TELECOM ParisTech **MDI720**
2019/2020 **François Portier, Anne Sabourin**

TP : Introduction: Python, Numpy, Pandas, *et al.*

Pour ce TP de test de la plate-forme "Classgrade" vous devez déposer un **unique** fichier **anonymisé** (votre nom doit apparaître uniquement dans le nom du fichier lui-même) sous format `ipynb` sur le site http://peergrade.enst.fr/.

Vous devez charger votre fichier, avant le lundi 16/09/2019 23h59. Entre le lundi 16/09/2019 et le mardi 17/09/2019, 23h59, vous devrez noter 2 copies qui vous seront assignées anonymement, en tenant compte du barème suivant pour chaque question du TP :

- 0 (manquant/ non compris/ non fait/ insuffisant)
- 1 (passable/partiellement satisfaisant)
- 2 (bien)

Ensuite, il faudra également évaluer de la même manière les points suivants (qui correspondent à 2 questions supplémentaires) :

- aspect global de présentation : qualité de rédaction, orthographe, présentation, graphes, titres, etc...
- aspect global du code : indentation, Style PEP8, lisibilité du code, commentaires adaptés
- Point particulier : absence de bug sur votre machine

Des commentaires pourront être ajoutés question par question si vous en sentez le besoin ou l'utilité pour aider la personne notée à s'améliorer, et de manière obligatoire si vous ne mettez pas 2/2 à une question. Enfin, veillez à rester polis et courtois dans vos retours.

Rappel : aucun travail par mail accepté !

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)
```

Rem: Pour tous les traitements numériques, on utilisera `numpy` (pour la gestion des matrices notamment) et `scipy`.

1 Introduction à Python, Numpy et Scipy

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. 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` et `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 (en anglais) :
http://pandas.pydata.org/pandas-docs/stable/tutorials.html

- Charger des données -

1. aussi connu sous le nom de code de César.

On utilise la base de données **Individual household electric power consumption Data Set**, que l'on pourra télécharger depuis https://bitbucket.org/portierf/shared_files/downloads/household_power_consumption.txt.

On s'intéresse aux grandeurs `Global_active_power` et `Sub_metering_1`.

10) Charger la base puis détecter et dénombrer le nombre de lignes ayant des valeurs manquantes.
11) Supprimer toutes les lignes avec des valeurs manquantes.
12) Modifier la variable `Sub_metering_1` en la multipliant par 0.06.
13) Utiliser `to_datetime` et `set_index` pour créer un DataFrame (on prendra garde au format des dates internationales qui diffère du format français).
14) 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 ici https://bitbucket.org/portierf/shared_files/downloads/TG_STAID011249.txt [2]. Ici les températures relevées sont celles d'Orly (noter cependant qu'on ne connaît pas le lieux de relève de la précédente base de données).

15) 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.
16) 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`.

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

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