**TELECOM ParisTech** **MDI720**
**2016/2017** **François Portier, Joseph Salmon**

**TP n° 1 : Introduction: Python, Numpy, Pandas, *et al.***

# Quelques remarques avant de démarrer

On va utiliser `ipython notebook` durant toutes les séances. Pour cela choisissez la version Anaconda sur les machines de l'école. Attention il y a deux versions et celle sans Anaconda est obsolète.

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)
```

# Introduction à Python, Numpy et Scipy

Cette section peut être faite rapidement pour ceux déjà familiers avec ces éléments. Pour les autres il est bon de parcourir les documents suivants :

http://perso.telecom-paristech.fr/~gramfort/liesse_python/1-Intro-Python.html
http://perso.telecom-paristech.fr/~gramfort/liesse_python/2-Numpy.html
http://perso.telecom-paristech.fr/~gramfort/liesse_python/3-Scipy.html

On pourra faire les questions associées :

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) À partir 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 $\pi$ 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 la lettre associe son nombre d'occurrences.

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

6) Calculer $2\prod_{k=1}^{\infty}\frac{4k^2}{4k^2-1}$, efficacement. On pourra utiliser `time` par exemple pour déterminer une version rapide. Proposer une version sans boucle utilisant `Numpy`

7) Créer une fonction `quicksort` en partant du pseudo-code suivant :

```
function quicksort('array')
 if length('array') <= 1
     return 'array'
 select and remove a pivot value 'pivot' from 'array'
 create empty lists 'less' and 'greater'
 for each 'x' in 'array'
    if 'x' <= 'pivot' then append 'x' to 'less'
    else append 'x' to 'greater'
 return concatenate(quicksort('less'), 'pivot', 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 $5\times 6$ aléatoire à coefficients uniformes dans $[-1,1]$, puis remplacer une colonne sur deux par sa valeur moins le double de la colonne suivante. Remplacer les valeurs négatives par 0 en utilisant un masque binaire.

9) Créer une matrice $M \in 5\times 20$ aléatoire à coefficients uniformes dans $[-1,1]$. Tester que $G = M^\top M$ est symétrique (semi-) définie positive, et que valeurs propres sont positives. Quel est le rang de $G$ ?
Aide : on pourra utiliser `np.allclose`, `np.logical_not`, `np.all` par exemple.

# 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 -

On utilise la base de données [1] **Individual household electric power consumption Data Set**.

10) Pour cela utiliser les commandes ci-dessous :

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

url = u'https://archive.ics.uci.edu/ml/machine-learning-databases/00235/'
filename = 'household_power_consumption'
zipfilename = filename + '.zip'
Location = url + zipfilename
if not(path.isfile('zipfilename')):
    urllib.urlretrieve(Location, 'zipfilename')
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)
```

1. https://archive.ics.uci.edu/ml/datasets/Individual+household+electric+power+consumption

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

11) Détecter et dénombrer le nombre de lignes avec des valeurs manquantes.
12) Supprimer toutes les lignes avec des valeurs manquantes.
13) Utiliser `to_datetime` et `set_index` pour créer une Time Series (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`.

On va maintenant ajouter des informations de température pour cette étude. Les données utilisées sont disponibles sur EOLE dans "`TG_STAID011249.txt`"[2]. Ici les températures relevées sont celles d'Orly (noter cependant qu'on ne connaît le lieux de relevée 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 une Time Series `pandas` des températures journalières entre le 1er janvier et le 30 avril 2007. Afficher sur un même graphique ces températures et la Time Series `Global_active_power`.

## Pour aller plus loin

— Prenez le jeu de données `airparif_abae1bd78def4fe8a409ab8c95fc4608.zip` sur EOLE, et proposer une visualisation de la pollution par année. Faites aussi une représentation par mois quel est le mois le plus pollué.

— 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