {
#       'surname': 'Léos',
#       'name': 'Julien',
#       'age': '20',
#       'address': '18 rue Tucpa',
#       'city': 'Montpellier',
#       'country': 'France'
# }, {
#       'surname': 'Aiguier',
#       'name': 'Maxime',
#       'age': '12',
#       'address': '8 avenue de la gare',
#       'city': 'Montpellier',
#       'country': 'France'
#   }
# ]
```