| Première<br>Programmation | Un environnement de programmation Python | |
|---|---|---|

**Objectifs de la séance :**

- *Apprendre à installer et utiliser une distribution Python*
- *Utiliser une console Python*
- *Utiliser l'éditeur Spyder*
- *Découvrir l'import de module et de bibliothèques*
- *Écrire des tests unitaires*

**Pré-requis :**

- *savoir écrire un petit script Python*

**Matériel et Logiciel nécessaire :PC avec connexion internet**

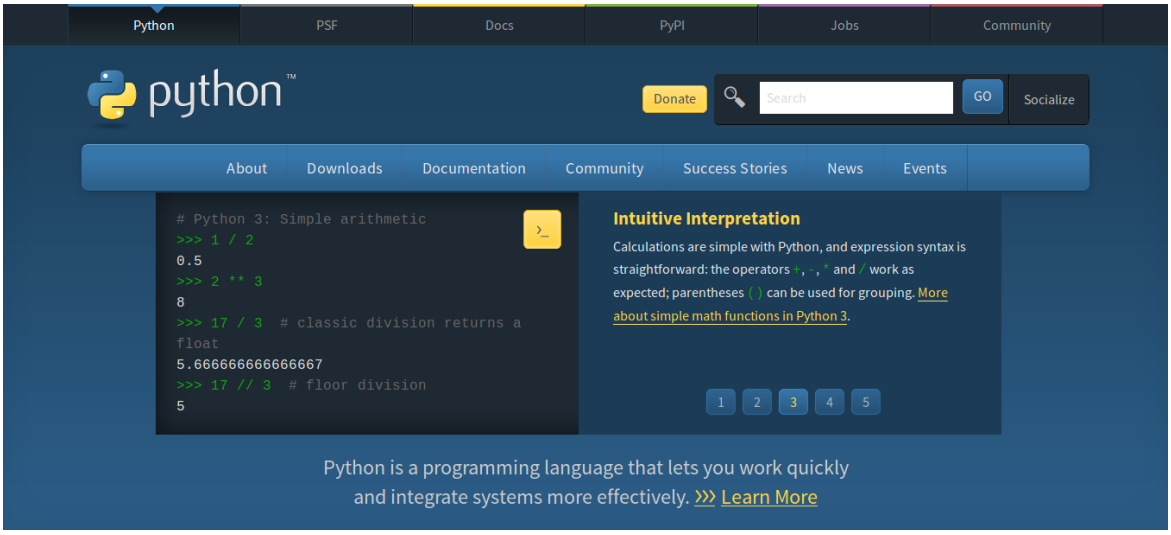
- *L'environnement Anaconda*
- *L'environnement Edupython*

**Durée estimée : 1,30 h**

## A - EduPython

Python[1] est un langage de programmation utilisé depuis déjà un certain temps dans le monde de l'éducation. Nous pouvons l'installer, sur tous les systèmes d'exploitation, directement à partir du site officiel :

https://www.python.org/

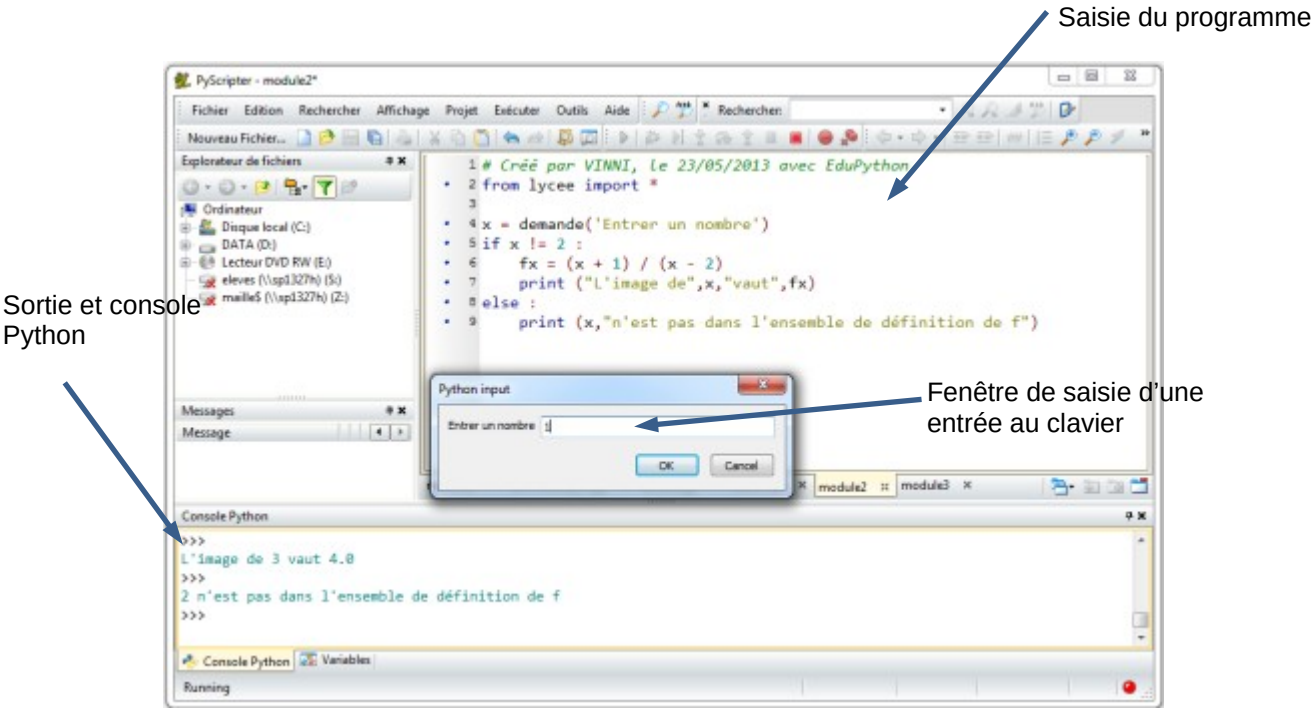



Bien que de nombreuses **bibliothèques** soient déjà présentes dans cet installation basique (os, math,…), il arrive très souvent qu'elles soient pas suffisantes. Heureusement il en existe bien d'autres : pygame, pyQT5, matplotlib, numpy, … et des outils pour les installer. C'est pour faciliter l'intégration de la majorité nécessaire pour l'éducation qu'une communauté d'enseignant a décidé de les regrouper dans **EduPython** :

> *« EduPython est une distribution clé en main et portable pour programmer avec vos élèves sous un environnement Python 3 »*

Cette distribution propose également un **éditeur de texte** (PyScripter) et une **console** Python et de nombreux outils bien pratiques (Jupyter NoteBook, QtDesigner, …). Son avantage est qu'elle est très facile à installer et portable (ça peut être bien pratique pour travailler sur n'importe quelle machine). Un désavantage non des moindres : son éditeur ne fonctionne que sous Windows.

1 Son auteur : Guido van Rossum




Tester ce logiciel avec le code suivant :

```
from math import sqrt

def f(x : float) -> float:
   if x >= 0:
       return sqrt(x)
   else :
       print('Le nombre x doit être positif')

x = float(input('Saisir une valeur positive pour x : '))

print('La racine carrée de ', x, ' est ', f(x))
```

## B - Anaconda et Spyder

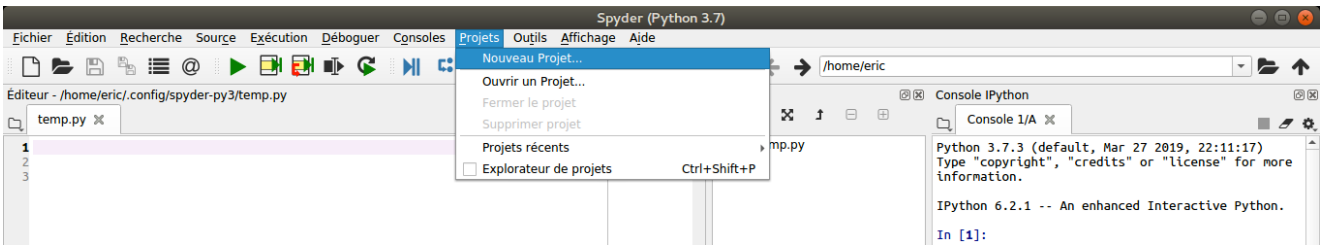
Pour l'enseignement de la spécialité NSI, nous avons décidé d'utiliser la distribution Anaconda[2] :

https://www.anaconda.com/distribution/

Elle s'installe sur tous les systèmes d'exploitation et intègre un éditeur doté d'outils bien utiles pour votre formation : la gestion des projets, tests unitaires et suivi des variables.

### B.1- La gestion des projets

1. Ouvrir Spyder et créer un nouveau projet : nous l'appellerons **Collatz**.




   Vous avez maintenant une nouvelle boîte de dialogue dans le bandeau de gauche contenant le dossier Collatz ainsi créé.

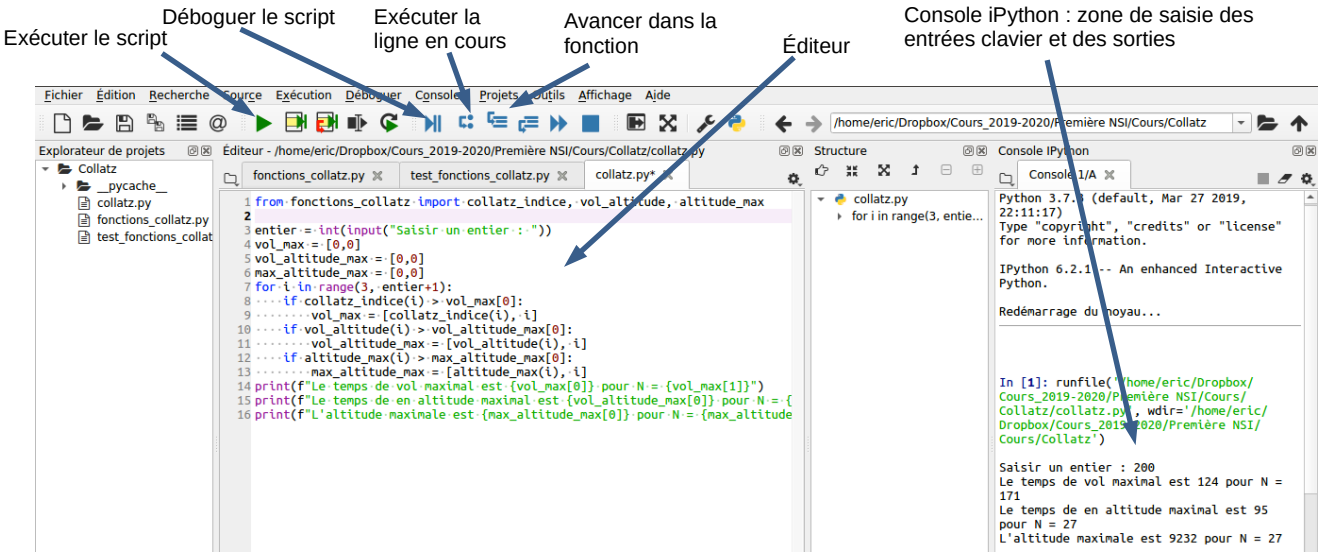
2. Créer un nouveau fichier nommé `fonctions_collatz.py` dans lequel vous recopiez toutes les fonctions créés dans le TP sur les tableaux indexés (le corrigé se trouve dans le dossier « partages » mais également dans Chamilo).

2 Le logiciel EduPython est d'ailleurs basé sur celle-ci.

Vous ferez apparaître le bandeau « Structure » qui donne les différentes fonctions définies dans votre fichier, vous devez avoir : `collatz_indice`, `collatz_liste`, `vol_altitude` et `altitude_max`.

3. Créer un nouveau fichier nommé collatz.py qui va contenir le programme écrit dans le TP. Pour intégrer les fonctions contenues dans le fichier fonctions_collatz.py vous devez l'importer au début de votre script. Par exemple en saisissant :

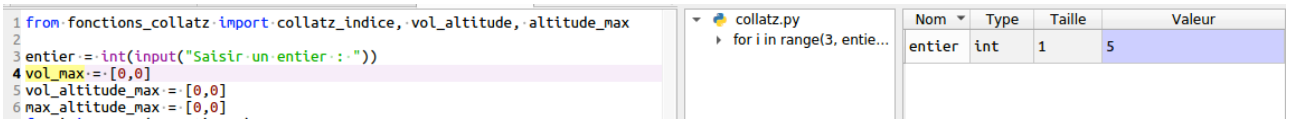
```
from fonctions_collatz import collatz_indice, vol_altitude, altitude_max
```




## B.2- L'explorateur de variables

L'outil « Déboguer le script » permet de suivre l'évolution des variables au cours de l'exécution du script, cela peut être bien pratique lors de la phase de débogage du programme.

1. Vous aller exécuter les les premières lignes, saisir par exemple « 5 », puis passer rendre visible l'onglet « Explorateur de variables » :




2. Vous avez la possibilité d' « avancer dans une fonction » : essayez de le faire lors de l'appel de la fonction `vol_altitude()`.

## B.3- Les tests unitaires

Lors du TP, après avoir créé les différentes fonctions vous les avez testé avec certaines valeurs pour lesquelles vous connaissiez le retour attendu. Il existe une bibliothèque dédiée à ce type de test : `unittest`

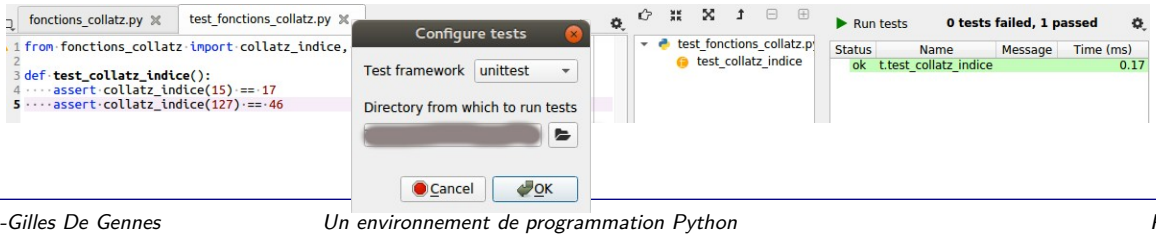
Ceci permet de décider une série de tests à faire passer à la fonction afin de valider le fonctionnement de celle-ci.

1. Créer un fichier nommé `test_fonctions_collatz.py` (ce fichier doit toujours commencer par `test_` suivi du nom du fichier contenant les fonctions).
2. Voici les quelques tests donnés dans le TP pour la fonction collatz_indice() :

```
from fonctions_collatz import collatz_indice

def test_collatz_indice():
    assert collatz_indice(15) == 17
    assert collatz_indice(127) == 46
```

3. Dans le menu « Exécution » choisir « Run unit tests », puis indiquer « unittest »



4. Ajouter les tests donnés dans le TP pour les autres fonctions.
5. Pour rendre votre projet encore plus modulaire et tester le programme principal, vous allez modifier le fichier `collatz.py` ainsi :

```
from fonctions_collatz import collatz_indice, vol_altitude, altitude_max

# On définit une fonction
def valeurs_max(N : int)->tuple:
    """La fonction retourne un tuple contenant
       les trois listes vol_max, vol_altitude_max, max_altitude_max
    """
    vol_max = [0,0]
...

    return ..., ..., ...

# Ne s'exécute pas lorsque le fichier est importé comme module
# dans le fichier des tests unitaires
if __name__ == '__main__':
    entier = int(input("Saisir un entier : "))
    (vol_max, vol_altitude_max, max_altitude_max) = valeurs_max(entier)

    print(f"Le temps ...
```

6. Ajouter les lignes nécessaires au fichier `test_fonctions_collatz.py` afin d'ajouter un test unitaire pour cette nouvelle fonction.

## B.4- Tracer les courbes de vol des suites de Collatz

La bibliothèque `matplotlib` contient un module `pyplot` pour, entre autres, tracer des courbes. Nous renommerons ce module `plt`. Compléter le code ci-dessous.

```
# La bibliothèque pour tracer des courbes
import matplotlib.pyplot as plt
# On importe la fonction nécessaire
from ...

N = int(input("Saisir une valeur pour N : "))
Liste = collatz_liste(N)

# On crée la liste des indices par compréhension
indices = [...]

plt.plot(indices, Liste)

plt.show() # affiche la figure a l'écran
```

Voici ce que l'on obtient pour N = 127.