Université de Montpellier **HLMA310**
2020/2021 **Joseph Salmon**

TP N° 1 : Introduction à `Python` et prise en main

Objectifs du TP : lancer, sauvegarder, relancer un `jupyter notebook`, découvrir les premières commandes `Python`, importer une librairie.

Aide : un fichier squelette est proposé pour vous accompagner, et est disponible à l'adresse suivante : http://josephsalmon.eu/enseignement/Montpellier/HLMA310/TP1-Introduction-skeleton.ipynb, vous pouvez le télécharger et le lancer en suivant la procédure vue en cous (ou dans le polycopié).

ATTENTION : si vous n'avez pas accès à `jupyter notebook` sur votre machine, vous pouvez utiliser le site : https://mybinder.org/. Renseignez alors dans la boite "GitHub repository name or URL" l'url suivant : https://gitlab.com/josephsalmon/hlma310/, en mettant l'option `Gitlab` puis cliquez sur "Launch". Cela permet de lancer un notebook de manière distante, et d'effectuer ce TP sans installation sur votre machine (le temps de lancement peut être plus ou moins long en fonction du trafic).

1 Utiliser un `jupyter notebook`

Selon l'environnement que vous utilisez, veuillez vous reportez 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 est détaillée dans le polycopié du cours, Section 2.3.1 disponible ici :
http://josephsalmon.eu/enseignement/Montpellier/HLMA310/IntroPython.pdf

1) Lancer un nouveau `jupyter notebook` sur votre machine.
2) Vérifier quelle version de `Python` vous avez (sur les machines de l'université cela devrait être 3.5.5), en utilisant :

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

3) Ajouter un titre dans une cellule sous format markdown (en utilisant l'onglet "Cell" ou "Cellule" puis "Cell Type" ou "Type de Cellule" et en choisissant "Markdown")[1]. Ce titre sera formé de la manière suivante :

```
# TP1 - HMLA310: Prénom Nom
```

où vous remplacerez "Nom" et "Prénom" par votre prénom et votre nom de famille. Pour obtenir quelques rudiments de markdown, vous pouvez consulter le lien :
http://www.boiteaoutils.info/2013/02/ecrire-tout-simplement-introduction/

4) Sauvegarder le notebook sous le nom `tp1_hmla_prenom_nom.ipynb`, où de nouveau vous remplacerez `prenom_nom` par votre nom de famille et votre prénom (sans majuscule). Vous pouvez soit utiliser le menu "File/save as" (ou "Fichier/enregistre sous"), ou bien modifier le titre tout en haut (à côté du logo `jupyter`) et cliquer sur l'icône de sauvegarde
5) Fermez l'onglet du navigateur correspondant à votre `notebook` et retrouvez son emplacement sur votre disque dur. Si besoin déplacer dans un dossier qui vous convient associé à ce cours. Rouvrez ensuite le fichier que vous venez de créer.
6) Tester l'auto-complétion de la manière suivante : exécuter la cellule de code que vous avez créer précédemment, puis dans une nouvelle cellule de type code taper `pyth` et appuyer sur la touche "tab" du clavier (celle à gauche de la lettre "a" sur un clavier azerty). Vous devriez alors avoir accès à un menu déroulant et pouvoir retrouver la fonction `python_version` sans avoir à taper le nom entier.

1. On peut faire cette opération en utilisant le menu déroulant, tout en haut, à côté des icônes

7) Exécuter la commande `pwd` (pour *print working directory*) dans une cellule code. À l'endroit obtenu dans votre arborescence, créer un fichier de texte brut (par exemple avec le logiciel `Editeur de texte` ou `gedit`) composé des lignes suivantes :

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

Enregistrer ce fichier sous le nom de `test.py` à l'endroit retourné par la commande `pwd`. Dans votre `jupyter notebook` vérifiez avec la commande `ls` que le fichier `test.py` est bien disponible. Lancer alors ce fichier en exécutant la commande suivante dans une cellule de code, et décrivez ce qu'il se passe :

```
run test.py
```

On veillera dans la suite a créer une cellule par question, et avant chaque question on ajoutera un titre markdown, en tapant la commande suivante (par exemple pour la Question 1)

```
## Question 1:
```

8) Comparer la différence d'affichage quand on utilise un `#, ##, ###`
9) Utiliser la commande "Restart & run all" (ou *Redémarrer & tout exécuter*) de l'onglet "Kernel (ou Noyau)". Quelle est la différence avec "Restart & clear output" (ou *Redémarrer & tout effacer*) ?

2 Chaînes de caractères et nombres

10) Ici, on veut créer la chaîne de caractères `tp1_hlma_nom_prenom.ipynb` automatiquement en partant seulement des éléments suivants :

```
prenom = "Joseph" # à remplacer par ce qu'il faut
nom = "Salmon" # à remplacer par ce qu'il faut
extention = ".ipynb"
tp = "TP1_HLMA310"
```

En particulier on pourra utiliser la concaténation (ou bien la fonction `join`) et l'attribut `.lower()`, pour transformer les majuscules en minuscules automatiquement. Vous pourrez alors copier-coller la chaîne de caractères obtenue pour remplacer le nom du fichier.

11) À partir de la chaîne de caractères `alphabet` obtenue comme il suit :

```
import string
alphabet = string.ascii_lowercase
```

générez par des opérations de *slicing* (*cf.* Section 4.1 du polycopié) les chaînes de caractère `'cfilorux'` et `'zxvtrpnljhfdb'`.

12) Afficher le nombre π à 22 décimales après la virgule. Afficher de même $e = \exp(1)$ cette fois à 6 chiffres après la virgule. On pourra partir de l'exemple suivant :

```
import math
s = "Le nombre {} est égal à {:08.2f}"
print(s.format("pi", math.pi))
```

Consultez l'aide sur `format` pour l'affichage de flottants à l'adresse suivante si besoin : https://pyformat.info/#number

3 Listes

13) En s'inspirant de la liste `list_of_ints` définie ci-dessous, proposez une manière d'obtenir une liste appelé `lst_of_even` qui contient les nombres pairs entre 2 et 21 :

```
list_of_ints = list(range(1,30,3))
```

14) Taper la commande :

```
list_of_ints.sort(reverse=False)
```

Quel est l'effet de cette commande sur la liste `list_of_ints` ? Proposer alors un moyen de trier la liste de manière décroissante.

15) Que ce passe-t-il après avoir lancer la commande suivante ?

```
list_of_ints.insert(0,0)
```

Observez de même ce qui se passe avec les commandes :

```
list_of_ints.remove(19)
```

et

```
list_of_ints.pop()
```

Enlever le 2e élément de la `list_of_ints`, en utilisant si besoin l'aide de `pop` avec la commande :

```
list_of_ints.pop?
```

16) Relancer la commande "Restart & run all" (ou *Redémarrer et tout exécuter*), et suivez l'évolution de `list_of_ints` au cours des cellules.

17) Exécuter la commande suivante :

```
print(2 in lst_of_even)
print(4 in lst_of_even)
print(3 in lst_of_even)
```

Pouvez vous interpréter la valeur des booléens créés ?

4 Dictionnaires

Commençons par créer un dictionnaire vide avec la commande suivante[2] :

```
dico_vide = dict() # ou alors avec: dictionnaire_vide = {}
```

18) Ajouter au dictionnaire `dico_vide`, le couple clé/valeur suivant :
clé : `"Dioxyde de carbone"` ; valeur : `"CO2"`.

19) Charger un dictionnaire de gaz polluants avec la commande[3] :

```
import pandas as pd
url="http://josephsalmon.eu/enseignement/datasets/polluants.csv"
dico_polluants = pd.read_csv(url, header=None).set_index(0).squeeze().to_dict()
print(dico_polluants)
```

2. Noter que la commande similaire pour créer une liste vide est : `list_vide=[]`
3. On verra extensivement la librairie `pandas` d'ici quelques séances. Pour le moment on ne demande pas de comprendre l'utilisation de de cette librairie, qui sert ici uniquement à créer un dictionnaire.

Si vous êtes sur une machine distante https://mybinder.org/, on pourra installer `pandas` avec la commande :

```
pip install pandas
```

20) Taper la commande

```
dico_polluants.
```

et appuyer sur "tab". En utilisant l'attribut `update` dans les attributs possibles, fusionnez le dictionnaire `dico_polluants` avec le dictionnaire `dico_vide`.

21) Enlever du dictionnaire `dico_polluants` l'entrée `'Particules ou poussières en suspension'` avec l'attribut `pop`.

5 Pour aller plus loin

— Des extensions sont utiles pour `jupyter-notebooks` et sont disponibles à l'adresse : https://github.com/ipython-contrib/jupyter_contrib_nbextensions.

— Quelques éléments de détails peuvent être nécessaires pour comprendre le type `range`. Une discussion riche d'enseignements est disponible ici : https://www.pythoncentral.io/pythons-range-function-explained/