Installation et prise en main de Python

Aymeric Jan (aymeric.jan@univ-paris1.fr)

January 30, 2024

1 Introduction

Python est un des langages de programmation les plus populaires actuellement. Très polyvalent, il permet aussi bien de faire de simples scripts scientifiques que de créer des logiciels ou des application web. Depuis plusieurs années, il est le language plébiscité pour l'intelligence artificielle (Data Science, Machine Learning, etc...).

2 Installation

2.1 Installation de Python

L'installation de Python est possible sur tous les OS (Windows, MacOS, Linux), pour ce faire, il suffit de télécharger une version compatible depuis le site officiel https://www.python.org/downloads/. Bien que les dernières versions possèdent d'intéressantes fonctionnalités, je recommande d'installer la version 3.8.10, disponible ici https://www.python.org/downloads/release/python-3810/. Pour ce faire, il suffit de télécharger la version correspondante depuis le tableau en bas de page, en général, il s'agit de la version Windows installer (64-bit). Il est conseillé de redémarrer l'ordinateur pour finaliser l'installation.

2.2 Installation des librairies

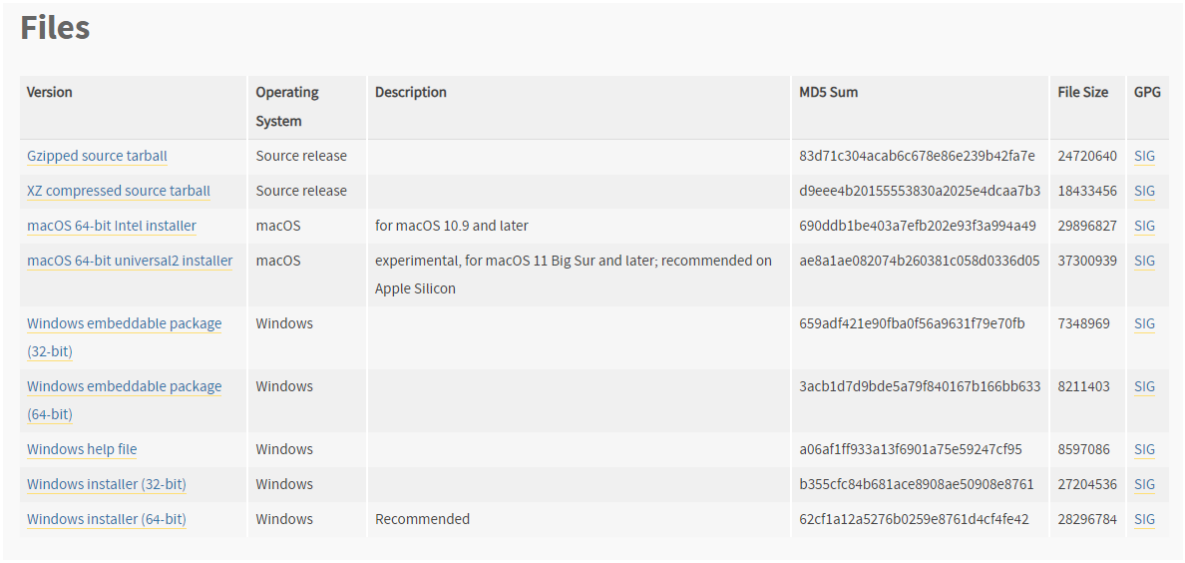
De nombreuses librairies sont disponibles en Python, la meilleure manière de les installer est de passer par la commande pip du terminal.

Dans notre cas, nous installerons les librairies suivantes

1. Numpy: Librairie de calcul scientifique.
2. Matplotlib: Librairie de visualisation.
3. Jupyter notebook: Notebook intéractifs.

Pour les installer, il suffit d'ouvrir le terminale et de taper les commandes suivantes

```
pip install numpy
pip install matplotlib
pip install notebook
```



Files

Version	Operating System	Description	MD5 Sum	File Size	GPG
Gzipped source tarball	Source release		83d71c304acab6c678e86e239b42fa7e	24720640	SIG
XZ compressed source tarball	Source release		d9eee4b20155553830a2025e4dcaa7b3	18433456	SIG
macOS 64-bit Intel installer	macOS	for macOS 10.9 and later	690ddb1be403a7efb202e93f3a994a49	29896827	SIG
macOS 64-bit universal2 installer	macOS	experimental, for macOS 11 Big Sur and later; recommended on Apple Silicon	ae8a1ae082074b260381c058d0336d05	37300939	SIG
Windows embeddable package (32-bit)	Windows		659adf421e90fba0f56a9631f79e70fb	7348969	SIG
Windows embeddable package (64-bit)	Windows		3acb1d7d9bde5a79f840167b166bb633	8211403	SIG
Windows help file	Windows		a06af1ff933a13f6901a75e59247cf95	8597086	SIG
Windows installer (32-bit)	Windows		b355cfc84b681ace8908ae50908e8761	27204536	SIG
Windows installer (64-bit)	Windows	Recommended	62cf1a12a5276b0259e8761d4cf4fe42	28296784	SIG

Figure 1: Installer Python.

3 Jupyter Notebook

Le plus pratique pour les TP/TD est d'utiliser ce qu'on appelle des notebooks. Similaire aux notebooks dans R, ils permettent d'exécuter de manière interactive seulement certaines parties du code. Ces parties de codes sont dans ce qu'on appelle des cellules. Vous trouverez sur le lien ci-après une documentation plus fournie sur les notebooks: https://docs.jupyter.org/en/latest/.

Pour démarrer un notebook, il faut ouvrir un terminal de commande et taper la commande suivante

```
jupyter notebook
```

Cela va ensuite ouvrir une page dans le navigateur web pour choisir le notebook à ouvrir.

3.1 Commandes de base sur les notebook

Voici quelques raccourcis très utiles pour les notebooks:

1. Créer une nouvelle cellule: b
2. Executer une cellule: Ctrl + return
3. Supprimer une cellule: d + d

4 Base en Python

4.1 Affectation et variables

En Python, il n'est pas nécessaire de déclarer le type de la variable, il suffit juste de donner le nom de la variable à gauche du signe = et sa valeur à droite. Des exemples sont donnés ci-après.

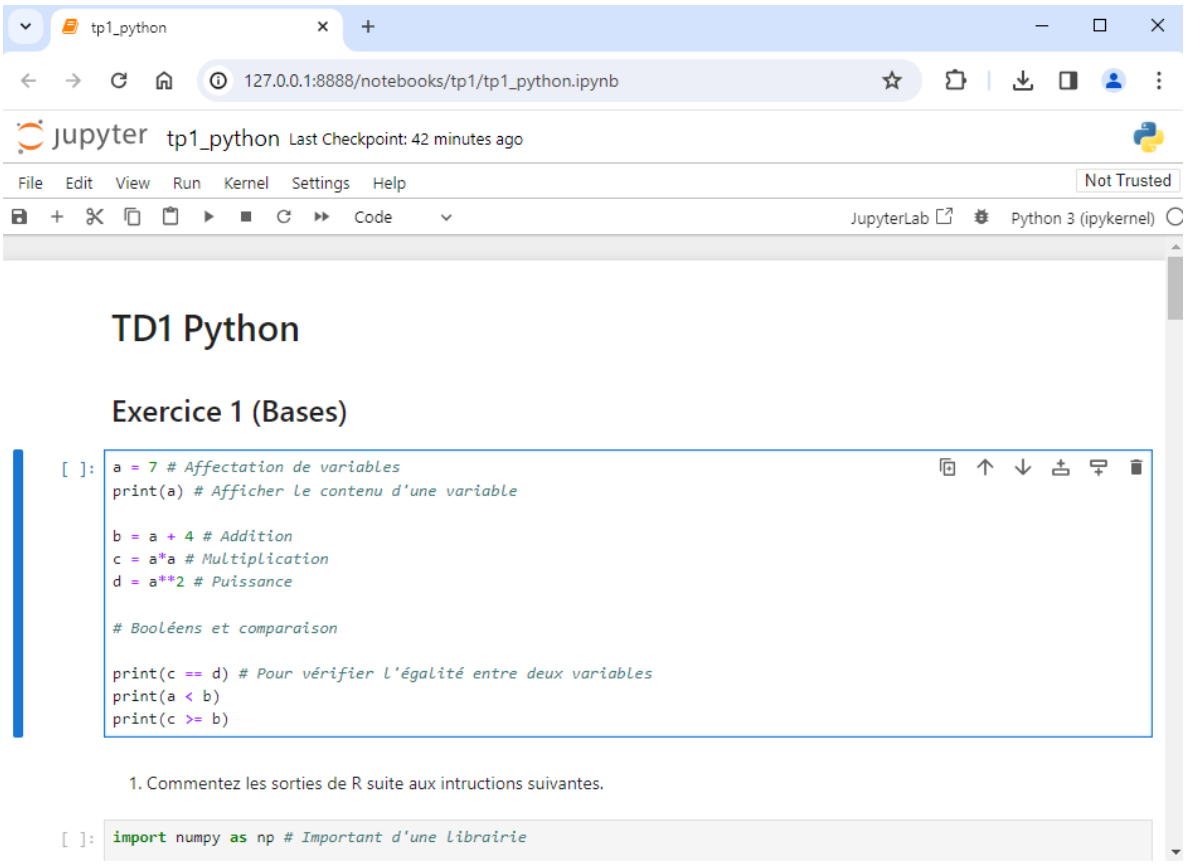


Figure 2: Exemple d'un Jupyter Notebook

```
var_int = 1 ## int
var_float = 1.4242 ## floats
my_string = "python" ## string
my_bool = True
print(var_float)
```

Les opérations de base comme l'addition, la soustraction, la multiplication etc... sont disponibles

```
result_addition = 1 + 2
result_mult = 1*2
result_puissance = 2**3
```

4.2 Boucles for, while et if statements

Il existe deux types de boucles en Python, les `for` et les `while`.

4.2.1 Boucle for

Une boucle `for` permet de répéter une action un certain nombre de fois défini à l'avance.

```
for ind in range(0, 10):
```

```
        print(ind)
```

4.2.2 Boucle while

Une boucle `while` permet de répéter une séquence de code plusieurs fois, tant que la condition est respectée.

```
ind = 0
my_condition = True
while my_condition:
    ind += 1
    my_condition = ind < 3
```

4.2.3 If et else

Il est possible d'ajouter des conditions avec `if` et `else` comme suit:

```
if 3+1 > 2:
    print("It's true")
else:
    print("You are wrong")
```

4.3 Structure de données

Il existe plusieurs types de structures de données en Python, en voici les principales.

4.3.1 Les listes

Une liste est une structure de données contenant une série de valeur ordonnées. Elle se définie à l'aide de crochets `[ ]` , ses éléments sont séparés par des virgules, et il est possible d'accéder à un élément de la liste à partir de son index en utilisant des crochets également. Il faut noter qu'en Python, les indices commencent à partir de 0.

```
my_list = [2, 3, 5]
print(my_list[2]) # Cela renvoie 5
```

On obtient la taille d'une liste en utilisant la fonction `len`

4.3.2 Les dictionnaires

Un dictionnaire est une structure de donnée où les valeurs sont identifiées par une clé. On utilise les accolades {} pour créer un dictionnaire. Chaque élément d'un dictionnaire est donc un couple (clé, valeur).

```
my_dict = {"pi":3.141593, "e":2.718282}
print(my_dict["pi"]) # Cela renvoie 3.141593
```

On peut obtenir la liste des clés d'un dictionnaire grâce à la méthode `keys`. Une méthode s'utilise comme suit: `my_dict.keys()`.

4.4 Les fonctions

En Python, une fonction se définie de la manière suivante

```
def my_func(arg_1, arg_2):
    var_interne = arg_1 + arg_2
    return var_interne

my_result =  my_func(arg_1, arg_2)
```

Le keyword `def` permet la définition d'une fonction, le nom de celle-ci est suivie de parenthèses contenant les arguments de la fonction séparés par des virgules. Le keyword `return` permet de renvoyer une ou plusieurs variables.

4.5 Les librairies

4.5.1 Numpy

NumPy est la librairie la plus utilisée en Python, elle est fondamentale pour le calcul scientifique. Elle permet notamment de définir des arrays, regroupant les vecteurs et les matrices. On y retrouve également toutes les fonctions liées à l'algèbre linéaire et à l'aléatoire. Plus d'informations sur la documentation (https://numpy.org/doc/stable/).

4.5.2 Matplotlib

Matplotlib est une librairie de visualisation. Elle permet la création de graphiques, parfois intéractifs. C'est celle-ci que nous utiliserons pour afficher des nombreux résultats. Plus d'informations sur la documentation (https://matplotlib.org/stable/users/index).