TP python - Structures de bases

Gaël Guibon

29 novembre 2016

Sommaire

Chapitre 1

Python en un coup d'oeil

1.0.1 Philosophie python : clarté, simplicité

The Zen of Python (TimPeters) :
https://www.python.org/dev/peps/pep-0020/
«

— Beautiful is better than ugly.
— Explicit is better than implicit.
— Simple is better than complex.
— Complex is better than complicated.
— Flat is better than nested.
— Sparse is better than dense.
— Readability counts.
— Special cases aren't special enough to break the rules.
  — Although practicality beats purity.
— Errors should never pass silently.
  — Unless explicitly silenced.
— In the face of ambiguity, refuse the temptation to guess.
— There should be one– and preferably only one –obvious way to do it.
  — Although that way may not be obvious at first unless you're Dutch.
— Now is better than never.
  — Although never is often better than right now.
— If the implementation is hard to explain, it's a bad idea.
— If the implementation is easy to explain, it may be a good idea.
— NameSpaces are one honking great idea – let's do more of those !

»

1.0.2 Python, un langage interprété

Contrairement à C++ ou Java, Python est un langage interprété. En voici les principales différences :

Compilé	**Interprété**
Tout le programme est compilé	Chaque instruction est interprétée
Code intermédiaire généré	Pas de code intermédiaire
Instructions de contrôle conditionnelles plus rapide	Instructions de contrôle conditionnelles plus lentes
Plus de mémoire requise	Moins de mémoire requise
Pas besoin de recompiler à chaque fois	Recompilé chaque fois qu'il est interprété
Erreurs au niveau du programme	Erreurs au niveau de l'instruction)
Java	Python

1.0.3 Ecosystème Python

Beaucoup de librairies python existent et permettent de traiter de nombreux cas. Voici un exemple de librairies python très utiles et reconnues :

Nom	**Lien**
Interfaces graphiques	
PyGTK	http://www.pygtk.org/
Traitement Automatique du Langage	
NLTK	http://www.nltk.org/
Formats de fichiers	
LXML	http://lxml.de/
Beautiful soup	https://www.crummy.com/software/BeautifulSoup/
Json	https://docs.python.org/2/library/json.html
Apprentissage / calculs	
NumPy	http://www.numpy.org/
SciPy	http://www.scipy.org/
Scikit-Learn	http://scikit-learn.org/
TensorFlow	https://www.tensorflow.org/
Base de données	
PyMongo	https://github.com/mongodb/mongo-python-driver
Application Web	
Django	https://www.djangoproject.com/
Parallelisme	
Thread	https://docs.python.org/2/library/threading.html

TABLE 1.1 – Sélection non exhaustive de librairies python

Chapitre 2

Lancer votre installation python en local

— Lancer ubuntu et se connecter avec vos identifiants
— Récupérer les documents de la clef USB
— Dans le terminal (CTRL + ALT + T) taper :

```
bash install.sh
```

— Lancer ipython avec la commande suivante :

```
~/.local/bin/ipython
OU
~/.local/bin/ipython notebook
```

Chapitre 3

Syntaxe de base

3.0.4 Variables

Python est un langage à typage dynamique.

Déclarer une variable en java :

```
int myInt = 0;
System.out.println(myInt);
```

Déclarer une variable en python :

```
myInt = 0
print myInt
```

3.0.5 Indentation

Les structures sont délimitées par l'indentation. *i.e.* **les tabulations remplacent les accolades**.
Boucle for en java :

```
for(String line : lines){
  System.out.println(line);
}
```

Boucle for en python :

```
for line in lines :
  print line
```

Chapitre 4

Structures de données

4.0.6 Afficher une variable

```
str_entree = raw_input()
#taper un mot
print str_entree
```

4.0.7 Les listes

1. Définir la liste : liste =[17, 38, 10, 25, 72], puis effectuez les actions suivantes [1] :

   — triez et affichez la liste ;
   — ajoutez l'élément 12 à la liste et affichez la liste ;
   — renversez et affichez la liste ;
   — affichez l'indice de l'élément 17 ;
   — enlevez l'élément 38 et affichez la liste ;
   — affichez la sous-liste du 2ème au 3ème élément ;
   — affichez la sous-liste du début au 2ème élément ;
   — affichez la sous-liste du 3ème élément à la fin de la liste ;
   — affichez la sous-liste complète de la liste ;

2. Ecrire une fonction récursive "palindrome(s)" qui renvoie *True* si la chaine de caractères *s* est un palindrome, *False* sinon. NB : un palindrome est une chaine de caractères qui peut se lire indifféremment dans les deux sens. [2]

```
#initialisation de liste
ma_liste = []
```

4.0.8 Les tuples

Les tuples sont différents des listes :
— Ils sont plus rapides
— On ne peut y ajouter ou enlever une valeur. Ils sont non-mutables.
— Ils n'ont pas de méthode (append, remove, etc)
Reprendre la fonction du palindrome en utilisant uniquement des tuples. Cette fonction retournera désormais un tuple contenant la valeur booléenne, et le mot associé.

```
#initialisation de tuple
mon_tuple = ()
```

1. Exercice issu du cours de l'iut d'Orsay.
2. Exercice de Alain Samuel et Sébastien Mavroumatis.

Named tuples (optionnel) :
Modifier la fonction en retournant un namedtuple, puis parcourir les champs du namedtuple.
Exemple d'initialisation d'un namedtuple :

```
import collections
results = collections.namedtuple('Results', ['xtrain', 'ytrain', 'xtest', 'ytest'])
```

4.0.9 Le JSON

En utilisant la librairie json (https://docs.python.org/2/library/json.html?highlight=json#module-json), lire le fichier JSON fourni. Vérifier si pour chaque mot du champ "content" il s'agit ou non d'un palindrome. Attribuer un nouveau champ "palindrome" à l'objet "mot" afin d'indiquer si oui ou non il s'agit d'un palindrome. Enfin, réécrire la liste d'objet json en un fichier "mots_checked.json".

```
import json
```

4.0.10 Les dictionnaires

Lire le json et intégrer son contenu dans un dictionnaire (clé = champ content ; valeur = champ palindrome). Trier ce dictionnaire en fonction des valeurs et afficher en premier tous les mots qui sont bel et bien des palindromes.

TIP

Les dictionnaires ne peuvent être triés directement ! Les tuples et les listes seront utiles ici...

4.0.11 Les matrices

— Générer une matrice à l'aide la librairie numpy (http://www.numpy.org/) :

```
import numpy as np
a = np.arange(15).reshape(3, 5)
```

— Afficher la taille, la forme, le nombre de dimensions et le nombre d'items de la matrice
— Multiplier les matrices et afficher le produit élément par élément

4.0.12 Les CSV

Transformer le JSON complet en un fichier CSV. Utilisez la librairie csv (https://docs.python.org/2/library/csv.html) pour cela.

```
import csv
```

4.0.13 Le XML

Transformer le CSV complet en un fichier XML à l'aide de la librairie LXML (http://lxml.de/).

```
from lxml import etree
from copy import deepcopy
```

Chapitre 5

Web service

5.0.14 Objectif

Lier un programme python lancant du clustering et une interface angular Js avec un service web en python.

Le programme se compose de plusieurs parties :

1. Interface Web
2. Service de requête d'image sur Pixabay (`https://pixabay.com/`)
3. Moteur de regroupement (*clustering*)
4. **Service web qui les relie tous**

Un squelette de programme est fourni pour faciliter la démarche.

5.0.15 Méthode getRandomSubjects

Compléter la méthode getRandomSubjects. Elle doit permettre de visualiser des sujets pré-sélectionnés à l'initialisation de l'interface.
Cette méthode doit :

1. Récupérer des sujets au hasard dans la liste fournie
2. Pour chaque sujet :
   — Récupérer l'url de l'image en utilisant la classe imageRequest()
   — Créer un objet json avec un champ "word" contenant le sujet, et un champ "image" contenant l'url
3. La méthode retourne la liste des objets json

5.0.16 Méthode getSubject

Compléter la méthode getSubjects. Elle doit permettre de lancer tout le moteur de clustering et de récupérer pour chaque résultat une url d'image associée.
Cette méthode doit :

1. Prendre en paramètre une query au format String
2. Récupérer les sujets en utilisant la query dans la méthode start de la classe clustering_engine
3. Pour chaque sujet :
   — Récupérer l'url de l'image en utilisant la classe imageRequest() (fichier imageRequest.py
   — Créer un objet json avec un champ "word" contenant le sujet, et un champ "image" contenant l'url
4. La méthode retourne la liste des objets json

5.0.17 Méthode GET

Cette méthode doit s'occuper de gérer les paramètres du service web, la query, ainsi que l'utilisation de la classe clustering_engine.

Lien vers la documentation de web.py : http://webpy.org/docs/0.3/.

Pour cela :

1. Instancier l'objet query_params à l'aide de web.py
2. Créer une entête standard 'status' (TIP : sous forme de dictionnaire)
3. Créer une query et la gérer :
   — SI la query == "documents" ALORS attribuer defaultDocuments au champ "documents"
   — Sinon si la requête n'est pas vide, utiliser la méthode getSubjects
   — Chercher des sujets au hasard si la requête est vide (getRandomSubject)
4. La méthode doit retourner toutes ces données au format json String

5.0.18 Tester son web service

Pour tester si web service focntionne et répond aux attentes :

```
~/.local/bin/python2.7 clustering_webservice.py

# Aller dans le navigateur et taper
http://localhost:8080/?query=europe
```

5.0.19 Voir le résultat avec l'interface sur votre machine

Accès administrateurs requis !

— **Installer Xampp (exemples pour ubuntu issus de https://doc.ubuntu-fr.org/xampp)**
   1. Télécharger le fichier d'installation : http://www.apachefriends.org/fr/download.html
   2. Taper : sudo chmod 755 xampp-linux-*-installer.run
   3. Taper : sudo ./xampp-linux-*-installer.run

— **Copier le dossier "gp1" dans /opt/lampp/htdocs**

— **Lancer l'application**
   1. Lancer xampp. Taper : sudo /opt/lampp/xampp start
   2. Lancer le service web. Taper : sudo python clustering_webservice.py
   3. Dans votre navigateur, aller à l'adresse http://localhost:8888/gp1/ pour voir l'interface