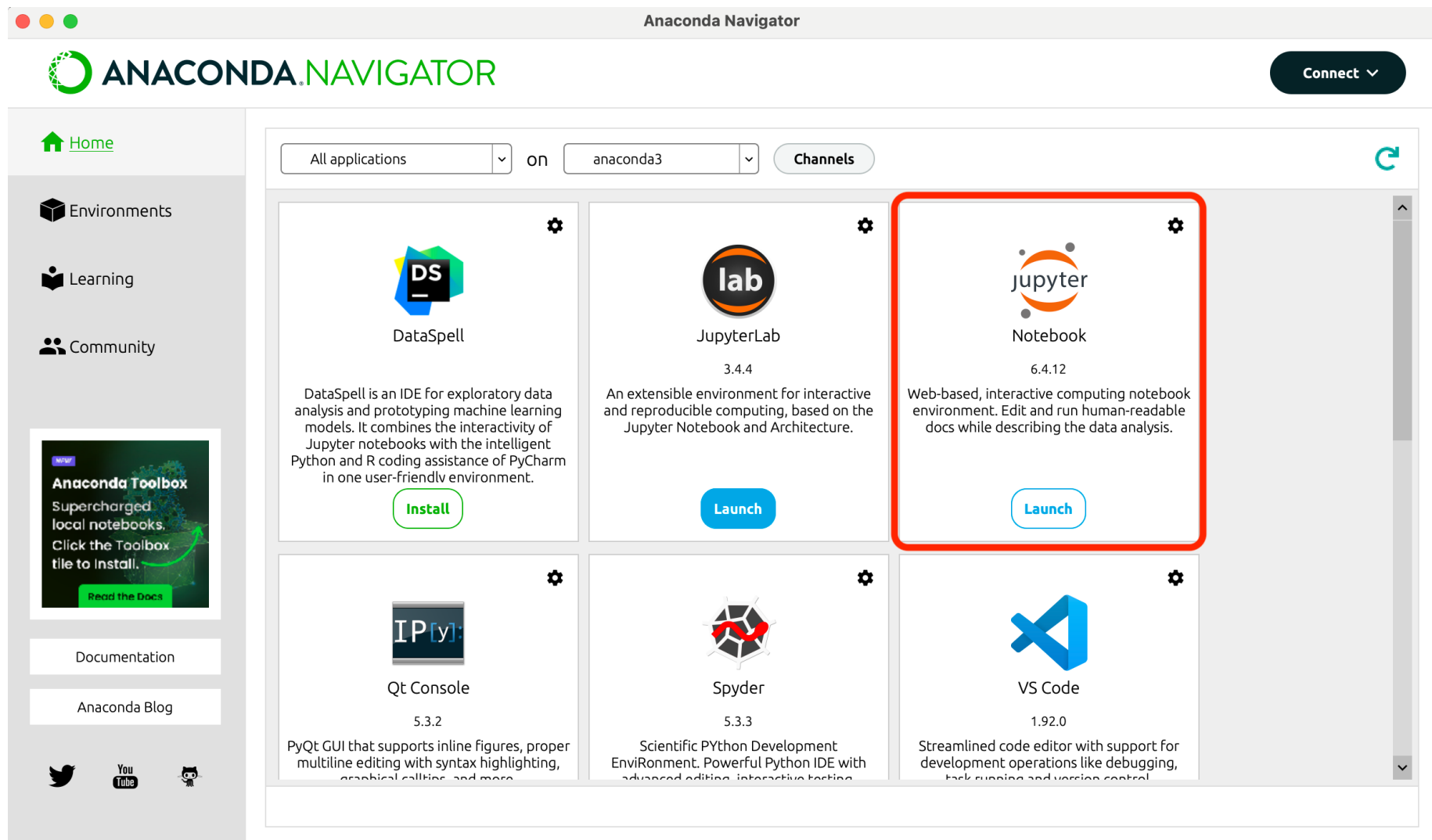


Installation



Application : Jupyter Notebook



Ouverture de fichier .ipynb

1. Sauvegarder l'énoncé de TPO : Téléchargez et enregistrez l'énoncé de TPO à l'emplacement de votre choix. Par exemple, vous pouvez créer un dossier appelé MATH203 dans votre dossier Documents.
2. Accéder à l'emplacement enregistré : Dans l'explorateur de fichiers Jupyter accédez à l'emplacement où vous avez sauvegardé le fichier.
3. Lancer le fichier .ipynb : Dans l'arborescence des fichiers, sélectionnez le fichier notebook Jupyter et ouvrez-le.



Quit

Se déconnecter

Fichiers

Actifs

Grappes

Sélectionner des éléments pour leur appliquer des actions.

Téléverser

Nouveau ▾


☐ 0 ▾ /

Nom ▾

Dernière modification

File size

☐ bin

il y a un an

☐ Desktop

il y a 3 heures

☐ Documents

il y a 2 jours

☐ Downloads

il y a une heure

Chaque programme python commence par l'importation des bibliothèques

1. Cliquez sur la cellule contenant des importations. La cellule sera entourée d'un cadre, indiquant qu'elle est sélectionnée.
2. Exécuter la cellule :
 - Cliquez sur le bouton Exécuter dans la barre d'outils (représenté par une icône en forme de triangle ou de flèche).
 - Ou utilisez le raccourci clavier Shift + Entrée pour exécuter la cellule et passer automatiquement à la suivante.
 - Vous pouvez aussi utiliser Ctrl + Entrée pour exécuter sans passer à la cellule suivante.



exécuter la cellule, sélectionner la suivante

Pour vraiment commencer : Importation des bibliothèques de Python

Les bibliothèques **Numpy** et **Matplotlib** sont les plus souvent utilisées en Python. La bibliothèque **Numpy** utilise un large éventail de fonctions mathématiques qui le rendent particulièrement utile pour des calculs scientifiques et problèmes de mathématiques appliquées. Le module **Matplotlib** permet de tracer des fonctions et d'afficher leurs courbes dans des graphiques.

Le programme ci-dessous permet à importer ces bibliothèques en fin de pouvoir les utiliser

```
Entrée [ ]: # importation des fonctions mathématiques, on peut maintenant utiliser les fonction sin, exp, la valeur de pi, etc.
from math import *

# on importe la bibliothèque numpy qui permet de travailler avec des tableaux
import numpy as np

# on importe la module matplotlib.pyplot qui permet tracer des fonctions
import matplotlib.pyplot as plt
```

La cellule affichera une indication Entrée [1], signifiant qu'elle a bien été exécutée.



Fichier Édition Affichage Insérer Cellule Noyau Widgets Aide Fiable Python 3 (ipykernel)

Pour vraiment commencer : Importation des bibliothèques de Python

Les bibliothèques **Numpy** et **Matplotlib** sont les plus souvent utilisées en Python. La bibliothèque **Numpy** utilise un large éventail de fonctions mathématiques qui le rendent particulièrement utile pour des calculs scientifiques et problèmes de mathématiques appliquées. Le module **Matplotlib** permet de tracer des fonctions et d'afficher leurs courbes dans des graphiques.

Le programme ci-dessous permet à importer ces bibliothèques en fin de pouvoir les utiliser

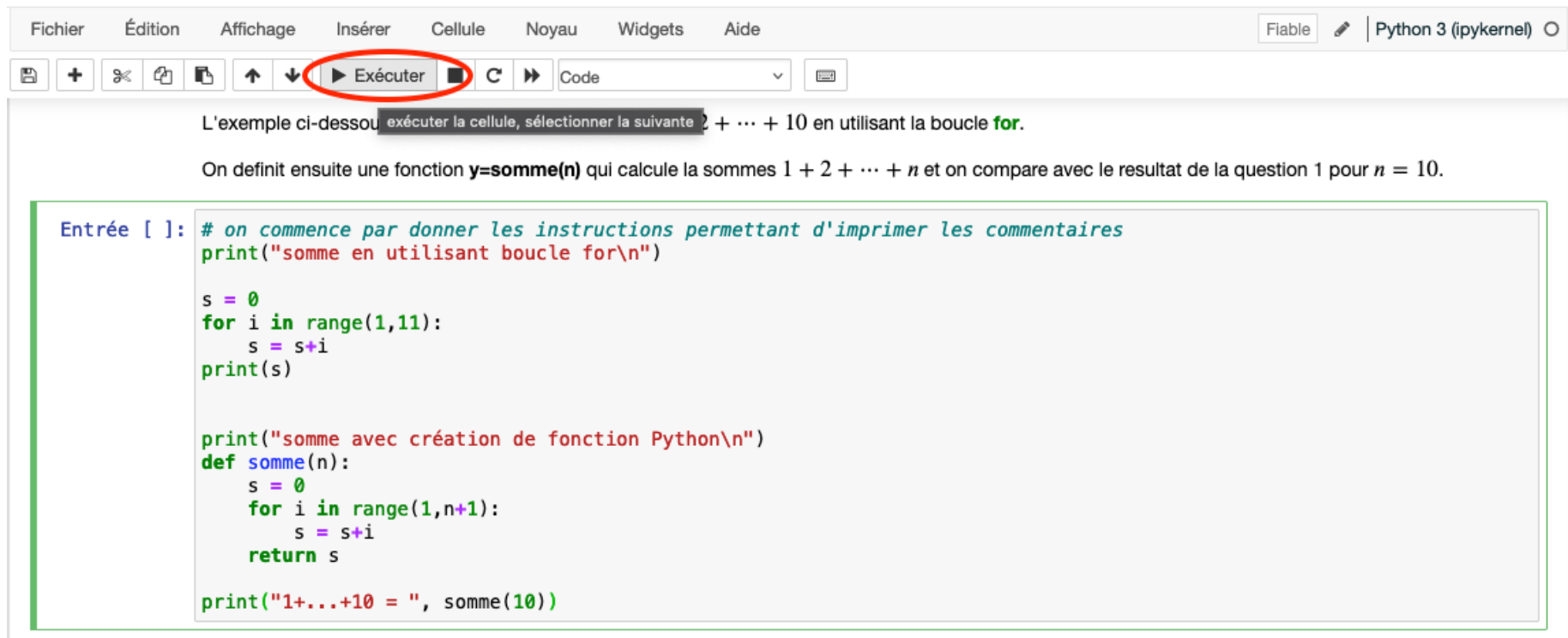
```
Entrée [1]: # importation des fonctions mathématiques, on peut maintenant utiliser les fonction sin, exp, la valeur de pi, etc.
from math import *

# on importe la bibliothèque numpy qui permet de travailler avec des tableaux
import numpy as np

# on importe la module matplotlib.pyplot qui permet tracer des fonctions
import matplotlib.pyplot as plt
```

Exécution d'une cellule avec print

Exécutez les cellules contenant des commandes print. Vous pouvez utiliser le raccourci clavier Shift + Entrée ou Ctrl + Entrée.



The screenshot shows a Jupyter Notebook interface. The top menu bar includes 'Fichier', 'Édition', 'Affichage', 'Insérer', 'Cellule', 'Noyau', 'Widgets', and 'Aide'. The 'Cellule' menu is open, and the 'Exécuter' button (a play icon) is highlighted with a red circle. Below the menu bar, there is a toolbar with various icons, including a play icon. The main area of the notebook contains a code cell with the following text:

L'exemple ci-dessous exécuter la cellule, sélectionner la suivante 2 + ... + 10 en utilisant la boucle **for**.

On définit ensuite une fonction **y=somme(n)** qui calcule la sommes 1 + 2 + ... + *n* et on compare avec le resultat de la question 1 pour *n* = 10.

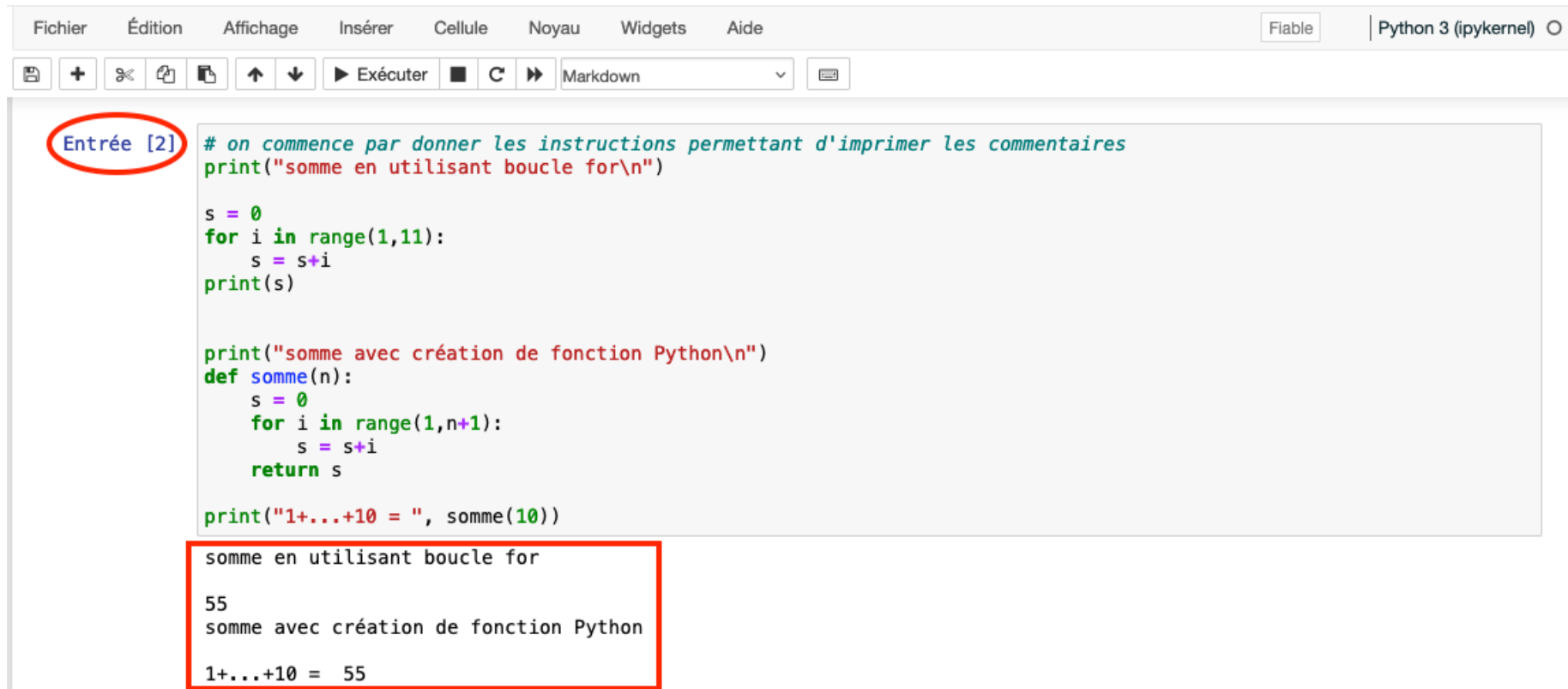
```
Entrée [ ]: # on commence par donner les instructions permettant d'imprimer les commentaires
print("somme en utilisant boucle for\n")

s = 0
for i in range(1,11):
    s = s+i
print(s)

print("somme avec création de fonction Python\n")
def somme(n):
    s = 0
    for i in range(1,n+1):
        s = s+i
    return s

print("1+...+10 = ", somme(10))
```

Les résultats s'afficheront directement sous la cellule concernée.



The screenshot shows a Jupyter Notebook interface. At the top is a menu bar with options: Fichier, Édition, Affichage, Insérer, Cellule, Noyau, Widgets, Aide. On the right, there's a 'Fiable' button and 'Python 3 (ipykernel)' with a refresh icon. Below the menu is a toolbar with icons for saving, adding, undo, redo, and running code. The main area contains a code cell labeled 'Entrée [2]' (circled in red). The code in the cell is as follows:

```
# on commence par donner les instructions permettant d'imprimer les commentaires
print("somme en utilisant boucle for\n")

s = 0
for i in range(1,11):
    s = s+i
print(s)

print("somme avec création de fonction Python\n")
def somme(n):
    s = 0
    for i in range(1,n+1):
        s = s+i
    return s

print("1+...+10 = ", somme(10))
```

Below the code cell, the output is displayed in a separate box (outlined in red):

```
somme en utilisant boucle for

55
somme avec création de fonction Python

1+...+10 = 55
```

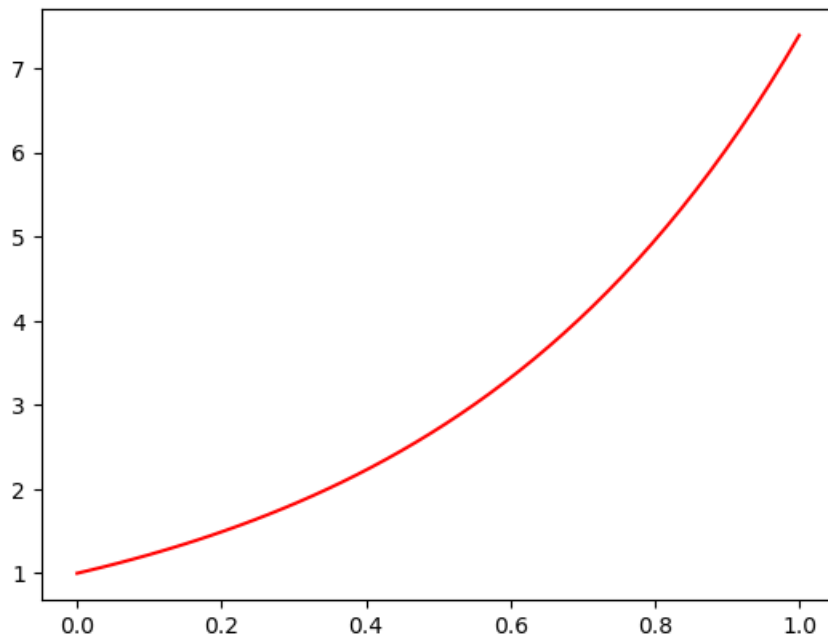
Graphiques avec Matplotlib

Si vous exécutez la première cellule de la section Fonctions et Graphiques le graphique correspondant s'affichera directement dans Jupyter Notebook.

```
Entrée [3]: def f(x):  
            return np.exp(2*x)  
  
            print('f(2) =', f(2))  
            print('f([1,3,4]) =', f(np.array([1, 3, 4])))  
  
            x = np.linspace(0,1,50)  
            plt.plot(x, f(x), 'r')
```

```
f(2) = 54.598150033144236  
f([1,3,4]) = [ 7.3890561  403.42879349 2980.95798704]
```

```
Out[3]: [<matplotlib.lines.Line2D at 0x7fe0c9865790>]
```



Remarque. Lorsque vous générez plusieurs graphiques dans un seul notebook, l'ajout de `plt.show()` après chaque graphique clarifie le rendu en séparant visuellement chaque figure.