

---

## TP N° 1 : Prise en main de jupyter notebook

---

Objectifs du TP : savoir lancer `jupyter notebook`, lancer / sauvegarder un notebook, utiliser les premières commandes Python standard, importer une librairie.

Quelques conseils de base : Pour chaque fiche de TP, nous utiliserons un script `jupyter notebook`. Il est conseillé de créer un répertoire HLMA408, puis un sous-répertoire de pour chaque TP, *e.g.*, TP1, TP2, TP3. Dans un tel répertoire TP1, vous stockerez :

- le sujet de TP au format PDF,
- les fichiers de données (qui seront téléchargés automatiquement),
- le fichier `TP-Introduction\_squelette.ipynb` qui contient les commandes (pratiquement) pré-remplies correspondant au TP, et que vous pouvez télécharger sur le site du cours.

## 1 Utiliser un jupyter notebook

Selon l’environnement que vous utilisez, veuillez vous reporter à l’aide correspondant à votre système d’exploitation (🇬🇧 : *operating system, OS*). Pour ceux qui utilisent les machines des salles TP, les indications concernant le chargement de `jupyter notebook` sont détaillées dans le polycopié du cours, Section 2.3.1 disponible ici :

<http://josephsalmon.eu/enseignement/Montpellier/HLMA310/IntroPython.pdf>

En distanciel, nous recommandons d’utiliser <https://colab.research.google.com/> pour effectuer le TP si votre machine ne permet pas de lancer `jupyter notebook` (si vous utilisez `colab`, une charger le fichier `TP-Introduction\_squelette.ipynb` et passez en question 4).

- 1) Télécharger le fichier `TP-Introduction\_squelette.ipynb` dans votre répertoire HLMA408/TP1 <sup>1</sup>
- 2) Lancer `jupyter notebook` de la manière suivante : cliquer sur l’icône `Anaconda3`, puis dans le terminal taper la commande `jupyter notebook` ou sur `colab`.
- 3) Dans votre explorateur internet apparaît un environnement `jupyter notebook` qui vous permet de naviguer et de vous déplacer là où vous avez sauvegardé votre fichier. Cliquez alors sur le nom du fichier `TP-Introduction\_squelette.ipynb` pour l’ouvrir.
- 4) Vérifiez quelle version de Python est installée en utilisant les commandes <sup>2</sup> :

```
from platform import python_version
print(python_version())
```

- 5) Ajoutez une nouvelle cellule tout en haut du `jupyter notebook` : soit en cliquant sur “+” soit en appuyant sur “`esc+A`” sur le clavier (sous Linux).
- 6) Mettez un titre dans cette cellule sous format markdown (en utilisant l’onglet “Cell” puis “Cell Type” et en choisissant “Markdown”) <sup>3</sup>. Ce titre sera formé de la manière suivante :

```
# TP1 - HMLA408: Prénom Nom
```

où vous remplacerez “Prénom” et “Nom” par votre prénom et votre nom de famille. Pour obtenir quelques rudiments de markdown (titres de niveau 1, 2 etc., écriture en gras, en italique,...) vous pouvez consulter le lien :

<http://www.boiteaoutils.info/2013/02/ecrire-tout-simplement-introduction/>

- 
1. n’utiliser **JAMAIS** d’espace dans le nom d’un fichier ou d’un dossier !
  2. sur les machines de l’université la version était 3.5.5 au 1er septembre 2018
  3. sous Linux on obtient de même en pressant “`Esc/Echap`” suivi de la touche “`M`”

- 7) Sauvegardez le `jupyter notebook` sous le nom `tp1_hlma408_prenom_nom.ipynb`, où vous remplacerez `prenom_nom` par votre nom de famille et votre prénom (sans majuscule). Pour cela cliquez sur le haut du fichier à côté de l'icône `jupyter` et remplacez le mot (par défaut pour un nouveau notebook il est écrit "Untitled", mais ici son nom était "TP-Introduction\_squelette").
- 8) Fermez votre notebook (que ce soit avec `colab` ou `jupyter notebook`). Rouvrez et relancez votre fichier `tp1_hlma408_prenom_nom.ipynb`.
- 9) Testez l'**auto-complétion** de la manière suivante : dans une cellule de type code<sup>4</sup> commencez à taper `pyth` et appuyer sur la touche "tab" du clavier. Vous devriez alors avoir accès à un menu déroulant et pouvoir retrouver la fonction `python_version` sans avoir à taper le nom entier.
- 10) Tapez la commande `pwd` (pour *print working directory*) dans une cellule code. À l'endroit obtenu dans votre arborescence, créez un fichier texte brute (par exemple avec le logiciel `gedit`, mais **SURTOUT** pas avec Word!) composé des deux lignes suivantes :

```
from platform import python_version
print(python_version())
```

Enregistrez ce fichier à cet endroit sous le nom suivant : `test.py` Dans votre `jupyter notebook` vérifiez avec la commande `ls` que le fichier `test.py` est bien disponible. Lancez alors ce fichier en exécutant la commande :


```
run test.py
```

On veillera dans la suite à créer une cellule par question, et avant chaque question on ajoutera un titre markdown, en tapant le type de commande suivant (par exemple pour la Question 9)

```
## Question 9:
```

## 2 Télécharger une base de données

### 2.1 La fonction `read_csv`

Pour utiliser la table de données ( : *data frame*) nommée `babies23.data` utilisée dans le cours, lancer les commandes suivantes<sup>5</sup> (elles sont déjà remplies dans le `jupyter notebook` fourni `TP1_ebauche.ipynb`)

```
# Download
import os
from download import download
import pandas as pd
url = "http://josephsalmon.eu/enseignement/datasets/babies23.data"
df_name = "babies23.data"
path_target = os.path.join("./", df_name)
download(url, path_target, replace=False)
df_babies = pd.read_csv("babies23.data", skiprows=38, sep='\s+')
# \s+ : for handling spaces
```

Remarque : il y a deux façons d'importer des fonctions d'une librairie. On peut soit importer la librairie telle quelle *e.g.*, avec

```
import os
```

on importe la librairie `os` ou bien on peut lui donner un acronyme, *e.g.*, avec

```
import numpy as np
```

4. pour obtenir le type de cellule cliquer l'onglet `Cell` puis sur `Cell Type`

5. si le package `download` n'est pas installé sur votre machine lancer `pip install download` dans votre prompt Anaconda

on importe la librairie `numpy` sous le nom raccourci `np`, comme fait dans le cours.

Il est aussi possible d'importer seulement une fonction, *e.g.*, on importe la fonction

```
python_version
```

de la librairie `platform` avec la commande `from platform import python_version`.

- 11) Utiliser la commande “Restart & run all” (■ ■ : *redémarrer & lancer tout*) de l’onglet “Kernel” (■ ■ : *noyau*). Quelle est la différence avec “Restart & clear output” (■ ■ : *redémarrer & effacer la sortie*)?
- 12) Pour lire l’aide de la fonction `download` et comprendre l’utilité de l’option “replace=False” tapez :

```
download?
```

- 13) Vérifiez avec la fonction `ls` que le fichier est bien téléchargé localement.
- 14) Ouvrir le fichier `babies23.data`<sup>6</sup> (par exemple avec `gedit` ou un autre éditeur de texte) et parcourir le début du fichier pour comprendre brièvement ce que contient cette base de données. On pourra aussi consulter le fichier `babies.readme.txt`<sup>7</sup> pour plus de détails.  
Rem. : pour plus d’indication sur la base de données voir les notebooks des premiers cours.
- 15) Afficher le début du tableau de données `babies` avec la fonction `head`. Dans le fichier texte, il y avait deux colonnes (c’est-à-dire deux variables) qui s’appelaient `wt`. Comment Pandas a-t-il résolu le conflit ?

## 2.2 Accéder aux lignes et colonnes de la table de données

Une table de données est un objet de type `dataframe` en `pandas`. Les différentes variables sont rangées dans des colonnes alors que les lignes correspondent aux différentes observations. Les lignes et les colonnes sont numérotées à partir de 0 (c’est la convention de Python qui diffèrent de R qui compte à partir de 1).

Les colonnes ont des noms que l’on obtient ici avec la commande `df_babies.columns`. Il est donc important de nommer les noms de manière pertinente pour les retrouver aisément. On peut extraire une sous-table de données ou bien un vecteur colonne depuis une table de données. Pour cela on utilise la commande `NomDeLaTable['NomDeLaColonne']`.

- 16) Extraire la colonne ‘wt’ de la base.

Pour extraire les observations (qui sont **pratiquement toujours** les lignes du data frame) on peut utiliser la commande `iloc`. Ainsi la commande `df_babies.iloc[0]` extrait la première ligne du data-frame.

- 17) Observez ce que font les commandes : `df_babies.iloc[0]`, `df_babies.iloc[-1]`, `df_babies.iloc[:8]`, `df_babies.iloc[-5:]` et en déduire comment extraire les lignes des numéros 100 à 110 (c’est-à-dire les individus d’`id` 1552 à 1688).

## 3 Mettre en forme les colonnes du jeu de données

Les commandes de Python et de `pandas` s’adaptent au type d’objet sur lesquels on les applique. Il est donc fondamental de mettre correctement en forme la table de données (🇬🇧 : *data frame*). Pour cela, il faut particulièrement :

- re-coder les valeurs manquantes avec le code spécial `nan` (acronyme de *not a number*)
- potentiellement changer les unités pour qu’elles vous parlent (mettre dans le système métrique internationale les tailles, poids, etc.)

La fonction `replace` est particulièrement efficace pour remplacer certaines valeurs par une autre. Ainsi la commande suivante charge le package `numpy` et remplace les 99 par des `nan` :

```
import numpy as np
df_babies['ht'].replace(99, np.nan, inplace=True) # gère les données manquantes
```

6. <http://josephsalmon.eu/enseignement/datasets/babies23.data>

7. <http://josephsalmon.eu/enseignement/datasets/babies.readme.txt>

En résumé on vient de remplacer dans la colonne `ht`, qui représente la taille (🇬🇧 : *height*) des mères en pouces<sup>8</sup> (🇬🇧 : *inches*) le nombre 99 (qui code pour les tailles manquantes : personne (?) ne mesurant  $99 \times 2.54 = 251.46$  cm dans l'étude...) par des `nan`.

18) Décrire ce que produit les instructions suivantes :

```
is_preprocessing_done = False
if is_preprocessing_done is False:
    df_babies['ht'] = df_babies['ht'] * 2.54
    is_preprocessing_done = True
```

19) Proposez un traitement similaire pour la taille des pères.

20) Lancez la commande suivante qui permet de supprimer toutes les données manquantes (ce qu'on fera pour l'instant).

```
df_babies.dropna(inplace=True)
```

21) Enlever toutes les colonnes du data frame sauf celles qui correspondent aux tailles des pères et des mères. On utilisera l'extraction d'une liste de colonnes : par exemple `df_babies[['ht', 'dht', 'sex']]` extrait la base de données composée de trois colonnes 'ht', 'dht' et 'sex'. On affectera ce nouveau data frame dans l'ancien (*i.e.*, on utilisera le même nom pour le nouveau data frame et pour l'ancien).

## 4 Statistiques descriptives

On présente ci-dessous quelques fonctions pour résumer des données.

### Commandes pour obtenir des résultats numériques

Commande	Description succincte
<code>df_babies.describe()</code>	Calcule des statistiques résumées de <code>df_babies</code>
<code>df_babies['ht'].mean()</code>	Calcule la moyenne de <code>df_babies['ht']</code>
<code>df_babies['ht'].std()</code>	Calcule l'écart-type de <code>df_babies['ht']</code>
<code>df_babies['ht'].median()</code>	Calcule la médiane de <code>df_babies['ht']</code>
<code>df_babies['ht'].quantile(0.99)</code>	Calcule les quantiles de <code>df_babies['ht']</code>

22) Testez ces fonctions sur la taille des pères et celle des mères.

23) Pour ces deux variables, donner la valeurs des quantiles suivants : 0.01, 0.05, 0.1, 0.9, 0.95, 0.99.

Passons maintenant à la partie graphique. On utilisera surtout les packages `matplotlib` et `seaborn` (une extension graphiquement plus élaborée que `matplotlib`).

24) Lancer la commande :

```
import matplotlib.pyplot as plt
import seaborn as sns
```

On résume ci-dessous un certain nombres de commandes utiles pour la visualisation.

### Commandes pour obtenir des graphiques

Commande	Description succincte
<code>plt.figure(figsize=(5,5))</code>	créer une figure d'une taille $5 \times 5$
<code>plt.hist(df_babies['ht'], bins=18)</code>	créer un histogramme de la variable <code>ht</code>
<code>ax = sns.kdeplot(df_babies['ht'])</code>	trace un estimateur à noyaux de la densité
<code>plt.scatter(df_babies['ht'], df_babies['dht'])</code>	affiche un nuage de points

On pourra regarder l'aide et les exemples du cours pour améliorer :

8. 1 inch = 2.54 cm

- les légendes
  - les titres
  - le nom des axes
  - les couleurs
  - les styles de marqueurs
  - etc.
- 25) Afficher un histogramme de la taille des mères avec 12 boîtes (🇬🇧 : *bins*).
  - 26) Afficher un estimateur de la taille des mères avec un estimateur à noyaux de la densité (🇬🇧 : *kernel density estimator, KDE*)
  - 27) Afficher côte à côte<sup>9</sup> deux boîtes à moustache de la taille des pères et des mères
  - 28) Afficher côte à côte deux violons de la taille des pères et des mères.
  - 29) Afficher un nuage de points avec pour abscisse (axe des  $x$ ) la taille des pères et pour ordonné (axe des  $y$ ) la taille des mères. En particulier on proposera un moyen de distinguer visuellement les couples  $(x, y)$  qui apparaissent souvent.
  - 30) Afficher un widget qui sur deux graphiques alignés l'un au-dessus de l'autre (avec `subplot` donc) affiche la masse des pères et des mères, et dont le curseur contrôle le nombre de boîtes (🇬🇧 : *bins*).
  - 31) Afficher un widget qui sur deux graphiques alignés l'un au-dessus de l'autre (avec `subplot` donc) affiche la taille des pères et des mères, et dont le curseur contrôle la taille de la fenêtre de lissage (🇬🇧 : *bandwidth*).

## 5 Pour aller plus loin

Tutos/Vidéos de Jake Vanderplas : [Reproducible Data Analysis in Jupyter](#)<sup>10</sup>

---

9. pour cela on pourra considérer la commande suivantes : `fig, axs = plt.subplots(nrow, ncol, figsize=(8, 5))`

10. <http://jakevdp.github.io/blog/2017/03/03/reproducible-data-analysis-in-jupyter/>