TP 1. Utilisation des notebooks Jupyter

Exercice 1 : prise en main de Python et Jupyter

Python est un langage de programmation très polyvalent et peut notamment être utilisé pour la programmation scientifique et le traitement d'images. Nous utiliserons l'environnement Jupyter, qui s'exécute dans un navigateur web, pour écrire des programmes. Ces programmes sont enregistrés dans des *notebooks*.

1. Pour commencer, lancez Jupyter en tapant dans un terminal :

   ```
   jupyter-notebook
   ```

2. Créez un nouveau notebook Python 3 en cliquant sur le bouton Nouveau en haut à droite.
3. Sélectionnez le dossier de travail, puis changez le nom de votre notebook, en "premierNotebook" en cliquant sur le menu `File > Rename`.

Un *notebook* est structuré en cellules (*cells*), dans lesquelles on peut écrire du code ou du texte.

4. Dans la première cellule du notebook, écrivez

   ```
   1+1 Shift + Entrée
   ```

   Le code est exécuté, le résultat s'affiche puis une nouvelle cellule est créée.
5. Comme tout langage de programmation, le code s'écrit en utilisant des variables, des opérateurs et des fonctions. Une variable permet de stocker une ou plusieurs valeurs [1]. Tapez les instructions ci-dessous dans la deuxième cellule :

   ```
   annee = 2020 Entrée
   cours = "ISSD" Shift + Entrée
   ```

   Vous avez ainsi stocké la valeur 2020 dans la variable `annee`, et la chaîne de caractères « ISSD » dans la variable `cours`.
6. Modifiez la cellule précédente en ajoutant l'instruction suivante :

   ```
   print(cours + " " + str(annee)) Shift + Entrée
   ```

   Le code de la cellule est alors réexécuté.

   Vous pouvez suivre le tutoriel sur python pour apprendre ce langage de programmation, https://docs.python.org/fr/3/tutorial/introduction.html.

Exercice 2 : Mise en forme et présentation, Markdown et RISE

L'intérêt d'un notebook, outre le fait d'exécuter les cellules comme on le souhaite, est d'ajouter du texte en utilisant le langage markdown.
La syntaxe markdown est accessible depuis le menu `Help` ou sur help.github.com/en/articles/basic-writing-and-formatting-syntax.

1. Sélectionnez une cellule vide, puis cliquez sur le menu `Cell > Cell type > Markdown`. Vous pouvez alors y écrire du texte qui pourra être mis en forme. Cela peut être intéressant pour insérer des titres ou conserver des commentaires et des notes.

1. Le nom de la variable peut contenir des lettres, des chiffres (à l'exception du premier caractère) ou le tiret bas. La casse est importante (`a` et `A` sont deux variables distinctes).

2. Ajoutez le titre “Mon premier Notebook Jupyter”.
3. Commentez le code afin d’obtenir un résultat semblable à celui du fichier premierNotebook.ipynb.

Un notebook peut être transformé et affiché sous la forme d’un diaporama, en conservant l’interaction avec python, avec l’extension RISE que nous avons incluse dans notre environnement de travail. Une présentation de RISE est disponible sur https://rise.readthedocs.io/en/maint-5.6/usage.html. Comme vous pouvez le lire en suivant le lien précédent, une diapo est une nouvelle diapositive qui apparaît par la droite alors qu’une sous-diapo porte mal son nom et est aussi une nouvelle diapositive mais qui apparaît par le bas. Les extraits permettent de faire afficher au fur et à mesure le contenu de la diapositive. Nous allons transformer votre premier Notebook en une présentation :

4. Faites une première diapositive avec le titre seulement.
5. Faites une seconde diapositive avec la seconde cellule, celle du premier calcul, et faites en sorte que le code python 1+1 apparaisse sur cette diapositive quand on appuie à nouveau sur la barre d’espace.
6. Faites une troisième diapositive de la même façon : le texte d’abord, puis le code.
7. Placez le curseur sur la première cellule et lancez le diaporama, en cliquant sur le bouton illustré par un histogramme ou par le raccourci alt-r.

Vous devriez obtenir un résultat semblable à celui du fichier premierePresentation.ipynb.

Exercice 3 : modules Python et ouverture d’un fichier au format csv

Python est complété par des *modules*, notamment :
- `numpy` et `scipy` pour la programmation scientifique,
- `pandas` pour le traitement de fichier de données et la manipulation de ces données,
- `matplotlib` et `seaborn` pour l’affichage des résultats.

Ces modules regroupent des sous-modules qui regroupent des fonctions.

1. Créez un nouveau notebook.
2. Dans une cellule, tapez les instructions suivantes pour utiliser le sous-module `stats` du module `scipy` et le sous-module `pyplot` du module `matplotlib` (notez que ce dernier est renommé `plt` : c’est plus court !)

```
from scipy import stats
import matplotlib.pyplot as plt
```

3. Importez le package pandas, abrégé par `pd` :

```
import pandas as pd
```

Le jeu de données est fourni au format CSV (Comma Separated Values). C’est un format qui peut notamment être manipulé avec un tableur (Microsoft Excel ou LibreOffice Calc). Chaque ligne correspond à un film, et les différentes informations le concernant sont séparées par des point-virgules. La première ligne du fichier correspond à l’en-tête des colonnes. Chaque ligne correspond ensuite à un élément du jeu de données et à ses caractéristiques étudiées.

4. Chargez le jeu de données stocké dans le fichier `movie_metadata2.csv` à l’aide de la fonction `read_csv()` du package `pandas` :

```
DATA = pd.read_csv('data/movie_metadata2.csv', delimiter=';',
index_col='movie_title') Shift + Entrée
```

5. Affichez les 15 premières lignes de ce fichier à l’aide de la fonction `head()` du package `pandas`

```
DATA.head(n = 15) Shift + Entrée
```

Exercice 4 : Manipulation de données simples avec l'objet Series de Pandas

Dans le langage python, la structure permettant de stocker un tableau de valeurs est appelé `Array`. Un `Array` peut être de dimension 1 (une colonne ou une ligne), de dimension 2 (un tableau à N lignes et M colonnes), ou de dimension supérieure (inutile dans le cadre de ce cours). L'objet `Array` ne permet pas de nommer les lignes et les colonnes autrement que par leur indice (0, 1, ..., N-1). Le package `pandas` permet justement de nommer les lignes et les colonnes d'un tableau, autorisant ainsi une manipulation aisée de l'information. Différentes structures permettent de manipuler des jeux de données sous `pandas` dans le cadre de ce TP nous allons nous intéresser à deux objets : les `Series` et les `DataFrame`.

L'objet `Series` est la structure la plus simple, il s'agit d'un tableau de taille $N \times 1$ capable de contenir N éléments de n'importe quel type (chaine de caractères, nombre, date, etc), l'initialisation d'un objet `Series` se fait par la syntaxe suivante :

```
s = pd.Series(data, index=index)
```

L'argument data est un tableau (`Array`) à une dimension contenant N valeurs. Si l'on ne précise par l'argument `index`, l'index par défaut sera le même que pour un `Array` : (0, 1, ..., N-1). L'argument `index` doit être de la même taille que l'argument `data`.

1. Commençons par créer un petit jeu de données avec une variable : la température moyenne des 12 mois de l'année à Strasbourg (source climatdestrasbourg.fr).

   ```
   temp = [1.9 , 2.9, 7, 10.5, 15, 18.1, 20.1, 19.8, 15.8, 11.2,
    5.8, 2.8] Entrée
   mois = ['Janvier', 'Fevrier', 'Mars', 'Avril', 'Mai', 'Juin',
    'Juillet', 'Aout', 'Septembre', 'Octobre', 'Novembre', 'Decembre'] Entrée
   s = pd.Series(temp, index=mois) Shift + Entrée
   ```

2. Dans une nouvelle cellule affichez la série `s`

   ```
   s  Shift + Entrée
   ```

3. Affichez le tableau d'indices de la série `s` :

   ```
   s.index  Shift + Entrée
   ```

4. Testez la création d'une série `s2` en ne précisant pas l'argument `index` et afficher cette série. Observer la différence avec la série `s`

5. Pour accéder à la valeur correspondant au mois 'Janvier' la syntaxe est la suivante :

   ```
   s['Janvier']  Shift + Entrée
   ```

   et pour la série `s2` :

   ```
   s2[0]  Shift + Entrée
   ```

Exercice 5 : Manipulation de données avec l'objet DataFrame de Pandas

L'objet `DataFrame` de Pandas est une structure en 2 dimensions $N \times M$ où N représente le nombre d'individus (nombre de lignes) et M le nombre de variables caractérisant ces individus (nombre de colonnes). Pour reprendre l'exemple de l'exercice précédent, les individus étudiés ici sont les 12 mois de l'année, les variables que l'on va considérer sont : la moyenne des températures moyennes entre 1980 et 2010 (valeurs de l'exercice précédent) et la température moyenne de l'année 2018 pour chaque mois de l'année :

```
temp2018 = [6.7, 1.0, 5.5, 14.6, 17.7, 20.3, 22.7, 22.4, 17.7, 12.5,
          6.7, 5.2]
```
Entrée + Shift

1. Nous allons créer un `DataFrame` avec la syntaxe suivante :

   ```
   df = pd.DataFrame({'Moyenne' : temp, 'temp2018' : temp2018}, index=mois)
   ```
   Entrée
   ```
   df
   ```
   Entrée + Shift

2. Affichez l'index de ce `DataFrame` avec la syntaxe suivante :

   ```
   df.index
   ```
   Entrée + Shift

3. Affichez le nom des colonnes de ce `DataFrame` avec la syntaxe suivante :

   ```
   df.columns
   ```
   Entrée + Shift

4. Il est maintenant possible d'ajouter une nouvelle colonne au `DataFrame` avec la syntaxe suivante :

   ```
   df['Difference'] = ['+', '-','-', '+', '+','+','+','+','+','+',
    '+','+']
   ```
   Entrée + Shift

   Affichez à nouveau le `DataFrame`.

5. La troisième variable appelée 'Difference' a un type différent des deux premières. Nous pouvons afficher le type des différentes variables avec la commande suivante :

   ```
   df.dtypes
   ```
   Entrée + Shift

6. Nous souhaitons maintenant récupérer une colonne du `DataFrame` avec le code suivant :

   ```
   s3 = df['temp2018']
   ```
   Entrée
   ```
   s3
   ```
   Entrée + Shift

   ou de manière equivalente :

   ```
   s3 = df.temp2018
   ```
   Entrée
   ```
   s3
   ```
   Entrée + Shift

   Quel est le type de l'objet `s3` ? (La fonction `type(obj)` vous permet de récupérer le type de l'objet Python `obj`).

   Pour récupérer plusieurs colonnes simultanément, nous utilisons la syntaxe suivante :

   ```
   df2 = pd.DataFrame(df, columns = ['temp2018', 'Difference'])
   ```
   Entrée
   ```
   df2
   ```
   Entrée + Shift

7. Nous voulons maintenant récupérer une portion des données en se limitant aux mois de Février à Mai. Pour cela nous utilisons la syntaxe suivante :

   ```
   df3 = df['Fevrier':'Mai']
   ```
   Entrée
   ```
   df3
   ```
   Entrée + Shift

   Ce code renvoie un nouveau `DataFrame` avec seulement N = 4 individus et toujours les mêmes variables. Nous pouvons obtenir le même résultat en utilisant les indices (la position) des individus au lieu de leur nom. Nous rappelons qu'en python les indices commencent à 0. Dans ce cas, il faut utiliser la syntaxe :

```
df4 = df[1:5] Entrée
df4 Entrée + Shift
```

Pour aller plus loin dans la manipulation des structures Pandas, vous pouvez vous reporter à la page 10 minutes to Pandas et appliquer les différentes commandes au `DataFrame df` que vous venez de créer.