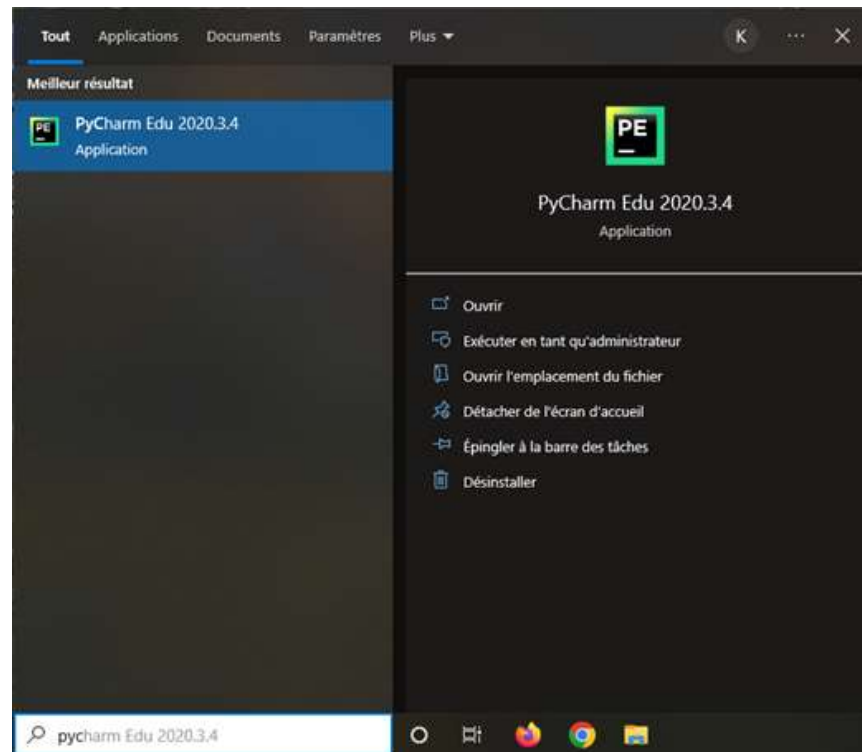


# Guide de démarrage de l'IDE Pycharm (version 2020.3.4 )

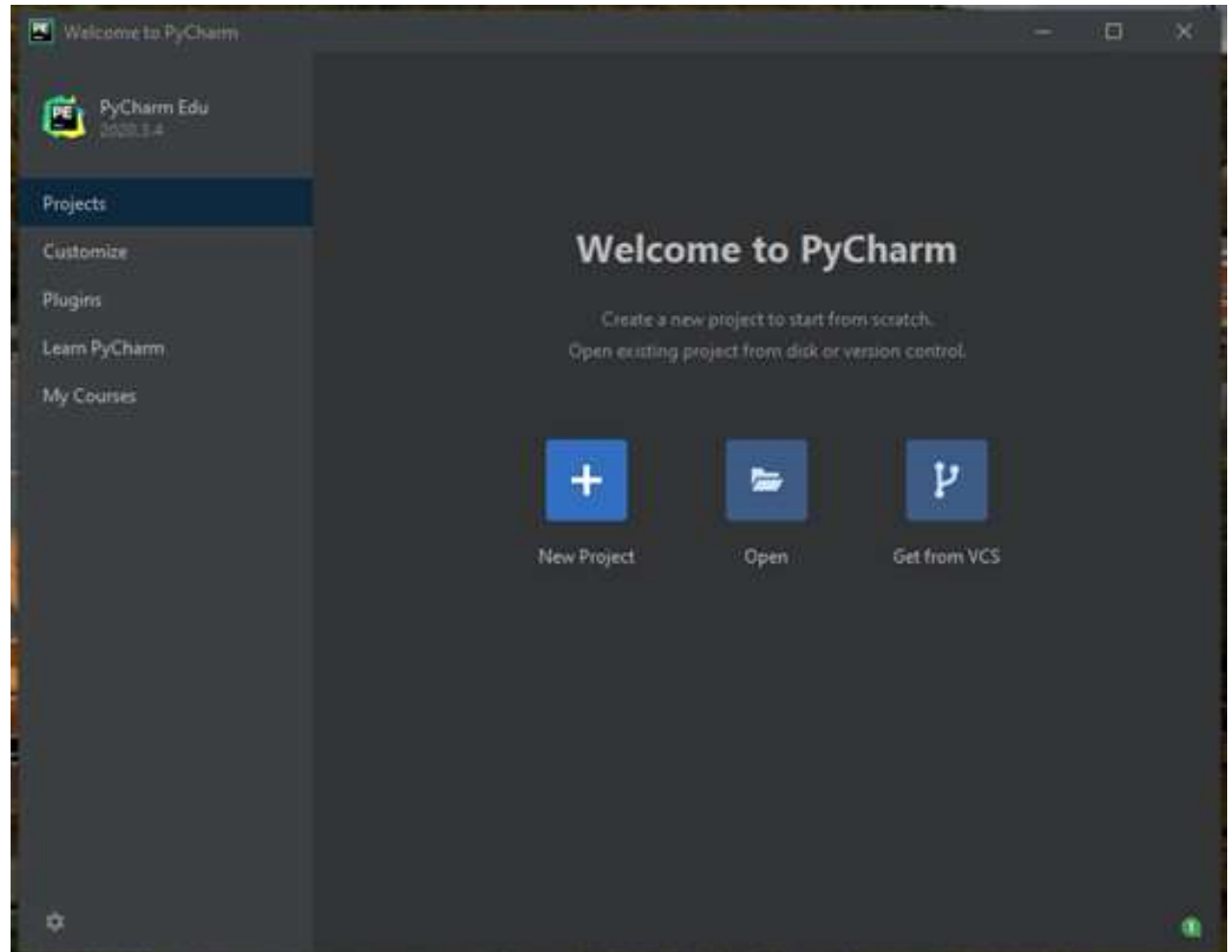
APP2- S2

Par Serge A. Kodjo, ing. Ph.D

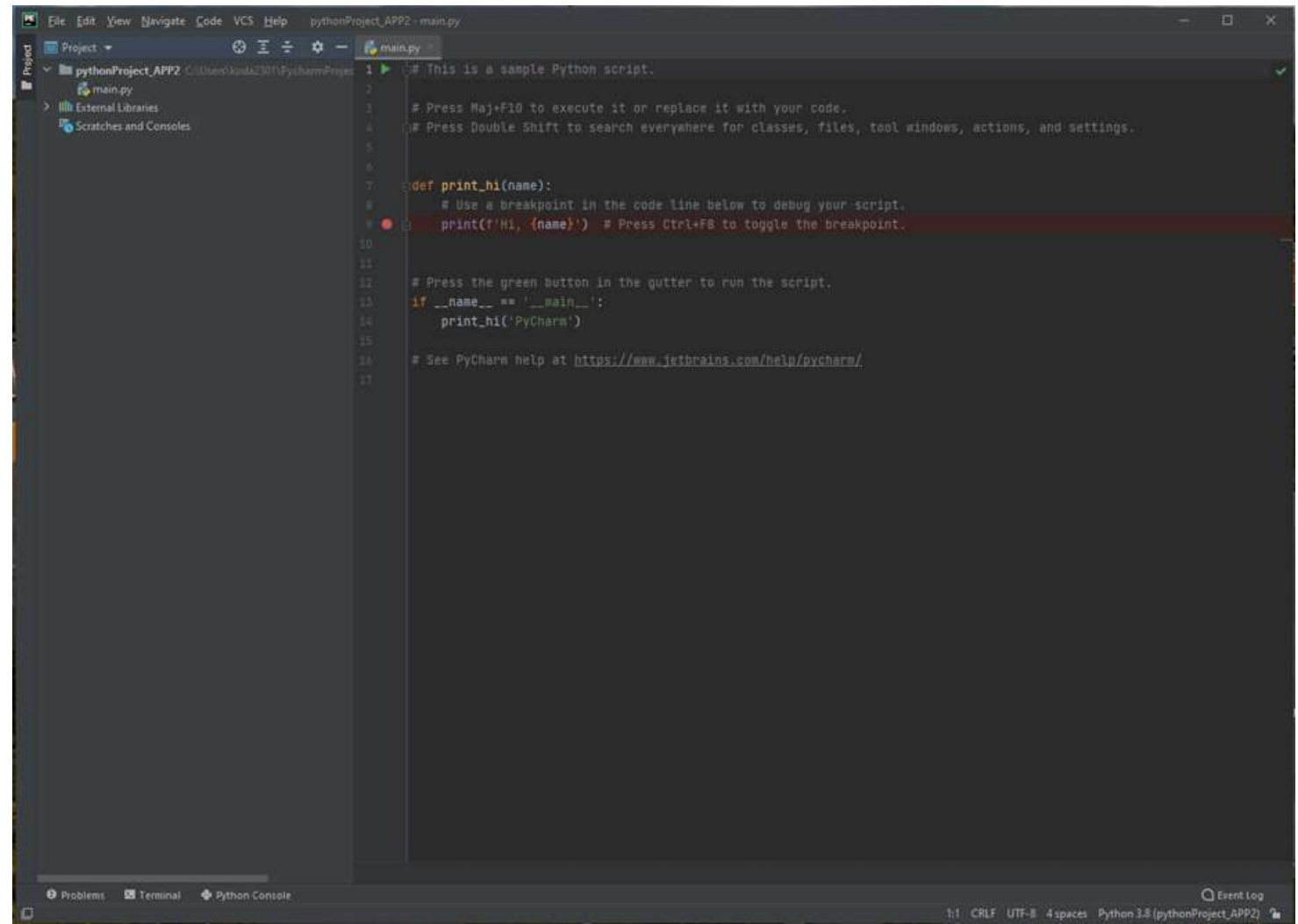
On ouvre l'application PyCharm



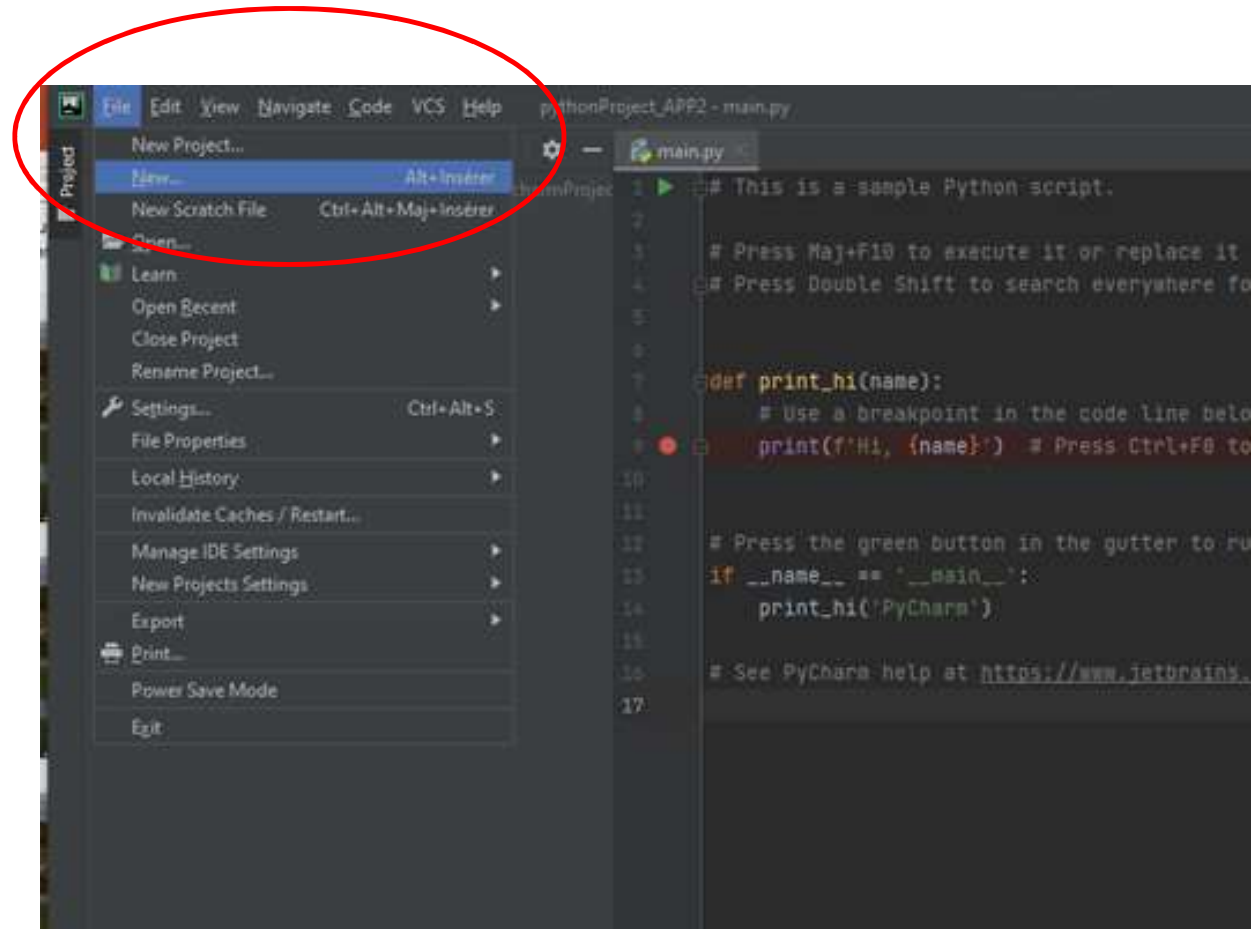
On crée un Nouveau projet.



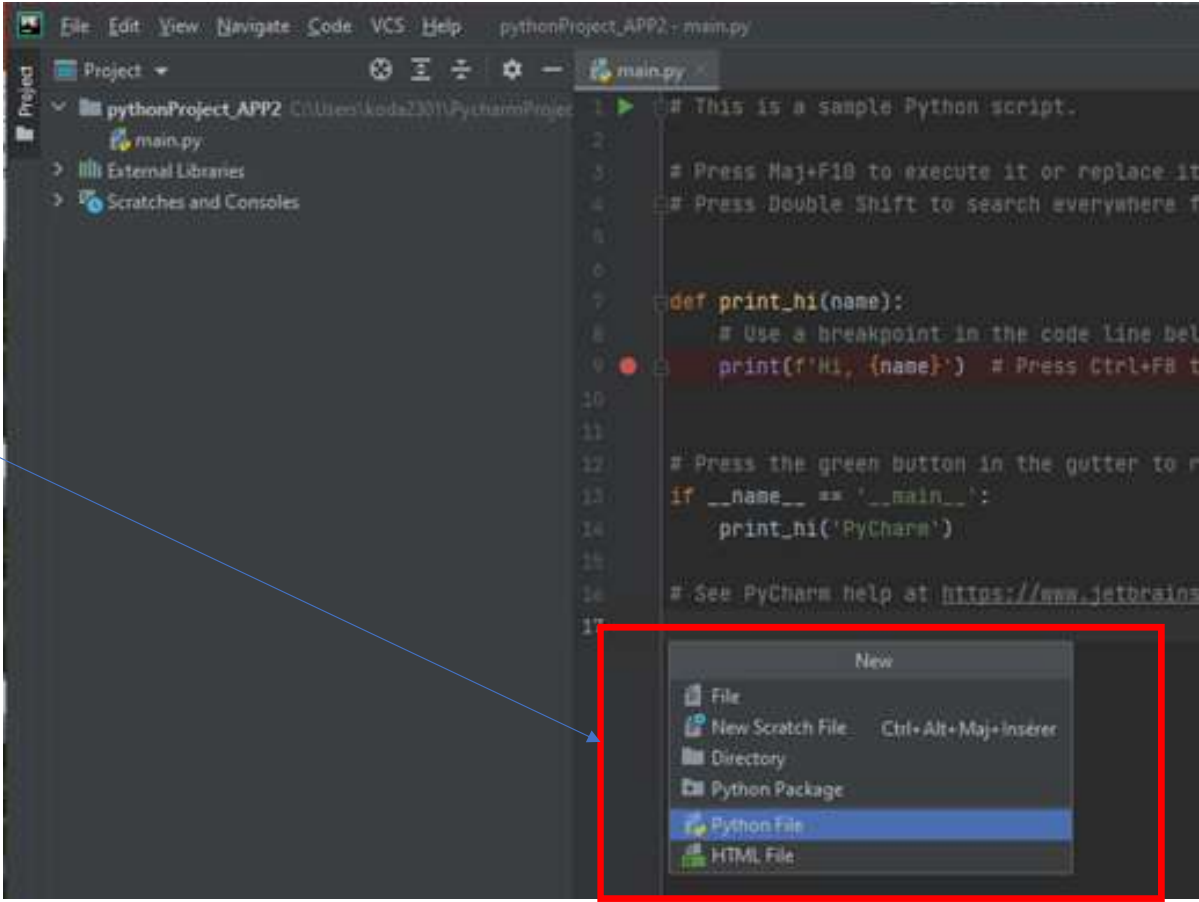
Dans le projet créé,  
un fichier Python  
du nom « main.py »  
est créé par défaut



À partir de : → File → New,  
créer un nouveau fichier .py  
dans lequel vous allez écrire  
votre code.

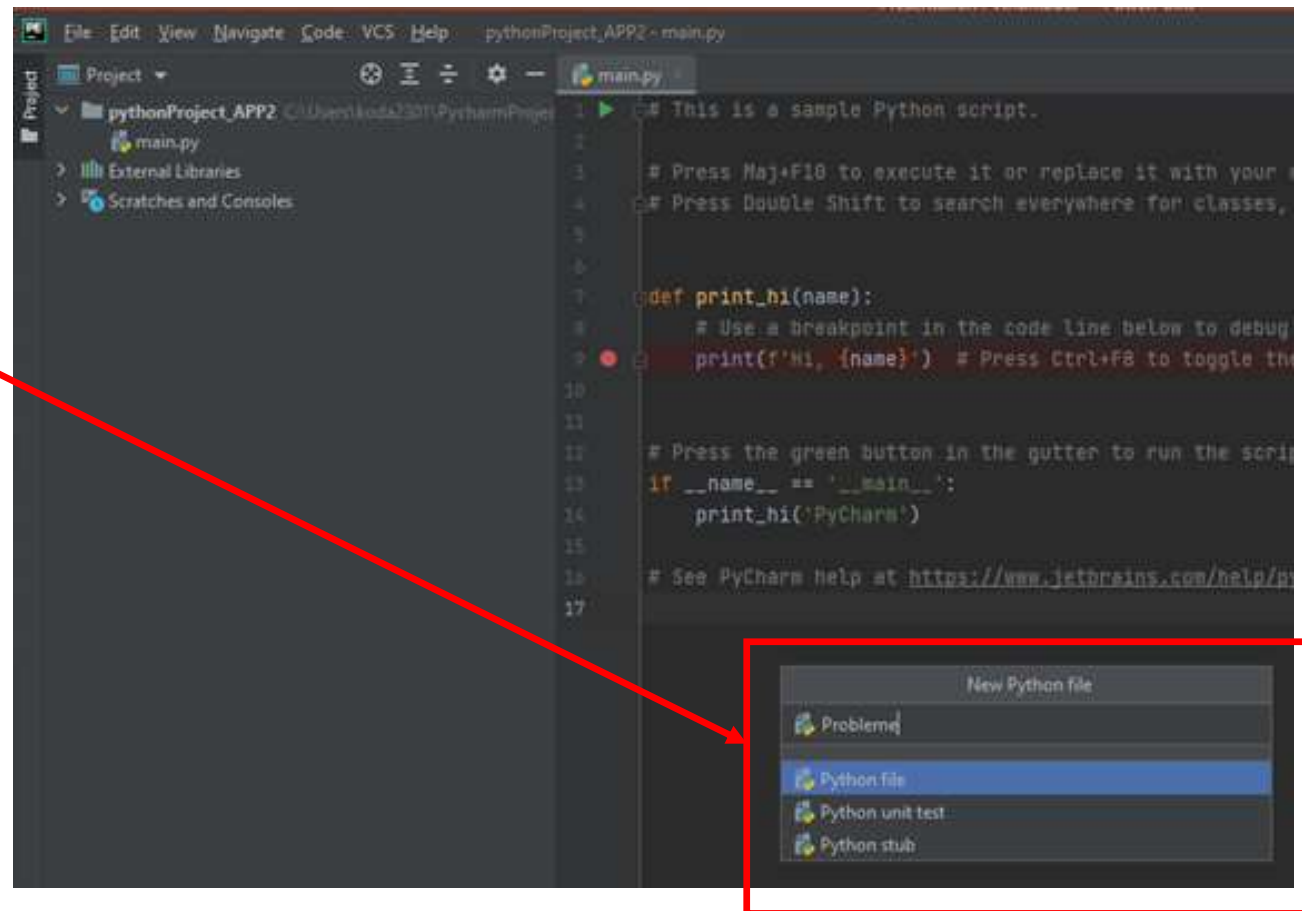


Choisissez Python File

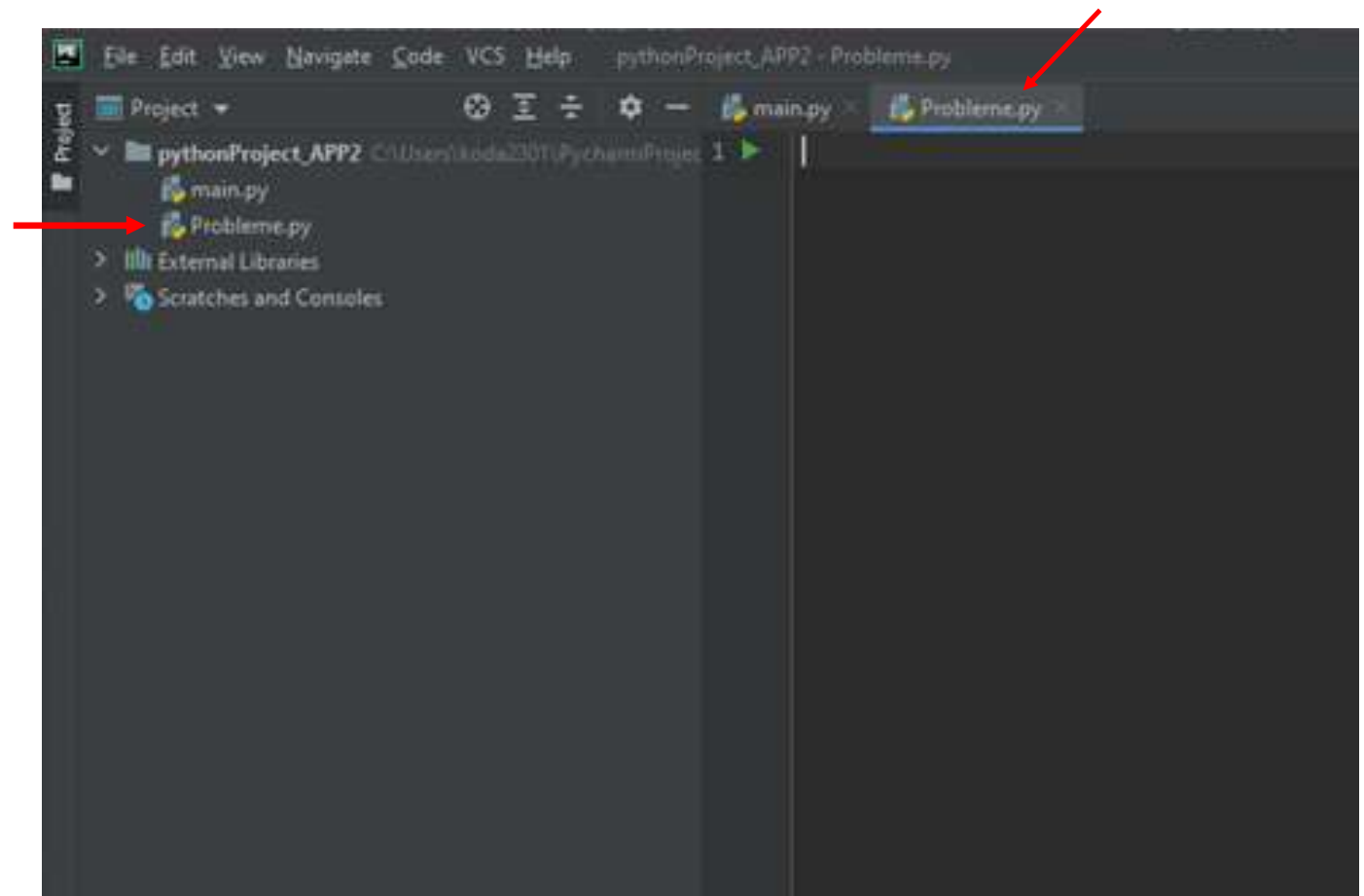


On crée le nouveau fichier.py  
en évitant les accents dans  
les noms du fichiers

Exemple: probleme.py



Le Fichier est créé

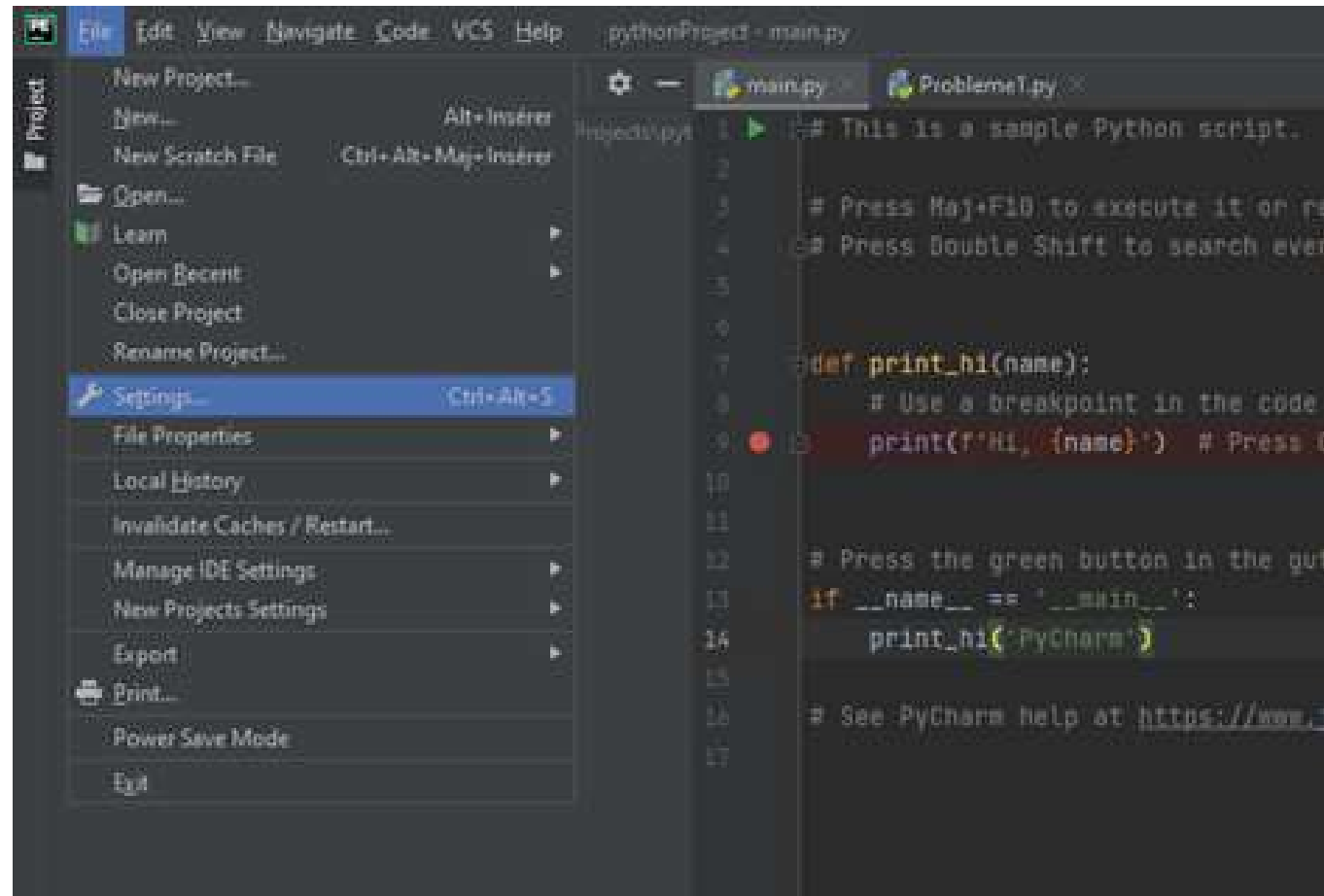




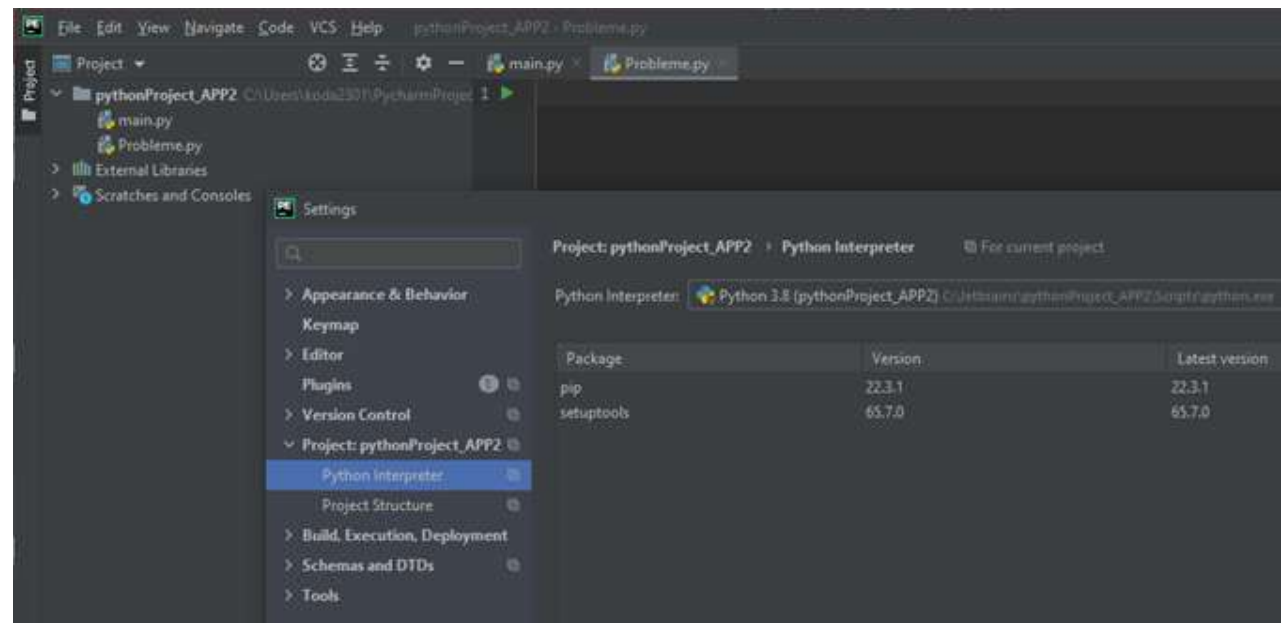
Ajouter les librairies dans votre espace de travail pour pouvoir les importer dans vos codes

- Dans l'APP2 nous avons besoin des librairies numpy et matplotlib.
  - <https://numpy.org/doc/stable/user/index.html#user>
  - <https://matplotlib.org/>

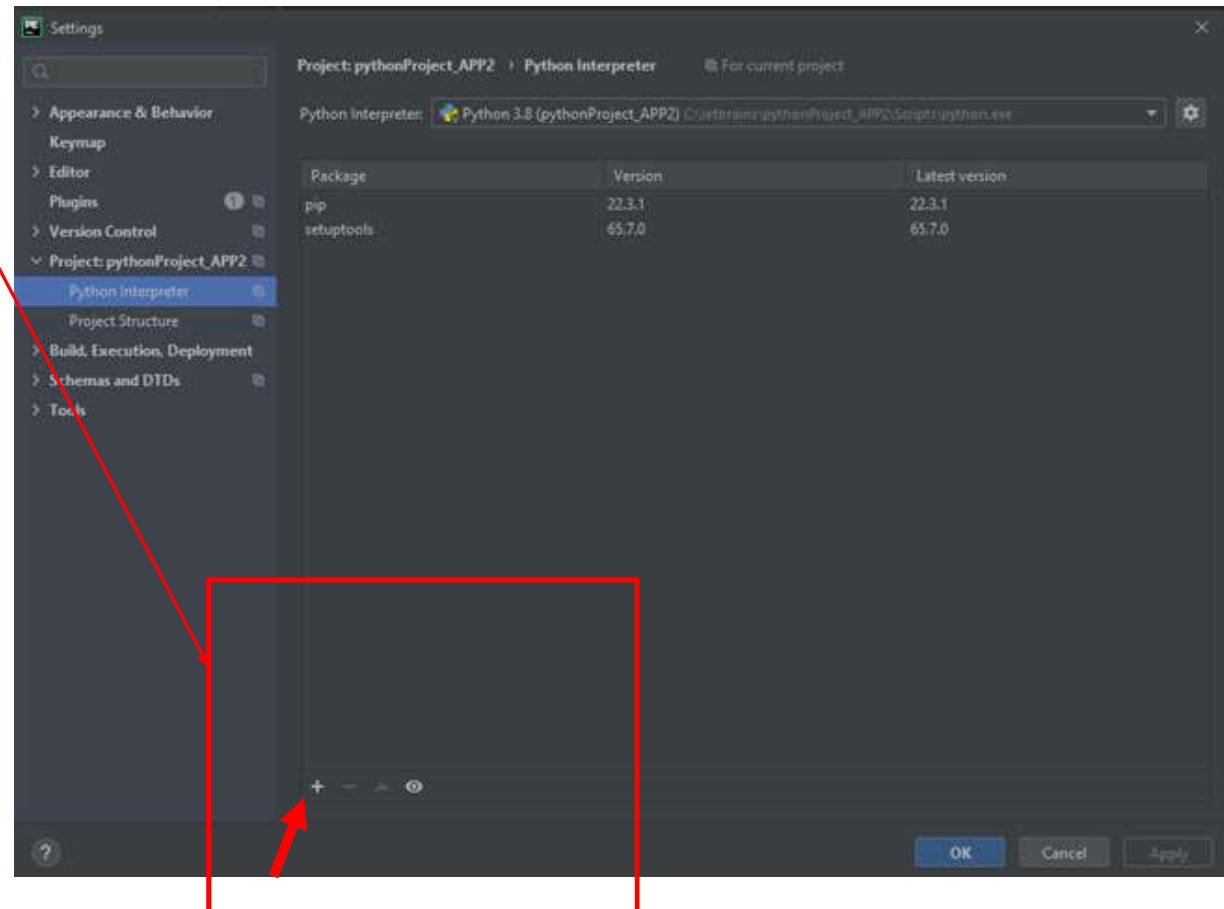
Dans « File », allez dans « settings »



Dans « settings », choisissez « Python interpreter ». On voit afficher les librairies qui sont installées dans l'environnement du projet. On remarque que les librairies numpy et matplotlib ne sont pas installées.



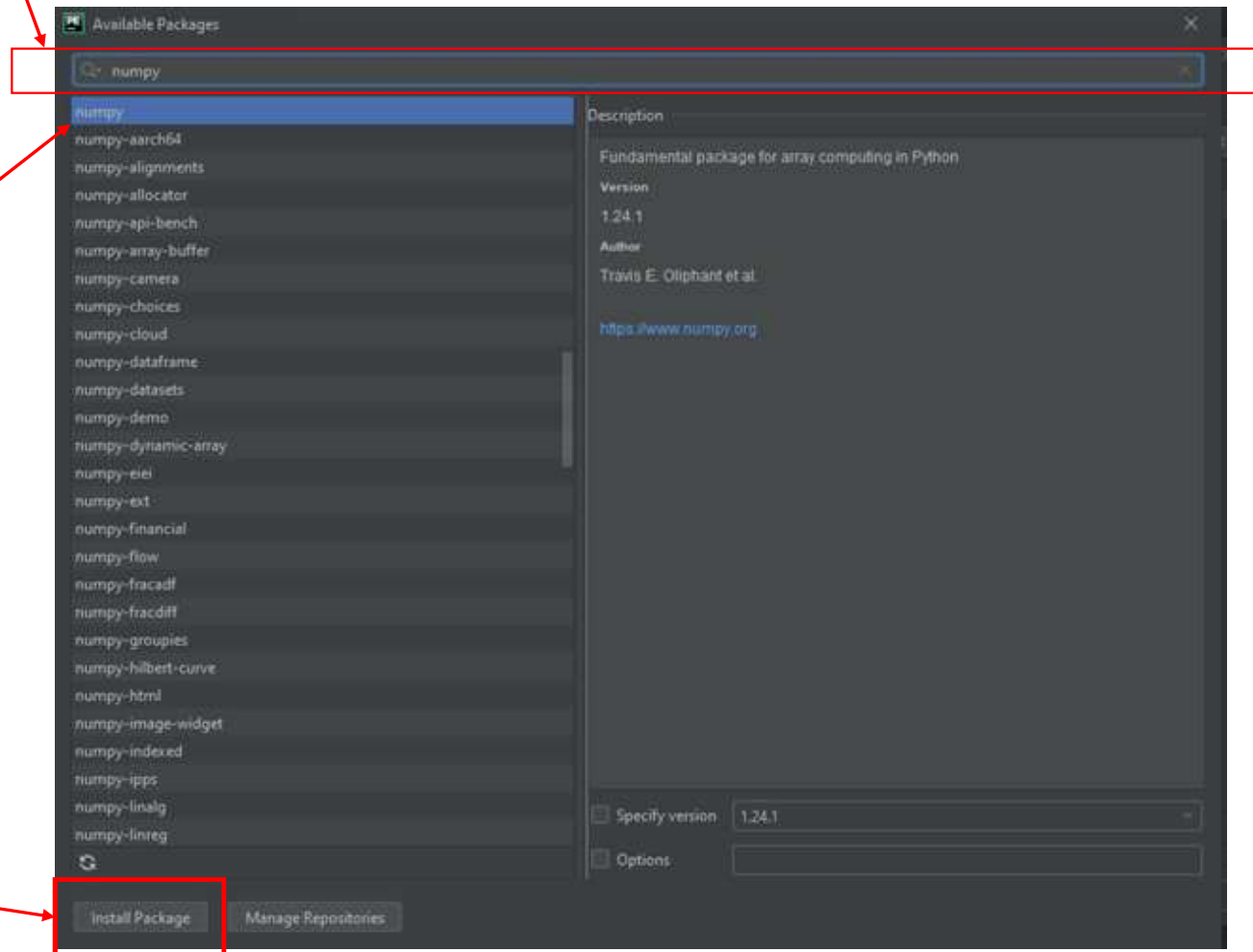
Pour ajouter une nouvelle librairie, on clique sur « + »



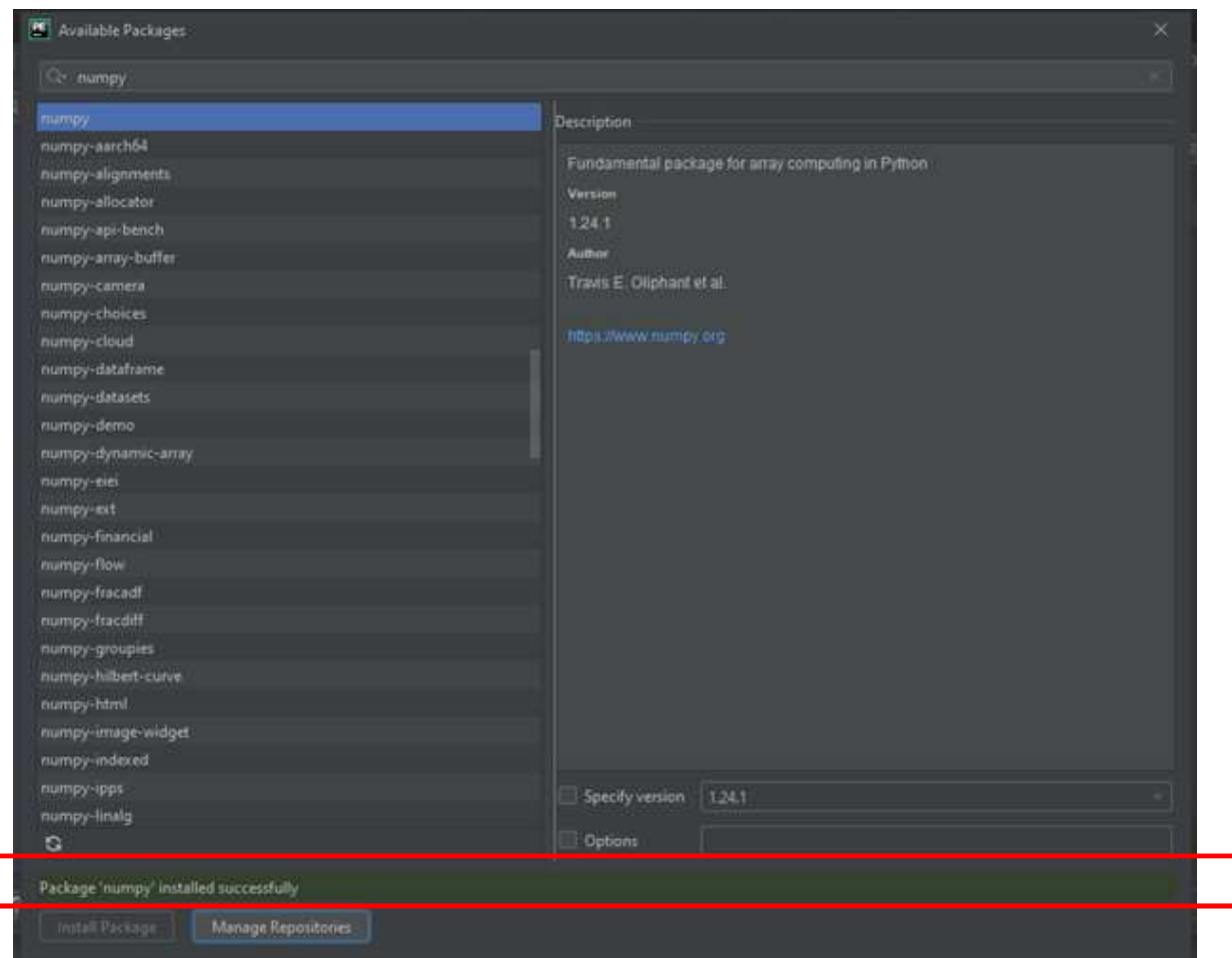
1) On cherche le nom de la librairie ici

2) On la sélectionne dans la liste

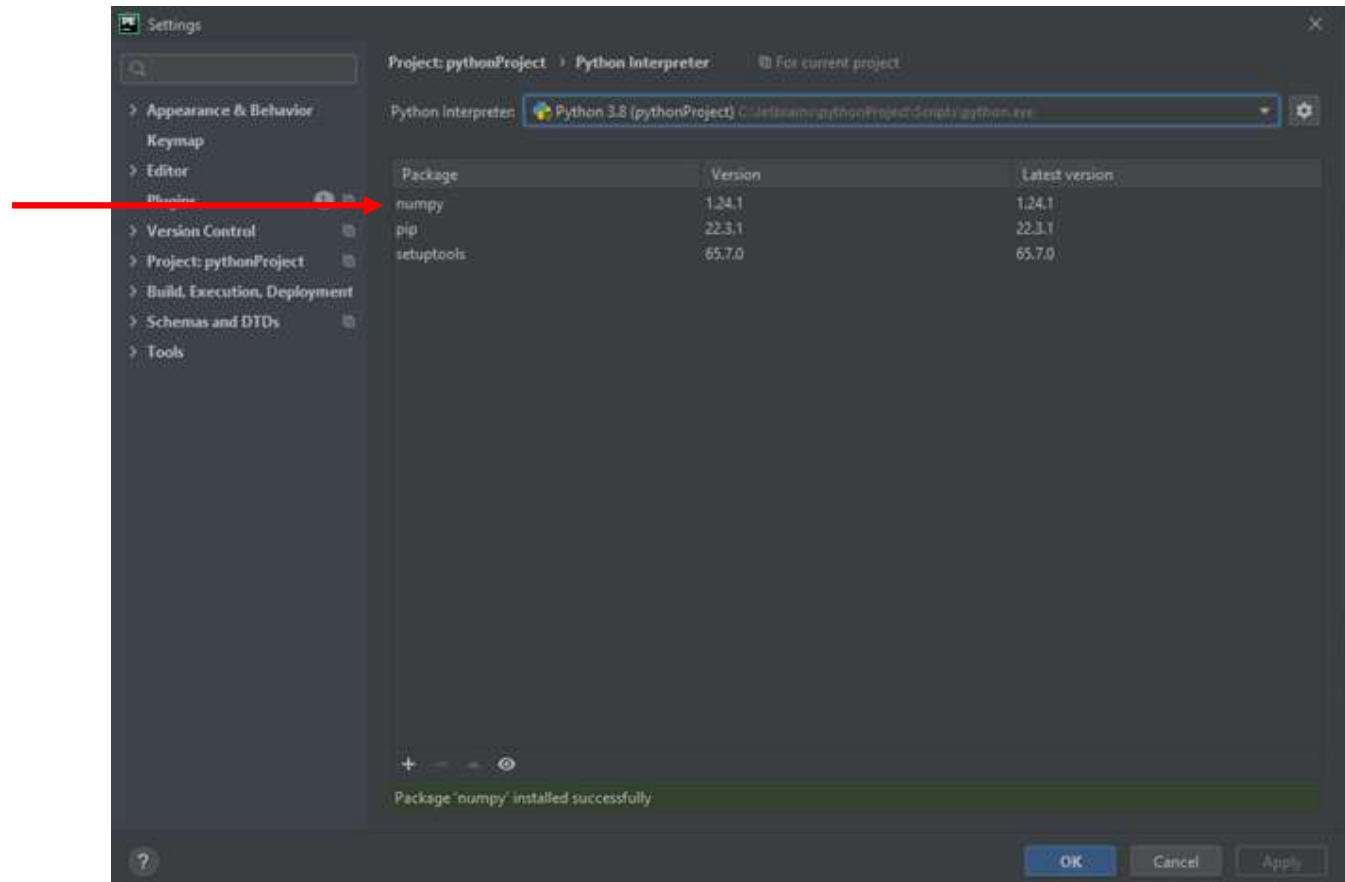
3) Et on l'installe



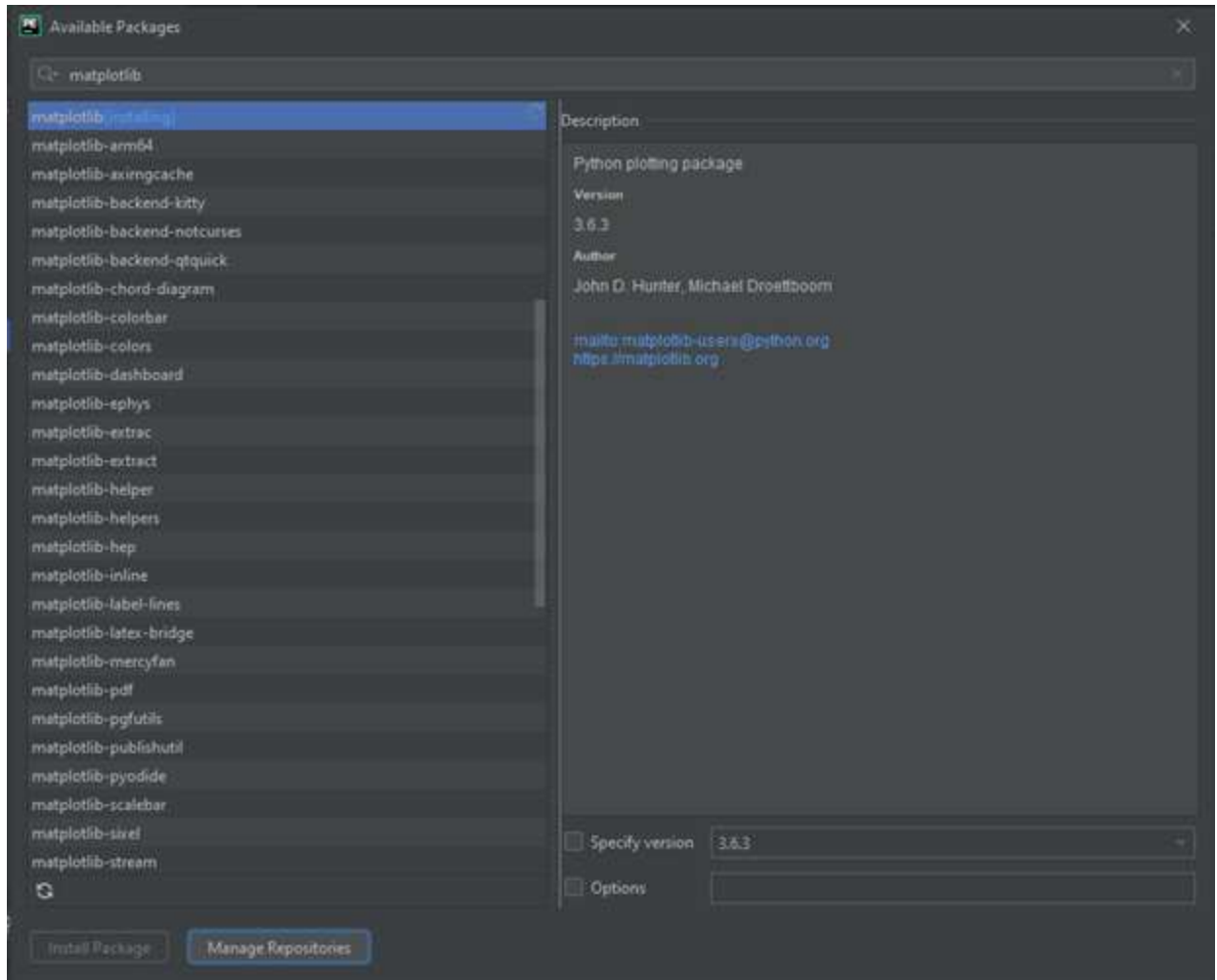
Une fois installée, vous avez ce message qui vous confirme que la librairie est bien installée.



On le voit maintenant  
apparaître dans la liste

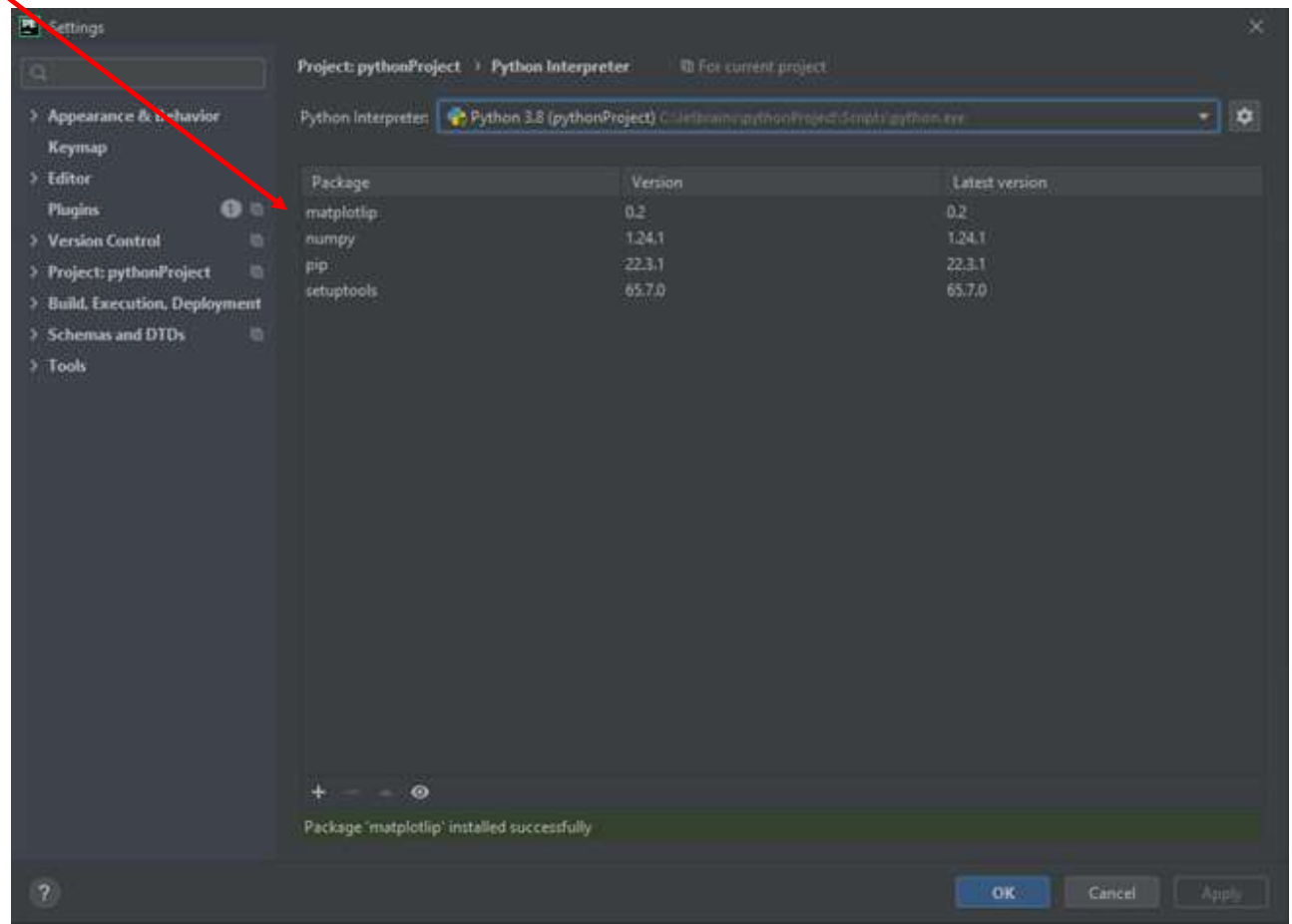


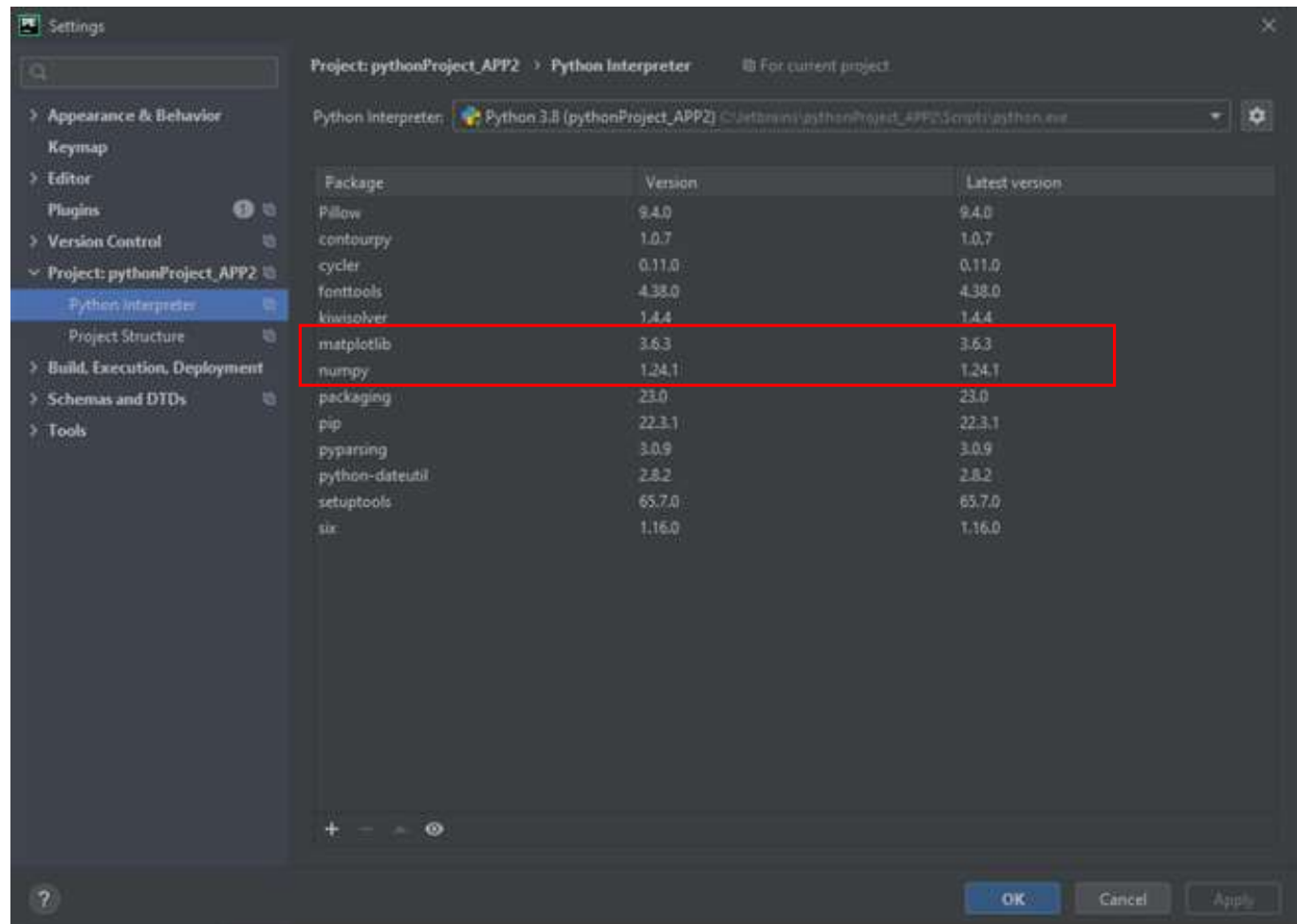
On recommence les mêmes étapes pour installer matplotlib





On peut vérifier que c'est bien installé

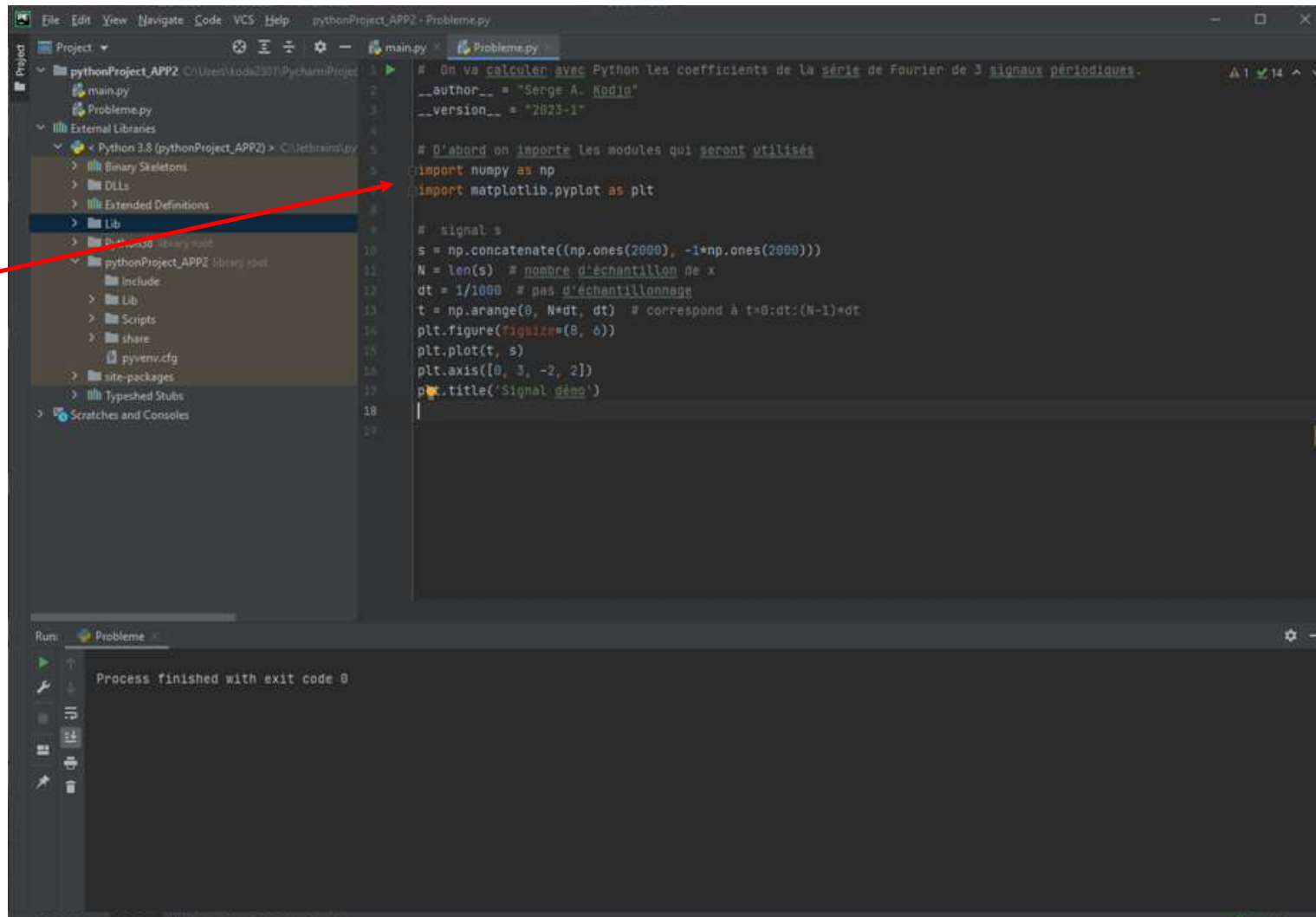




On écrit notre code dans le nouveau fichier. Au début du code on importe les librairies dont on a besoin.

Numpy est importé sous le nom raccourci « np » et matplotlib.pyplot est importé sous le nom (raccourci) « plt ». On peut maintenant appeler les fonctions de ces librairies dans notre code, en utilisant leur nom raccourci.

Exemple: la fonction `arange()` de la librairie `numpy` est appelée par `np.arange()`

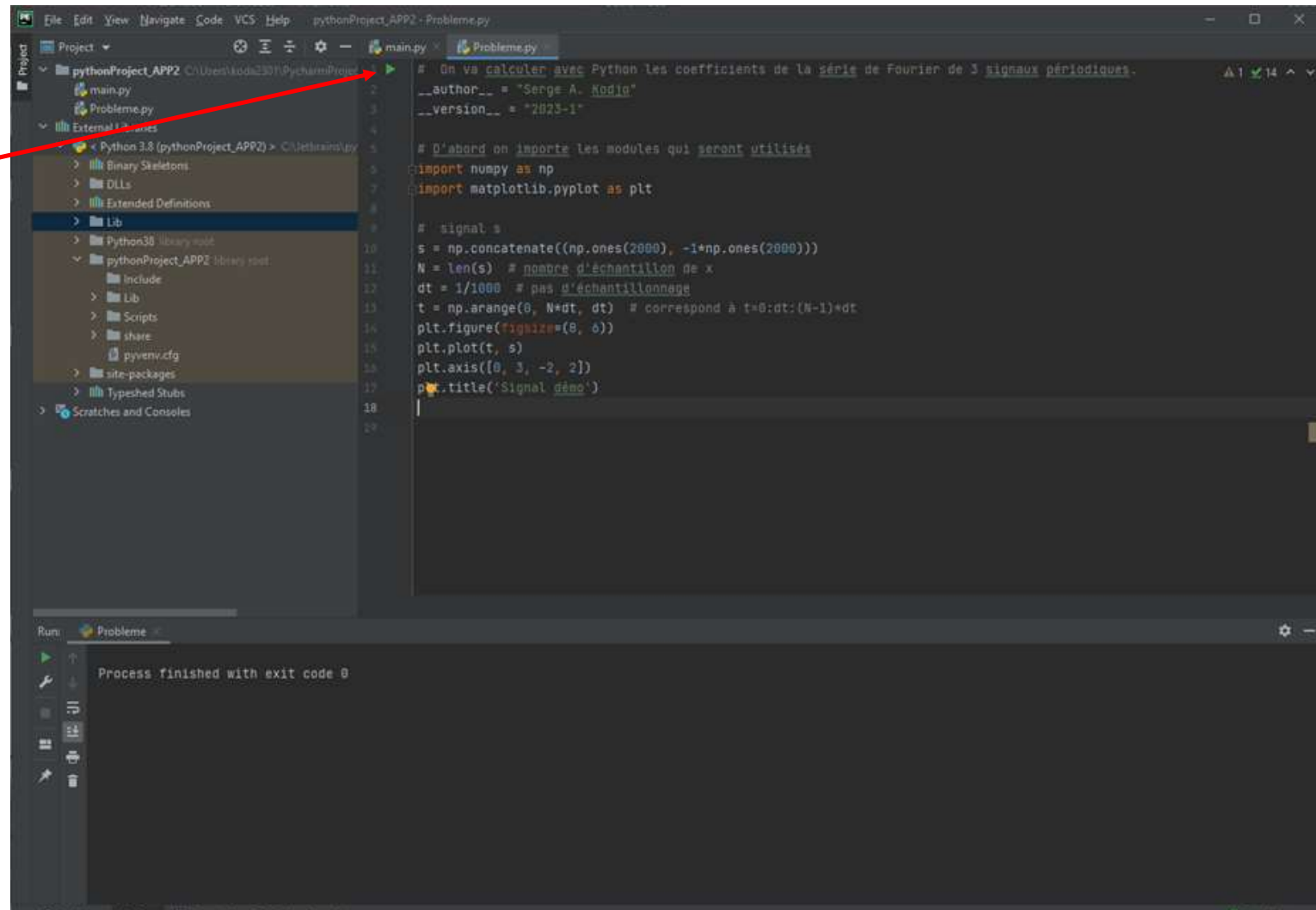


The screenshot shows an IDE window titled 'pythonProject\_APP2 - Probleme.py'. The left sidebar displays a project tree with 'pythonProject\_APP2' containing 'main.py' and 'Probleme.py'. The 'External Libraries' section is expanded, showing 'Python 3.8 (pythonProject\_APP2)' and its subfolders: 'Binary Skeletons', 'DLLs', 'Extended Definitions', 'Lib', 'Python38 library root', 'pythonProject\_APP2 library root', 'Include', 'Scripts', 'share', 'pyvenv.cfg', 'site-packages', 'Typeshed Stubs', and 'Scratches and Consoles'. A red arrow points from the 'Lib' folder to the code editor. The code editor shows the following Python code:

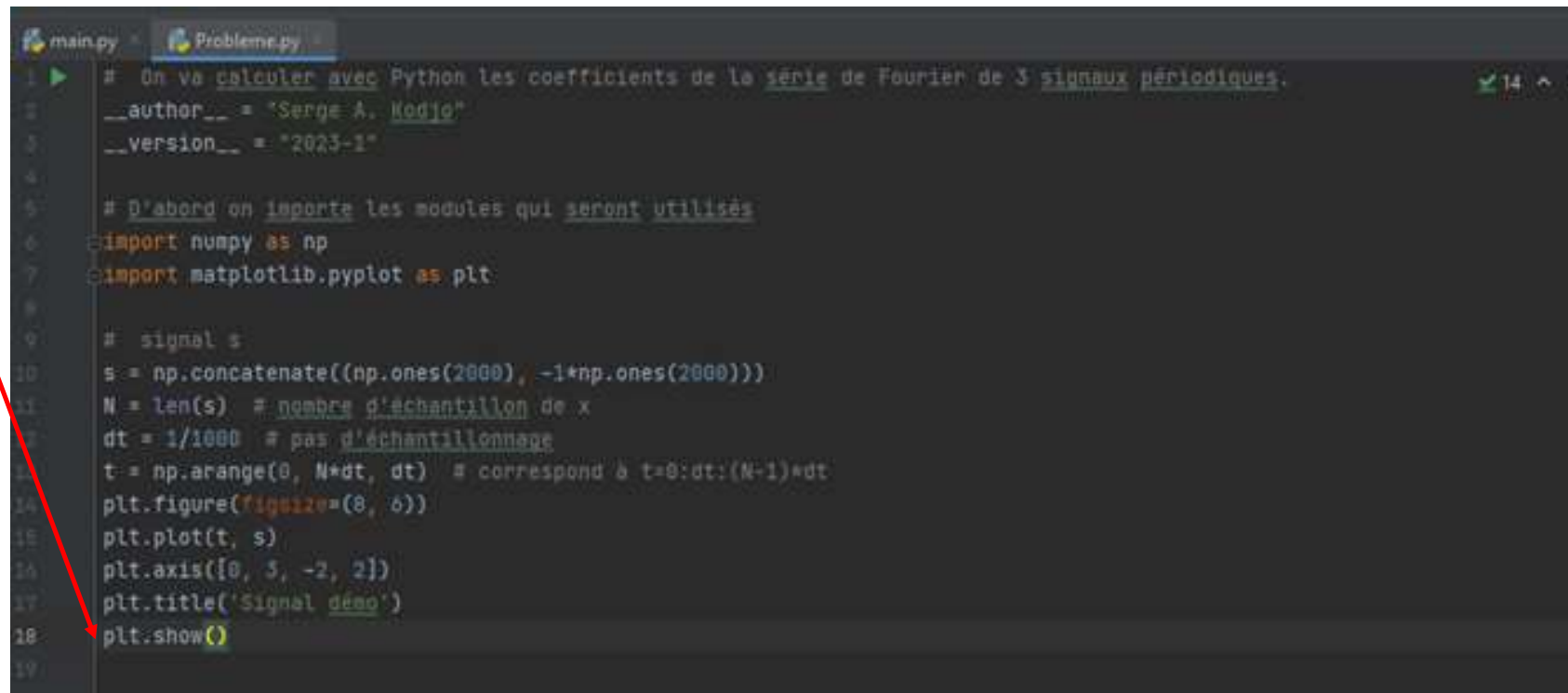
```
1 # On va calculer avec Python les coefficients de la série de Fourier de 3 signaux périodiques.
2 __author__ = "Serge A. Kodja"
3 __version__ = "2023-1"
4
5 # D'abord on importe les modules qui seront utilisés
6 import numpy as np
7 import matplotlib.pyplot as plt
8
9 # signal s
10 s = np.concatenate((np.ones(2000), -1*np.ones(2000)))
11 N = len(s) # nombre d'échantillon de x
12 dt = 1/1000 # pas d'échantillonnage
13 t = np.arange(0, N*dt, dt) # correspond à t=0:dt:(N-1)*dt
14 plt.figure(figsize=(8, 6))
15 plt.plot(t, s)
16 plt.axis([0, 3, -2, 2])
17 plt.title('Signal démo')
18
19
```

The bottom panel shows the 'Run' output with the message 'Process finished with exit code 0'.

On exécute ensuite le code en cliquant sur le triangle vert ici.

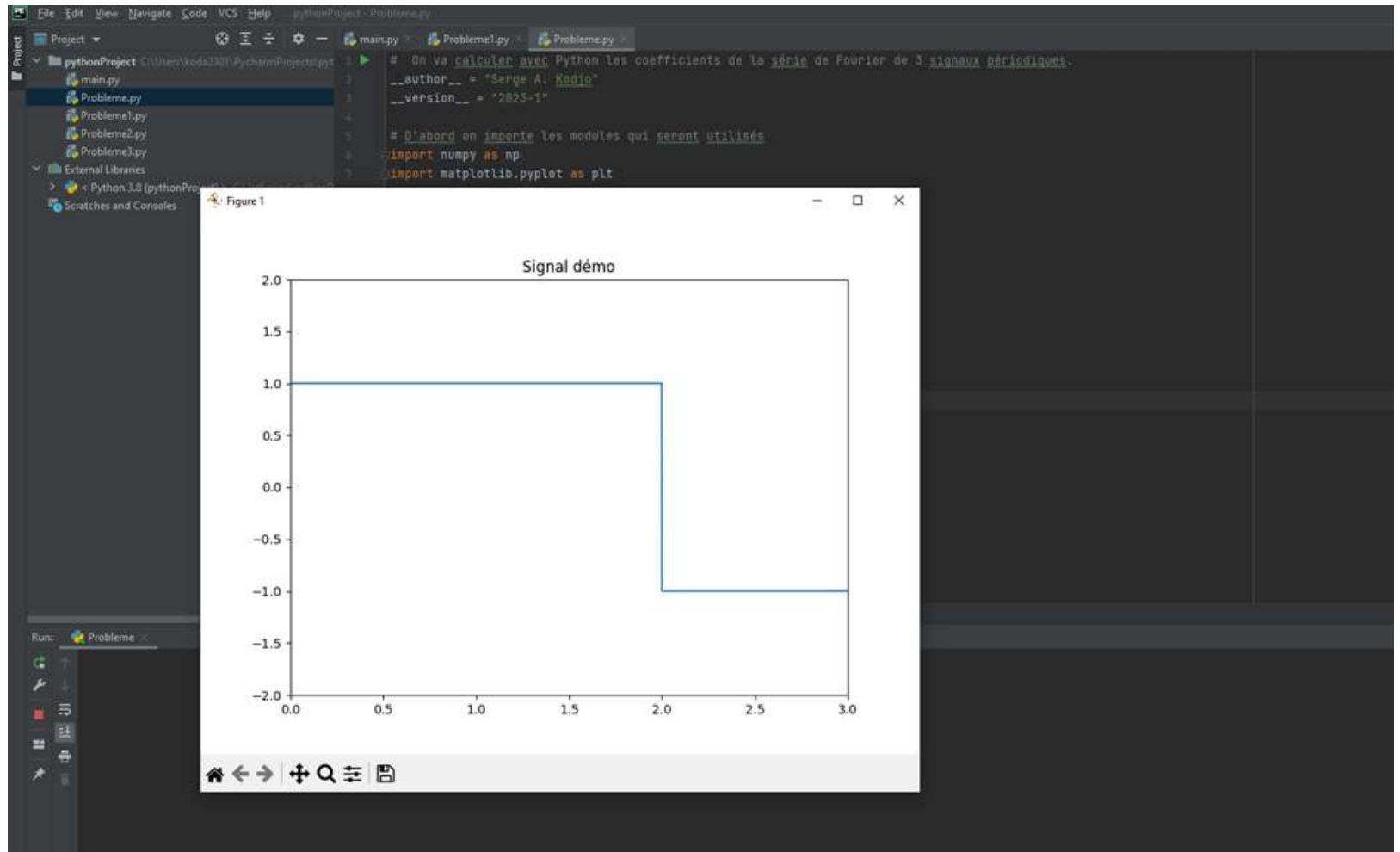


Pour afficher les figures, on a besoin de la fonction `show()` de matplotlib

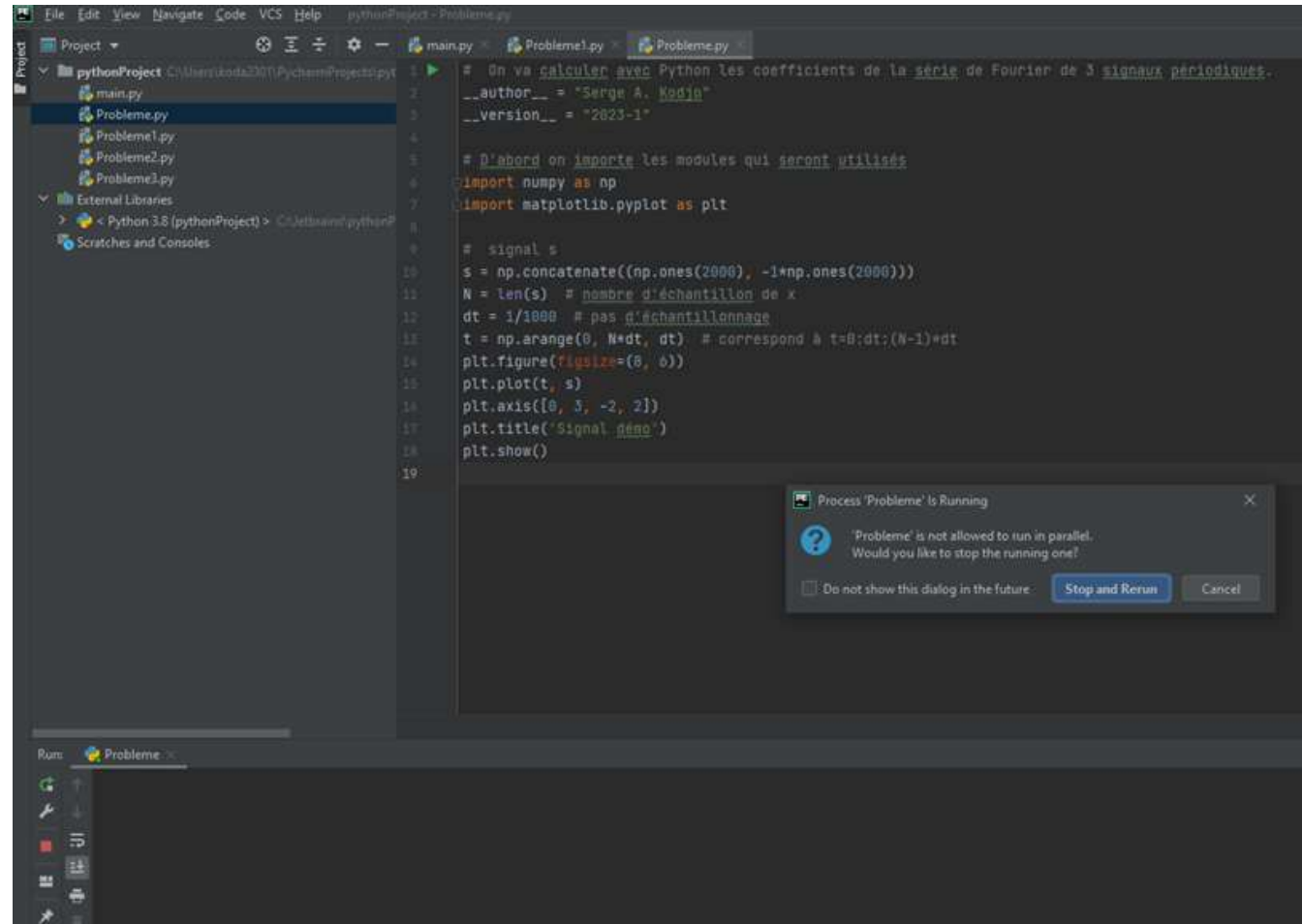


```
main.py - Probleme.py
1 # On va calculer avec Python les coefficients de la série de Fourier de 3 signaux périodiques.
2 __author__ = "Serge A. Kodjo"
3 __version__ = "2023-1"
4
5 # D'abord on importe les modules qui seront utilisés
6 import numpy as np
7 import matplotlib.pyplot as plt
8
9 # signal s
10 s = np.concatenate((np.ones(2000), -1*np.ones(2000)))
11 N = len(s) # nombre d'échantillon de x
12 dt = 1/1000 # pas d'échantillonnage
13 t = np.arange(0, N*dt, dt) # correspond à t=0:dt:(N-1)*dt
14 plt.figure(figsize=(8, 6))
15 plt.plot(t, s)
16 plt.axis([0, 3, -2, 2])
17 plt.title('Signal démo')
18 plt.show()
19
```

On obtient ceci après l'exécution du code dans cet exemple. On voit la figure affichée.



Mais le code est toujours en exécution tant que toutes les figures ne sont pas fermé



## Les opérateurs arithmétiques dans Python

Opérateur	Nom
+	Addition
−	Soustraction
*	Multiplication
/	Division
%	Modulo
**	Puissance
//	Division entière