Environnement Python

Un bloc de transport facile

Sommaire

- Fonction, Méthodes et Attributs
- Modules et Librairies (packages)
- Les packages pour la data sciences
- Gestion des packages et environnements avec Anaconda

Fonction, Méthodes et Attributs

Une fonction :

- C'est une portion de code à l'intérieur d'un fichier qui permet d'effectuer des tâches bien précises et ce de manière répétée.
- Elle attend généralement des objets en entrée et retourne un ou plusieurs objets.
- Python possède un nombre conséquent de fonctions « **built-in** », c'est-à-dire de fonctions intégrées au langage Python, qui sont présentes dès l'installation du langage : **print()**, **type()**,…
- Pour savoir ce qu'une fonction prend comme arguments, vous pouvez utiliser la fonction **help(ma_fonction)**.

Une méthode :

- C'est une fonction qui appartient à une class (Un type d'objet), liste, tuple, dictionnaire, ...
- Chaque class a ces méthodes qui lui sont propre. Par exemple, un objet de type list possède ses méthodes insert(), count(), etc…
- On précède le nom de la méthode par le nom de l'objet sur lequel on souhaite effectuer le traitement. Et si on souhaite ajouter des options, celles-ci seront données entre parenthèses.

Fonction, Méthodes et Attributs

Un attribut :

- Les attributs sont des variables associées à une classe d'objets. Chaque classes d'objets possèdent un ensemble d'attributs qui lui sont spécifiques.
- Contrairement à la méthode, qui est suivie de parenthèses permettant de donner des arguments supplémentaires, l'attribut, lui, ne l'est pas. Il s'agit juste d'un nom de variable qui permet de récupérer des informations sur un objet.

```
a=complex(4,3)
print(a)
dir(a)
```

```
(4+3j)

['__abs__',
 '__add__',
 '__bool__',
 '__class__',
```

```
'conjugate',
'imag',
'real']
```

```
print(a.real)
print(a.imag)
```

```
4.0
3.0
```

Modules et Librairies (packages)

Les fonctions moins courantes que les fonctions « Built-in » sont regroupées dans des modules qu'il faut importer pour pouvoir utiliser ces fonctions.

Un module :

- Est un ensemble de fichiers .py (extension de Python) contenant des fonctions, des classes ou encore des variables pouvant être appelées dans votre code.
- Si votre programme devient conséquent, il est recommandé de ranger vos fonctions dans des **fichiers .py séparés**, que vous appellerez dans votre programme principal. Se sont des modules Python.
- Si on veut importer dans son programme le code contenu dans le fichier *mon_module.py*, on importera le module avec le nom *mon_module*.

Modules et Librairies (packages)

Trois manière d'importer une fonction :

- On import tout le modules et donc toute les fonctions -> Encombre la mémoire, mais pas de risque de duplicité avec d'autre module.
- On import seulement la fonction qui nous intéresse -> Rapidité du programme, mais risque qu'une fonction porte déjà se nom.
- On peut importer toutes les fonctions d'un module. Permet de ne plus spécifier le nom de la lib avant d'appeler la fonction.

❖On peut aussi attribuer un alias à la fonction importé.

```
import math
print(math.sqrt(16))
```

```
4.0
```

```
from math import sqrt
print(sqrt(16))
```

```
4.0
```

```
from math import *
print(sqrt(16))
```

```
4.0
```

```
from math import sqrt as rc
print(rc(16))
```

```
4.0
```

Modules et Librairies (packages)

Librairies (packages) :

- Une librairie (ou package) peut être vue comme un dossier contenant un ensemble de modules, un ensemble de fichiers .py.
- Ces librairies sont créées et mises à disposition pour faciliter la programmation en Python, mais aussi pour nous éviter d'avoir à réécrire du code utilisé par un grand nombre de personnes.

Les packages pour la data sciences

- **1. NumPy**

  NumPy est le package le plus utilisé en calcul scientifique et sert de base pour de nombreux autres packages qui dépendent de lui. Ce package a été optimisé pour traiter de grands tableaux ou matrices multidimensionnelles très rapidement et propose de nombreuses méthodes clé en main pour effectuer des opérations diverses sur ces tableaux ou matrices.

- **2. Pandas**

  Pandas est un package dédié à l'analyse de données et propose une structure originale appelée **DataFrame**, très utilisée en analyse de données. Le package Pandas est basé sur NumPy. Cette librairie est fournie avec un nombre impressionnant de méthodes qui facilitent grandement le travail du Data Scientist. Le but de ce package est vraiment de permettre d'explorer, filtrer, et manipuler des données très facilement.

- **3. Matplotlib**

  Matplotlib est le package Python le plus connu pour faire de la visualisation de données. Il permet, à partir de tableaux de données, de créer divers graphiques comme des nuages de points, des boîtes à moustache, etc. Ce package est très complet et permet de paramétrer finement chaque composant du graphique.

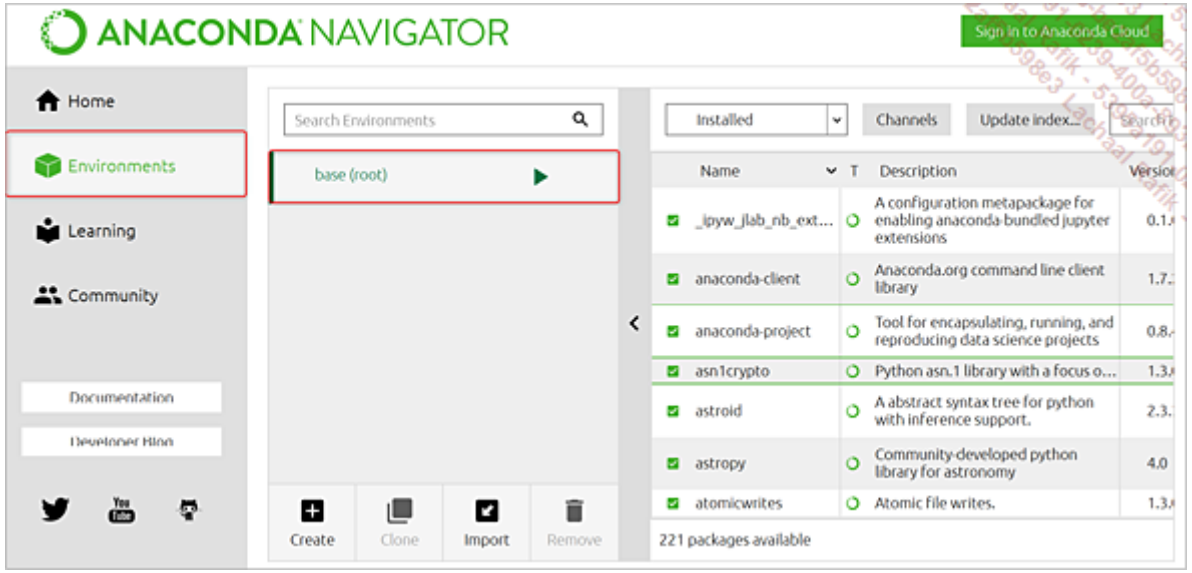
- **4. Seaborn**

  Seaborn est aussi un package Python très connu pour la visualisation de données. Il est basé sur Matplotlib et ajoute des fonctionnalités qui en font un package de plus haut niveau, permettant de créer des graphiques plus complexes.

Gestion des packages et environnements avec Anaconda

En plus de faciliter le lancement des applications, Anaconda Navigator facilite l'installation et la gestion des packages et environnements.

Cette partie de gestion est disponible sous l'onglet Environnements.



Gestion des packages et environnements avec Anaconda

- Un environnement sous Anaconda correspond à un ensemble de packages installés, avec des versions particulières.
- En effet, certains packages nécessitent des versions spécifiques d'autres packages pour pouvoir fonctionner. Il est alors possible de créer des environnements différents, avec des versions de packages différentes, et de choisir son environnement avant de commencer son analyse.
- Par défaut, un environnement existe sous Anaconda Navigator, qui est appelé base (root).

Pour créer et gérer ses environnements sous Anaconda on utilise le gestionnaire de paquets **Conda**.

Voici un lien vers un tutoriel Conda :

https://zestedesavoir.com/tutoriels/1448/installer-un-environnement-de-developpement-python-avec-conda/

Testé les commandes proposées. Vous pourrez les tester avec le packet « numpy ».