

Thème 0 Représentation des données (Types de base)	Thème 1 Représentation des données (Types construits)	Thème 2 Langages et programmation	Thème 3 Algorithmique	Thème 4 Traitement des données en tables	Thème 5 Architectures matérielles, systèmes d'exploitation et réseaux	Thème 6 Interactions entre l'homme et la machine sur le Web
Bases	Tableaux	Variables	Complexité	Manipulation csv	Von Neumann	Interactions page Web
Codage relatifs	Tuples	Boucles for & while	Tri par insertion	Trier des données	Réseaux	Protocole HTTP
Codage réels	Dictionnaires	Conditionnelles	Tri par selection		Protocoles de com.	Formulaires
Opérateurs booléens		Fonctions	Algo KNN		Systèmes d'exploitati	
Texte en machine		Bibliothèques	Dichotomie		Commandes Linux	
		Tests	Algo gloutons		IHM	

# Chapitre 1 - Programmer en Python

## Séance 1 - Prise en main de Python



# Programmer un ordinateur, c'est quoi ?

**Programmer, c'est créer des programmes** (suite d'instructions données à l'ordinateur) !

Un ordinateur sans programme ne sait rien faire.

Il existe différents langages qui permettent de programmer un ordinateur, mais le seul directement utilisable par le processeur est le **langage machine** (suite de 1 et de 0). Aujourd'hui (presque) plus personne ne programme en langage machine (trop compliqué).

Les informaticiens utilisent des instructions (mots souvent en anglais) en lieu et place de la suite de 0 et de 1. Ces instructions, une fois écrites par le programmeur, sont "traduites" en langage machine. Un programme spécialisé assure cette traduction. Ce système de traduction s'appellera interpréteur ou bien compilateur, suivant la méthode utilisée pour effectuer la traduction.

Il existe 2 grandes familles de langages de programmation :

- **Les langages de bas niveau** sont très complexes à utiliser, car très éloignés du langage naturel, on dit que ce sont des langages « proches de la machine », en contrepartie ils permettent de faire des programmes très rapides à l'exécution. L'**assembleur** est le langage de bas niveau. Certains "morceaux" de programmes sont écrits en assembleur encore aujourd'hui.
- **Les langages de haut niveau** sont eux plus "faciles" à utiliser, car plus proches du langage naturel (exemple : si  $a=3$  alors  $b=c$ ). Exemples de langages de haut niveau : C, C++ , Java, Python...

En NSI, notre langage de prédilection sera **Python**.

## C'est quoi Python ?

Le langage de programmation Python a été créé en 1989 par **Guido van Rossum**, aux Pays-Bas. Le nom Python vient d'un hommage à la série télévisée Monty Python's Flying Circus dont G. van Rossum est fan. La première version publique de ce langage a été publiée en 1991.

La dernière version de Python est la **version 3**.

La Python Software Foundation est l'association qui organise le développement de Python et anime la communauté de développeurs et d'utilisateurs.



<https://docs.python.org/fr/3/tutorial/>

## Caractéristiques de Python

- Il est **multiplateforme**. C'est-à-dire qu'il fonctionne sur de nombreux systèmes d'exploitation : Windows, Mac OS X, Linux, Android, iOS, depuis les mini-ordinateurs Raspberry Pi jusqu'aux supercalculateurs.
- Il est **gratuit**. Vous pouvez l'installer sur autant d'ordinateurs que vous voulez (même sur votre téléphone !).
- C'est un **langage de haut niveau**. Il demande relativement peu de connaissance sur le fonctionnement d'un ordinateur pour être utilisé.
- C'est un langage **interprété**. Un script Python n'a pas besoin d'être compilé pour être exécuté, contrairement à des langages comme le C ou le C++.
- Il est **orienté objet**. C'est-à-dire qu'il est possible de concevoir en Python des entités qui miment celles du monde réel (une cellule, une protéine, un atome, etc.) avec un certain nombre de règles de fonctionnement et d'interactions.

- Il est **relativement simple à prendre en main**.
- Enfin, il est **très utilisé** en bioinformatique et plus généralement en analyse de données.

Toutes ces caractéristiques font que Python est désormais enseigné dans de nombreuses formations, depuis l'enseignement secondaire jusqu'à l'enseignement supérieur.

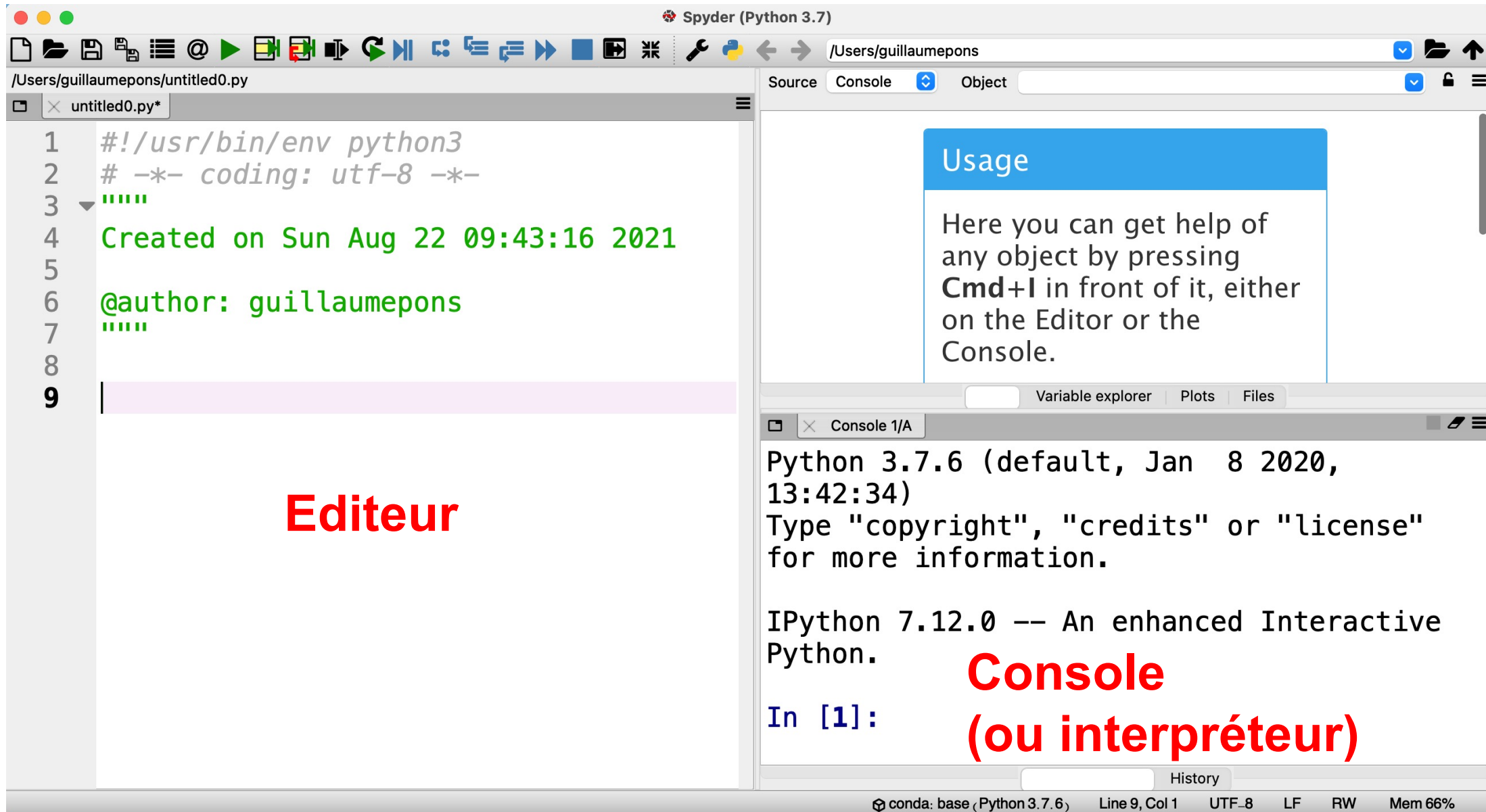
### Quel éditeur ou IDE pour Python ?

Un **IDE** (Integrated Development Environment) est un regroupement d'outils utiles pour le développement d'applications (éditeur de code, débbugger, builder, indexation du code pour recherches « intelligentes » dans les projets...), rassemblés dans un logiciel unique. Les IDE sont donc plus que des éditeurs de code.



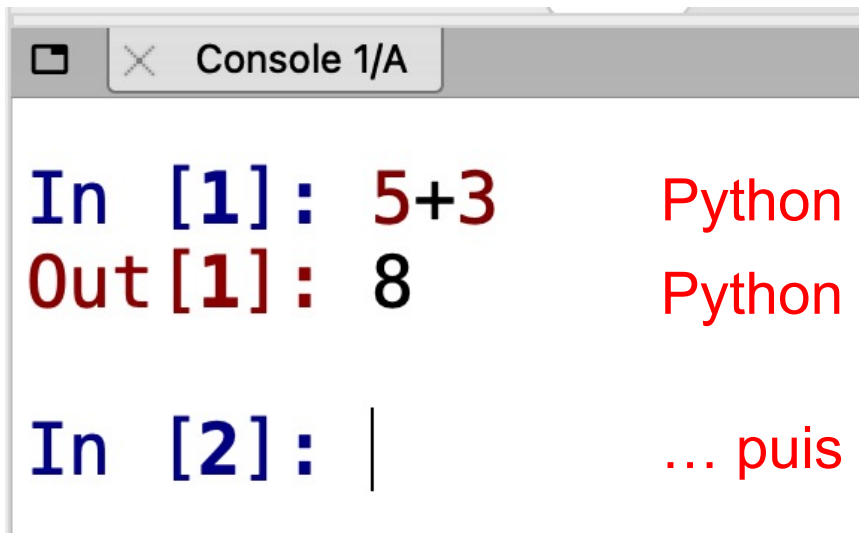
Pour écrire nos programmes en Python, nous utiliserons en classe le logiciel **Spyder**.

Une fois **Spyder** lancé, vous devriez obtenir quelque chose qui ressemble à cela :



Spyder se divise en plusieurs fenêtres, deux fenêtres vont principalement nous intéresser : la fenêtre "**éditeur**" et la fenêtre "**console**".

Dans la **console**, l'interpréteur attend vos instructions et les exécute quand vous tapez sur la touche entrée.

A screenshot of the Spyder console window. The window has a title bar with a close button and the text "Console 1/A". Inside the window, the text "In [1]: 5+3" is displayed in blue, followed by "Out [1]: 8" in red. Below this, "In [2]:" is displayed in blue with a vertical cursor line to its right. To the right of the console window, there are two lines of red text explaining the process: "Python affiche l'invite. L'utilisateur tape une expression." and "Python évalue et affiche le résultat." Below that, another line of red text says "... puis réaffiche l'invite."

Python affiche l'invite. L'utilisateur tape une expression.

Python évalue et affiche le résultat.

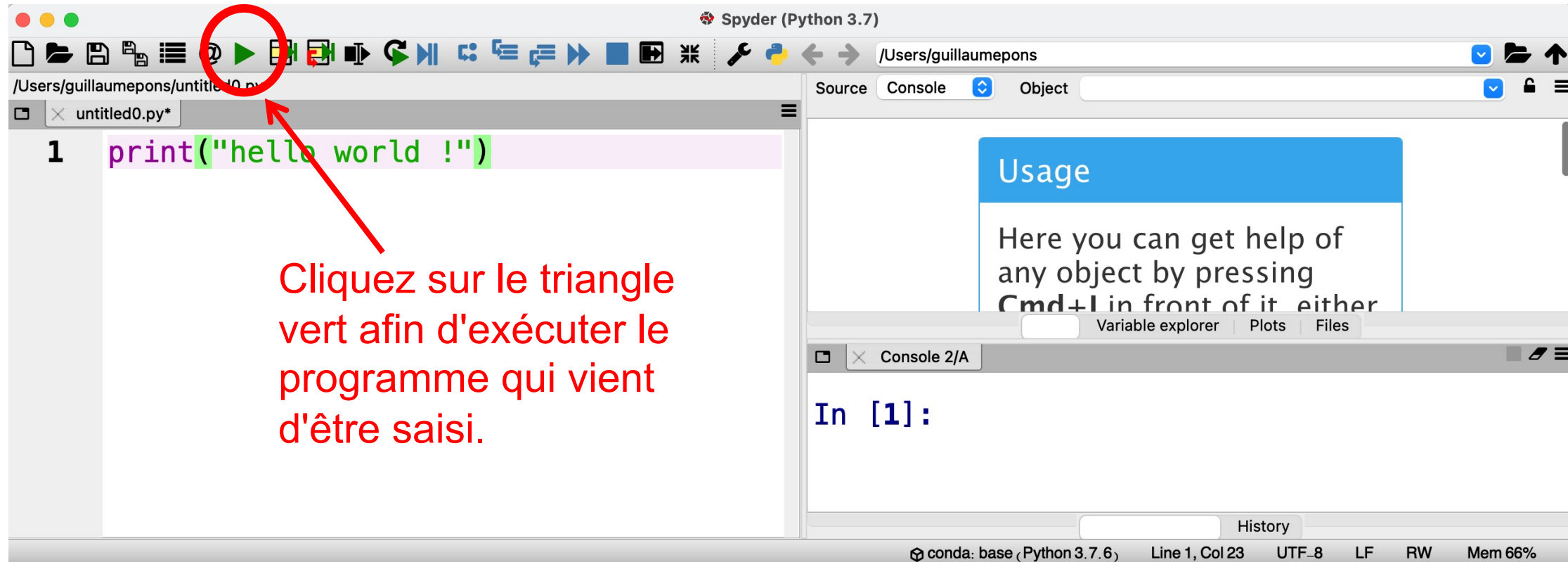
... puis réaffiche l'invite.



En mode programmation, on peut écrire des programmes, les sauvegarder et les faire fonctionner.

Dans la fenêtre **éditeur**, saisissez le programme suivant :

```
print("hello world !")
```



Spyder va vous demander d'**enregistrer** le programme, enregistrez-le dans un dossier qui vous servira de dossier de travail.

**Vous devez voir le message hello world ! apparaître dans la console :**

```
1 print("hello world !")
```

Usage

Here you can get help of any object by pressing **Cmd+I** in front of it. either

Variable explorer | Plots | Files

Console 2/A

```
In [1]: runfile('/Users/guillaumepons/untitled0.py', wdir='/Users/guillaumepons')
hello world !

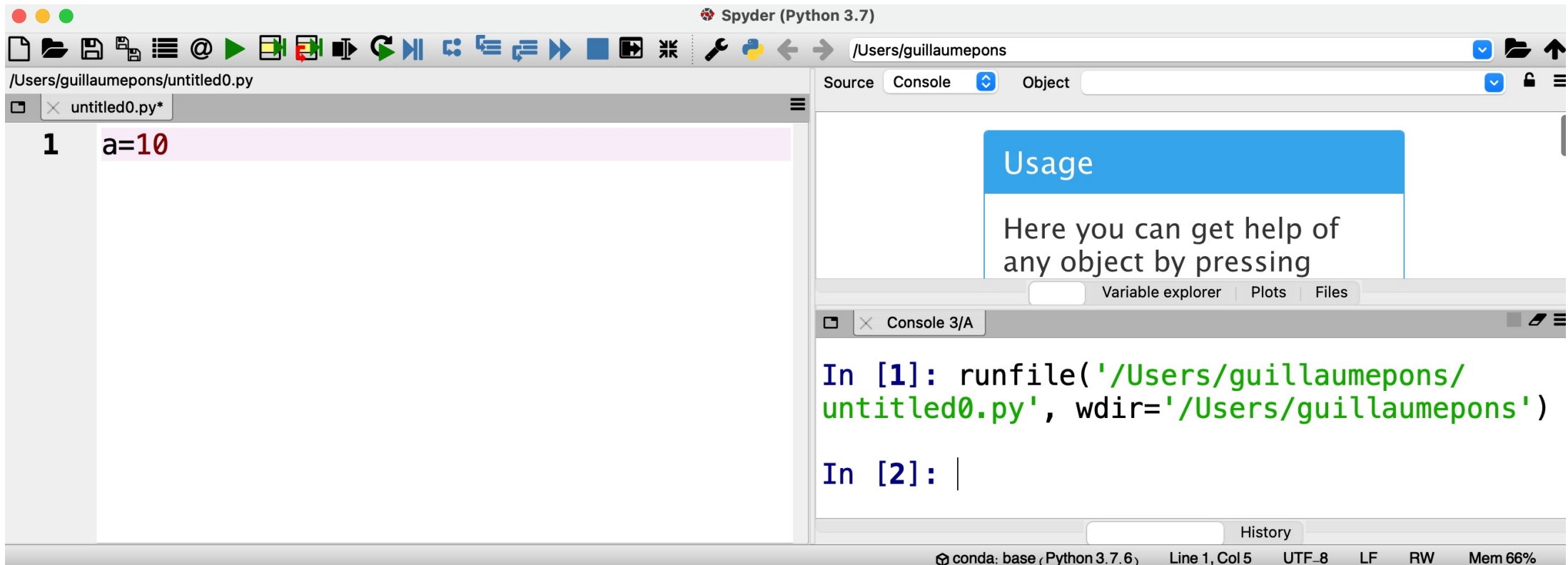
In [2]:
```

History

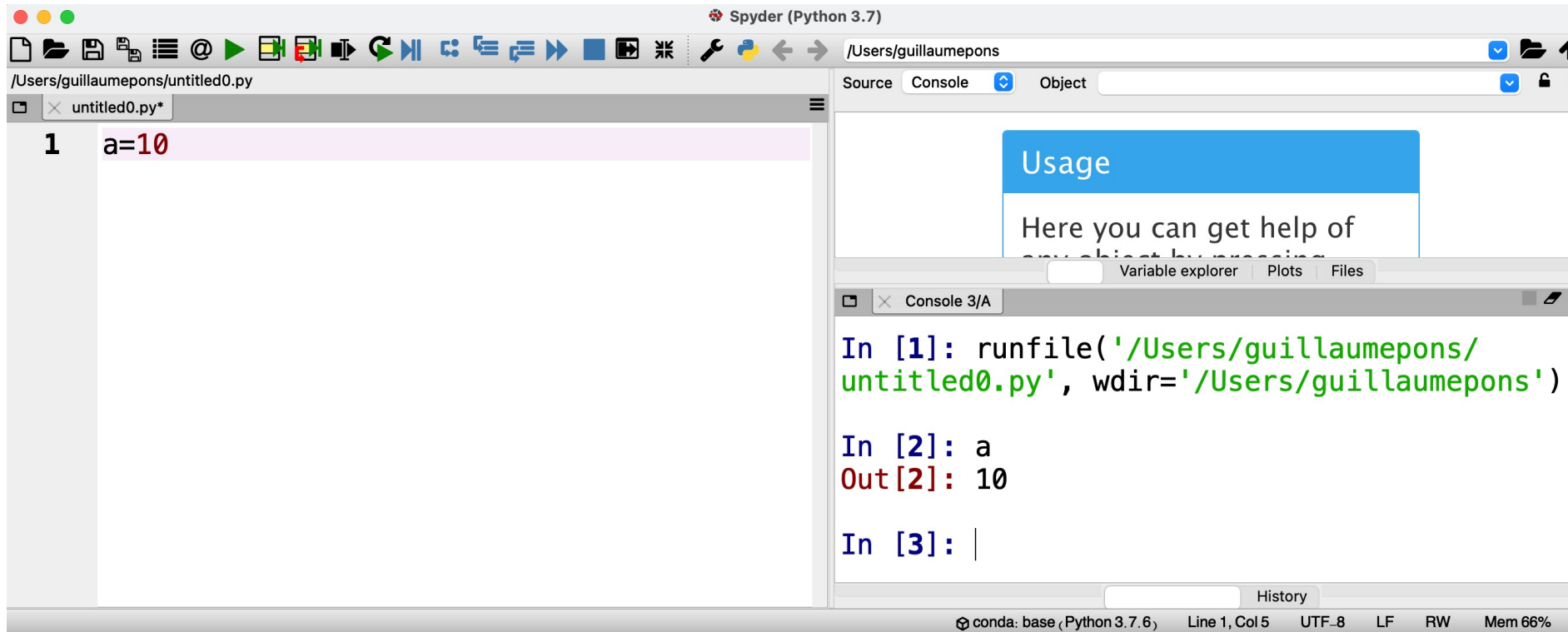
conda: base (Python 3.7.6) | Line 1, Col 23 | UTF-8 | LF | RW | Mem 66%

Avec la commande **New file** du menu **File**, ouvrez un nouveau fichier et saisissez la ligne suivante : **a = 10**

Après avoir exécuté le programme en cliquant sur le triangle vert, il est possible de connaître la valeur de la variable **a** en tapant le nom de la variable dans la console de Spyder.



Tapez **a** dans la partie console. Après avoir appuyé sur la touche Entrée, vous devriez voir la valeur associée au nom **a** s'afficher dans la console.

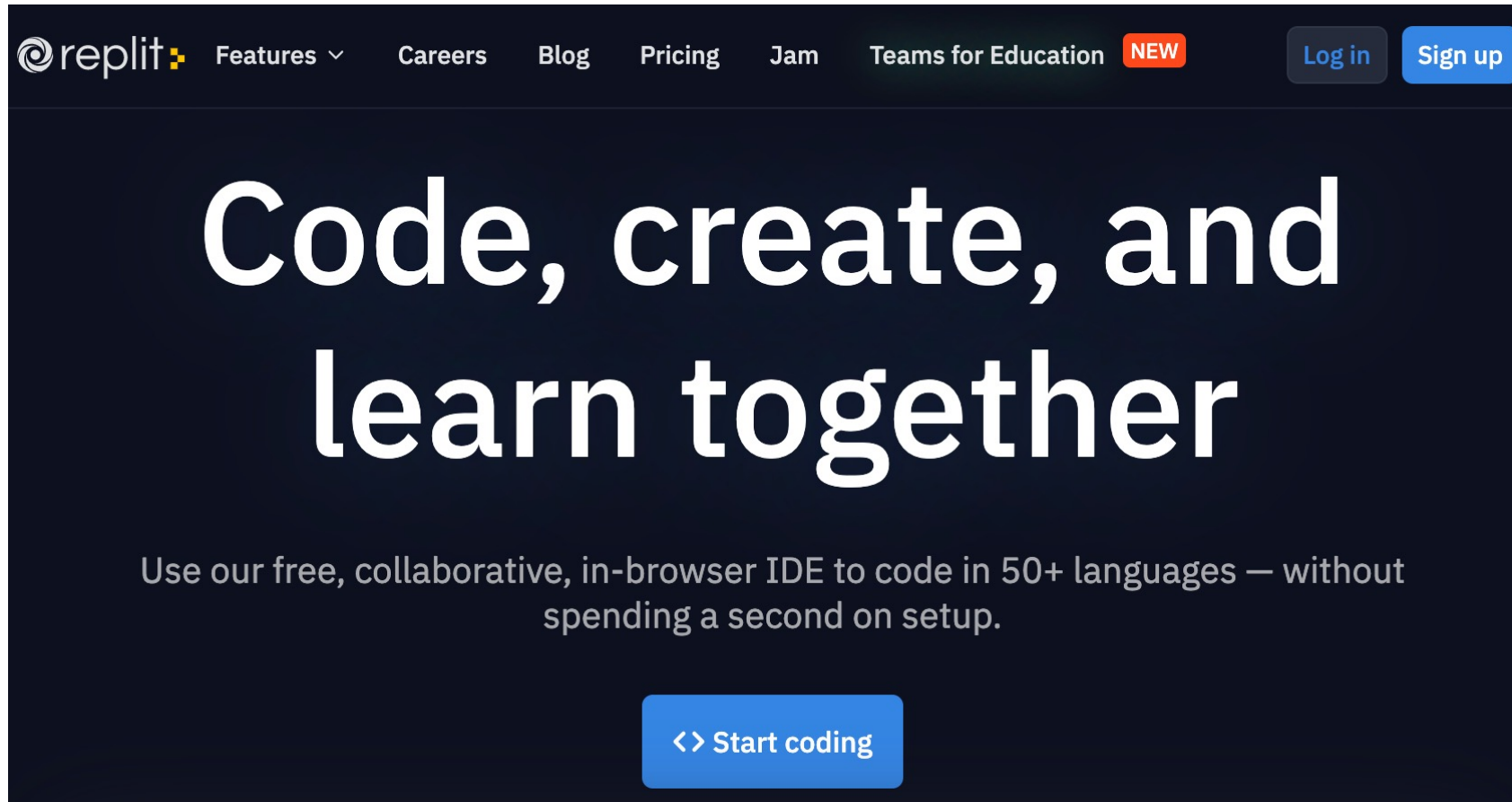


Dans la suite la procédure sera toujours la même :

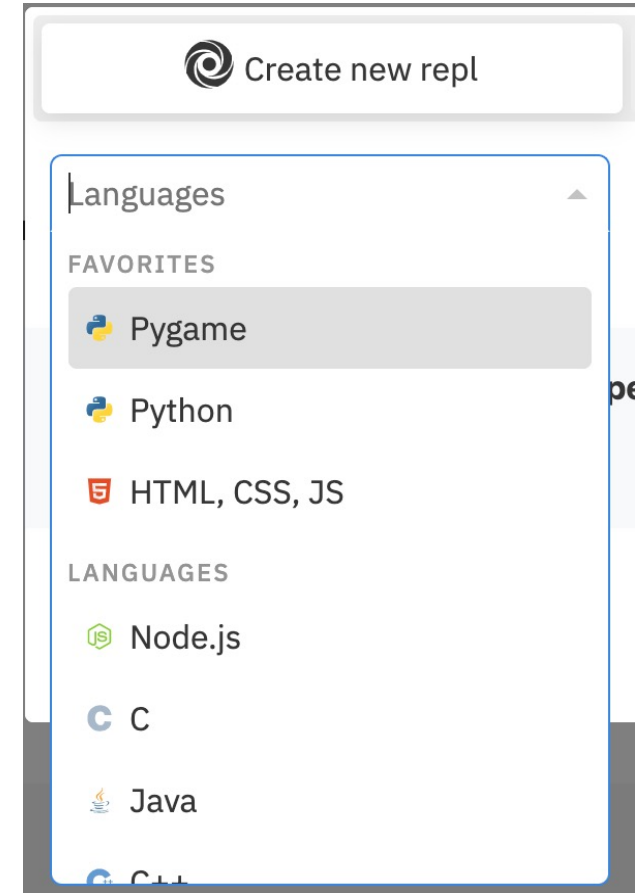
- vous utiliserez la partie "**éditeur**" pour saisir votre programme
- vous utiliserez la partie "**console**" pour afficher la valeur d'une variable



Si vous n'avez pas la possibilité d'utiliser Spyder (ou tout autre éditeur), vous pouvez utiliser un **éditeur en ligne**, comme <https://replit.com>



The image shows the Replit homepage. At the top, there is a navigation bar with the Replit logo, links for Features, Careers, Blog, Pricing, Jam, and Teams for Education (marked as NEW), and buttons for Log in and Sign up. The main section has a dark blue background with the text "Code, create, and learn together" in large white font. Below this, it says "Use our free, collaborative, in-browser IDE to code in 50+ languages — without spending a second on setup." and a blue button with a code icon and the text "Start coding".



The image shows a dropdown menu for selecting a language to create a new Repl. At the top, there is a button that says "Create new repl" with the Replit logo. Below it, the menu is titled "Languages". Under the "FAVORITES" section, there are three options: Pygame (with a Python icon), Python (with a Python icon), and HTML, CSS, JS (with a code icon). Under the "LANGUAGES" section, there are three options: Node.js (with a Node.js icon), C (with a C icon), and Java (with a Java icon). The menu is currently open, showing these options.

The image shows a code editor with a file explorer on the left, a code editor in the center, and a game window and console on the right.

**File Explorer:**

- Files
- main.py
- images
- B\_sprites.py
- Packager files
- poetry.lock
- pyproject.toml

**Code Editor (main.py):**

```
1  """
2  Programme de base de Mario.
3  Il n'utilise pas les classes de python, ce
4  qui compliquera le programme avec
5  l'ajout de personnages.
6  """
7  import pygame
8  from pygame.locals import *
9  from B_sprites import *
10 pygame.init()
11
12 index = 0
13 xp = 50
14 yp = 296
15 deplt = 10
16 sens = "droite"
17
18 continuer = 1
19
20 """
21 Boucle principale (événement) du jeu. Elle
22 s'exécute tant que Quit
23 n'est pas cliqué (dans ce cas la variable
```

**Game Window:**

The game window shows a Mario-like character on a platform. There are question mark blocks above the platform. The window title is "pygame window".

**Console:**

```
/usr/bin/run-project
pygame 2.0.1 (SDL 2.0.14, Python 3.8.10)
Hello from the pygame community. https://www.pygame.org/contribute.html
```



## Jupyter et ses notebooks

Les notebooks Jupyter sont des cahiers électroniques qui, dans le même document, peuvent rassembler du texte, des images, des formules mathématiques et du code informatique exécutable. Ils sont manipulables interactivement dans un navigateur web.

Initialement développés pour les langages de programmation Julia, Python et R (d'où le nom Jupyter), les notebooks Jupyter supportent près de 40 langages différents.

La cellule est l'élément de base d'un notebook Jupyter. Elle peut contenir du texte formaté au format Markdown ou du code informatique qui pourra être exécuté.

Lancez Jupyter puis ouvrez le fichier **exemple.ipynb**





```
1 Un exemple de cellule **Markdown**
2 # Grand titre
3 #### Jupyter & ses notebooks
```

Entrée [1]:

```
1 a=10
2 b="bonjour"
```

Entrée [2]:

```
1 print(a)
```

10

Entrée [3]:

```
1 print(b)
```

bonjour

Entrée [ ]:

```
1 |
```



Vous pouvez ouvrir également le fichier **exemple.ipynb** dans l'environnement de travail **JupyterLab** disponible sur <https://lyceeconnecte.fr>





**Lycée Grand Air**

identifiant

mot de passe



<https://replit.com>

identifiant

mot de passe



**Pronote**

identifiant

mot de passe



<http://www.france-ioi.org>

identifiant

mot de passe

groupe



**Lycée connecté**

identifiant

mot de passe



**Dépôt des cours sur GitHub**

<https://github.com/gpons971>



## Bibliographie

- David Roche <https://pixees.fr/informatiquelycee/>
- Université de Paris [https://python.sdv.univ-paris-diderot.fr/01\\_introduction/](https://python.sdv.univ-paris-diderot.fr/01_introduction/)