Télécom Paris **MDI220**
2020 **Pavlo Mozharovskyi**

TP n° 1 : Introduction à Python

Quelques remarques avant de démarrer

On va utiliser `Jupyter notebook` durant cette séance. Pour cela choisissez la version Anaconda sur les machines de l'école. Quelques points importants à retenir :

- Chargements divers -

```
import math # importe un package
import numpy as np # importe un package sous un nom particulier
from sklearn import linear_model # importe tout un module
from os import mkdir # importe une fonction
```

- Utilisation de l'aide -

```
help(mkdir) # pour obtenir de l'aide sur mkdir
linear_model.LinearRegression? # pour obtenir de l'aide sur LinearRegression
```

- Versions de package, localisation des fonctions -

```
print(np.__version__) # obtenir la version d'un package
from inspect import getsourcelines # obtenir le code source de fonctions
getsourcelines(linear_model.LinearRegression)
```

Remark 1. *Pour tous les traitements numériques ont utilisera* ***numpy*** *(pour la gestion des matrices notamment) et* ***scipy****.*

1 Introduction à Python, Numpy et Scipy

Il est bon de parcourir les documents suivants pour obtenir plus d'aide sur Python, pour ceux qui ne sont pas encore très familier avec ce langage :

https://docs.python.org/3/tutorial/introduction.html
https://docs.scipy.org/doc/numpy/user/quickstart.html?highlight=tutorial
https://docs.scipy.org/doc/scipy/reference/tutorial/index.html

1) Écrire une fonction `nextpower` qui calcule la première puissance de 2 supérieure ou égale à un nombre n (on veillera à ce que le type de sortie soit un `int`, tester cela avec `type` par exemple).
2) En partant du mot contenant toutes les lettres de l'alphabet, générer par une opération de *slicing* la chaîne de caractère `cfilorux` et, de deux façons différentes, la chaîne de caractère `vxz`.
3) Afficher le nombre π avec 9 décimales après la virgule.

4) Compter le nombre d'occurrences de chaque caractère dans la chaîne de caractères `s="HelLo WorLd!!!"`. On renverra un dictionnaire qui à chaque lettre associe son nombre d'occurrences.

5) Écrire une fonction de codage par inversion de lettres [1] : chaque lettre d'un mot est remplacée par une (et une seule) autre. On se servira de la fonction `shuffle` sur la chaîne de caractères contenant tout l'alphabet pour associer les lettres codées.

6) Calculer $2\prod_{k=1}^{\infty}\frac{4k^2}{4k^2-1}$ efficacement (approximativement, par exemple avec $k = 1, ..., 500$). On pourra utiliser `time` (ou `%timeit`) pour déterminer la rapidité de votre méthode. Proposer une version avec et une version sans boucle (utilisant `Numpy`).

7) Créer une fonction `quicksort` qui trie une liste, en remplissant les éléments manquants dans le code suivant. On testera que la fonction est correcte sur l'exemple `quicksort([-2, 3, 5, 1, 3])` :

```
def quicksort(ll):
    """ a sorting algorithm with a pivot value"""
    if len(ll) <= 1:
        return ll
    else:
        TODO # pivot = last element of the list ll.
        less = []
        greater = []
        for x in ll:
            if x <= pivot:
                TODO # append 'x' to 'less'
            else:
                TODO # append 'x' to 'greater'
        return TODO # concatenate quicksort(less), pivot and quicksort(greater)
```

Indices : la longueur d'une liste est donnée par `len(l)` ; deux listes peuvent être concaténées avec `l1 + l2` ; `l.pop()` retire le dernier élément d'une liste.

8) Sans utiliser de boucles `for` / `while` : créer une matrice $M \in \mathbb{R}^{5\times 6}$ aléatoire à coefficients uniformes dans $[-1, 1]$, puis remplacer une colonne sur deux par sa valeur moins le double de la colonne suivante. Remplacer enfin les valeurs négatives par 0 en utilisant un masque binaire.

9) Créer une matrice $M \in \mathbb{R}^{5\times 20}$ aléatoire à coefficients uniformes dans $[-1, 1]$. Tester que $G = M^\top M$ est symétrique et que ses valeurs propres sont positives (on parle de alors de matrice définie positive). Quel est le rang de G ?

Aide : on utilisera par exemple `np.allclose, np.logical_not, np.all` pour les tests numériques.

2 Introduction à Pandas, Matplotlib, etc.

On pourra commencer par consulter le tutoriel :
http://pandas.pydata.org/pandas-docs/stable/tutorials.html

- Charger des données -

On utilise la base de données [2] **Individual household electric power consumption Data Set**. Pour cela utiliser les commandes ci-dessous :

```
from os import path
import pandas as pd
import urllib
import zipfile
```

1. aussi connu sous le nom de code de César.
2. https://archive.ics.uci.edu/ml/datasets/Individual+household+electric+power+consumption

```
import sys

url = 'https://archive.ics.uci.edu/ml/machine-learning-databases/00235/'
filename = 'household_power_consumption'
zipfilename = filename + '.zip'
Location = url + zipfilename
# testing existence of file:
if sys.version_info >= (3, 0):
    if not(path.isfile(zipfilename)):
        urllib.request.urlretrieve(Location, zipfilename)
else:
    if not(path.isfile(zipfilename)):
        urllib.urlretrieve(Location, zipfilename)
# unzip part
zip = zipfile.ZipFile(zipfilename)
zip.extractall()
na_values = ['?', '']
fields = ['Date', 'Time', 'Global_active_power']
df = pd.read_csv(filename + '.txt', sep=';', nrows=200000,
na_values=na_values, usecols=fields)
```

On ne s'intéresse dans un premier temps qu'à la grandeur `Global_active_power`.

1) Charger la base puis détecter et dénombrer le nombre de lignes ayant des valeurs manquantes.
2) Supprimer toutes les lignes avec des valeurs manquantes.
3) Utiliser `to_datetime` et `set_index` pour indexer le DataFrame (on prendra garde au format des dates internationales qui diffère du format français).
4) Afficher le graphique des moyennes journalières entre le 1er janvier et le 30 avril 2007. Proposer une cause expliquant la consommation fin février et début avril. On pourra utiliser en plus de `matplotlib` le package `seaborn` pour améliorer le rendu visuel.

On ajoute des informations de température pour cette étude : les données utiles étant disponibles dans le fichier `ECAD_2016-09-11.txt` [3]. Ici les températures relevées sont celles d'Orly (noter cependant qu'on ne connaît pas le lieux de relevé de la précédente base de données).

5) Charger les données avec `pandas`, et ne garder que les colonnes `DATE` et `TG`. Diviser par 10 la colonne `TG` pour obtenir des températures en degrés Celsius. Traiter les éléments de température aberrants comme des NaN.
6) Créer un DataFrame `pandas` des températures journalières entre le 1er janvier et le 30 avril 2007. Afficher sur un même graphique ces températures et la séries `Global_active_power`.

On considère maintenant le jeu de données `20080421_20160927-PA13_auto.csv`.

7) Proposer une visualisation de la pollution pour l'ozone sur la période d'étude.
8) Proposer une visualisation de la pollution par année pour l'ozone et pour le dioxyde d'azote. Commenter l'évolution temporelle.
9) Proposer une représentation par mois de la pollution à l'ozone et au dioxyde d'azote. Quel est le mois le plus pollué (pour chacun des polluants) ?

Pour aller plus loin

Vous pouvez aussi consulter les pages suivantes :

— http://www.python.org
— http://scipy.org
— http://www.numpy.org
— http://scikit-learn.org/stable/index.html

3. on peut aussi trouver d'autres informations sur le site http://eca.knmi.nl/dailydata/predefinedseries.php

— http://www.loria.fr/~rougier/teaching/matplotlib/matplotlib.html
— http://jrjohansson.github.io/

Pour aller plus loin :

— http://blog.yhat.com/posts/aggregating-and-plotting-time-series-in-python.html
— http://www.math.univ-toulouse.fr/~besse/Wikistat/pdf/st-tutor2-python-pandas.pdf