CHAPITRE 2

Introduction à Python

Thème : Langage et programmation

Contenus	**Capacités attendues**
Mettre en évidence un corpus de constructions élémentaires	Séquences, affectation [...]

I. Un peu d'histoire

Python est un langage de programmation créé en 1989 par Guido Van Rossum [1]. Il est particulièrement répandu dans le monde scientifique, et possède de nombreuses bibliothèques optimisées destinées au calcul numérique. Il s'agit d'un langage de programmation impérative, c'est à dire que les lignes de code sont exécutées par l'ordinateur les unes à la suite des autres, dans l'ordre où elles ont été écrites. Il s'agit d'un langage open source (libre de droit et gratuit, développé par une large communauté d'utilisateurs)
La version qui sera utilisée dans cette spécialité est la version 3.7.
La documentation est disponible à cette adresse : `https://docs.python.org/3/`

1620	**Edmund Gunter** crée la première règle à calcul
1642	**Blaise Pascal** invente la première machine à calculer
1834	**Charles Babbage** énonce le principe d'un ordinateur : un programme fournit à une machine des données et des instructions
1840	**Ada Lovelace** réalise le premier programme informatique
1954	Création du langage FORTRAN
1960	Création du langage COBOL
1970	Création du langage PASCAL
1972	Création du langage C
1980	Création du langage C++
1985	Création du langage Java
1989	Création du langage Python
1994	Création du langage PHP
1995	Création du langage Javascript

1. `https://fr.wikipedia.org/wiki/Guido_van_Rossum`

II. Les outils utilisés

II. 1. L'IDE Spyder

Dans un IDE (Integrated Development Environment), les outils sont prévus pour être utilisés ensemble, et sont intégrés dès le départ dans une même fenêtre. Il existe de multiples IDE pour utiliser Python : Pyscripter, IDLE, wxPython, PyDev...

L'IDE qui sera utilisé dans cette spécialité est **Spyder** (fichier d'installation disponible à cette adresse : `http://winpython.github.io/`).

Ses principales fonctionnalités sont :

- ▷ la coloration syntaxique (les couleurs sont différentes pour les instructions, les variables, les tests...) ;
- ▷ l'autocomplétion (les premières lettres d'une variable ou d'une instruction sont reconnues et automatiquement complétées) ;
- ▷ la tabulation automatique (indispensable pour séparer les parties de code à exécuter) ;
- ▷ l'explorateur de fichiers et de variables.

Lors de l'installation de l'IDE Spyder, l'environnement Python sera aussi installé. Pour les utilisateurs de MacOS ou de Linux, consulter la FAQ :`https://docs.spyder-ide.org/installation.html`.

Attention

Tout autre IDE que Spyder pourra être utilisé à la maison. Toutefois, les projets rendus devront être compatibles avec l'environnement Spyder installé au Lycée (version de Python installée, modules installés, etc.).

La fenêtre principale se présente comme ci-dessous :

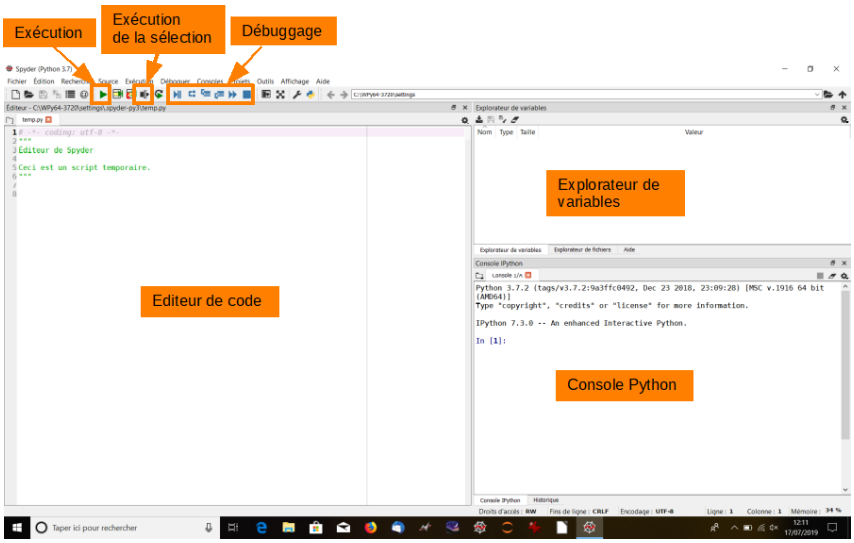

FIGURE 2.1 – Fenêtre principale de Spyder

II. 2. Les notebooks

Les notebooks sont accessibles via l'ENT, en utilisant le menu **JupyterLab**.

Un notebook est constitué d'une suite de cellules, soit textuelles, soit contenant du code. Les cellules de code sont facilement reconnaissables, elles sont précédées de **[ ] :**

La façon habituelle d'évaluer l'ensemble du notebook consiste à sélectionner la première cellule, et à taper **Shift+Enter** jusqu'à atteindre la fin du notebook.

Lorsqu'une cellule de code a été évaluée, Jupyter ajoute sous la cellule **[ ] :** une cellule qui donne le résultat.

Jupyter ajoute également un nombre entre les crochets, par exemple **[1] :**. Ce nombre vous permet de retrouver l'ordre dans lequel les cellules ont été évaluées.

Vous pouvez naturellement modifier ces cellules de code pour faire des essais, ou ajouter des cellules avec le bouton + de la barre de boutons.

Attention à bien évaluer les cellules dans l'ordre.
Il est important que les cellules de code soient évaluées dans le bon ordre. Si vous ne respectez pas l'ordre dans lequel les cellules de code sont présentées, le résultat peut être inattendu.

Réinitialiser l'interpréteur
Si vous faites trop de modifications, ou perdez le fil de ce que vous avez évalué, ou que le programme « patine », il peut être utile de redémarrer votre interpréteur. Le menu **Kernel → Restart** vous permet de faire cela.

Revenir à la version du cours
Vous pouvez toujours revenir à la version "du cours" grâce au menu **File → Reload notebook from Disk**. Attention, avec cette fonction vous restaurez tout le notebook et donc vous perdez vos modifications sur ce notebook.

Télécharger le notebook
Vous pouvez télécharger un notebook sur votre ordinateur grâce au menu **File → Save as**. L'extension d'un notebook est « .ipynb ».

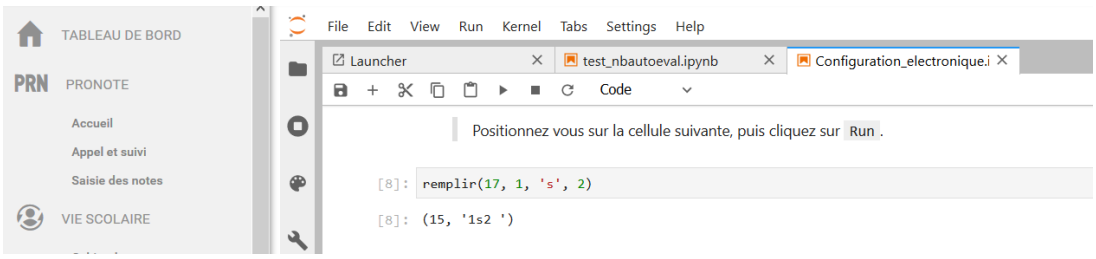

FIGURE 2.2 – L'environnement Jupyter

II. 3. Python Tutor

Python Tutor a été créé par Philip Guo, professeur assistant à l'université de Californie. Il a créé Python Tutor pour aider les gens à comprendre ce qu'il se passe chaque fois qu'une ligne de code est exécutée, étape qui, d'après lui, constitue une barrière majeure à l'apprentissage de la programmation.

Disponible en licence libre, cet outil est accessible depuis `http://pythontutor.com/visualize.html#`.

Il permet l'exécution du code en mode « pas à pas », en observant les modifications de chacune des variables.

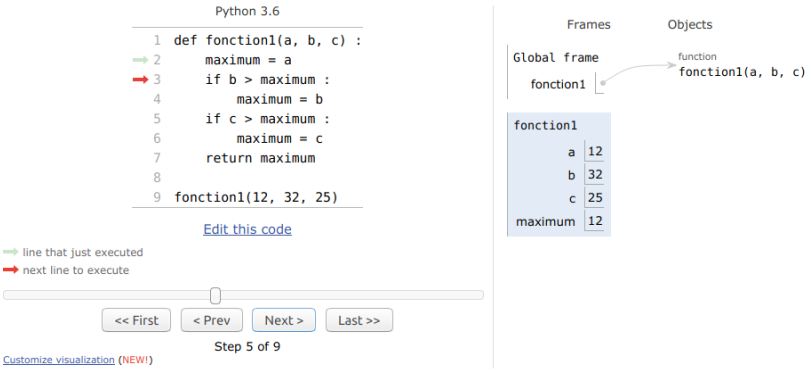

FIGURE 2.3 – Capture d'écran du site http ://pythontutor.com

III. Les variables

III. 1. Nommage des variables

Les noms de variables peuvent contenir les caractères minuscules, majuscules, les chiffres et quelques caractères spéciaux (_ par exemple). Une variable ne peut jamais commencer par un chiffre. Pour une plus grande uniformité, il existe des conventions de nommage, que vous allons détailler ici [2] :

- ▷ une variable ne commence jamais par un chiffre ;
- ▷ une variable ne commence jamais par un tiret bas (_) ;
- ▷ on utilise des noms de variable explicites, même s'ils sont longs ;
- ▷ on n'utilise pas de caractères accentués ;
- ▷ si une variable possède plusieurs mots, ils sont séparés par un tiret bas (ma_variable).

Attention

Il existe des mots réservés en Python `and`, `or`, `break`, `return`, `lambda`, ... [a]. Faîtes attention à ne pas les utiliser.

a. `https://www.pierre-giraud.com/liste-mots-clefs-reserves-python/`

III. 2. L'affectation

Une affectation se fait en utilisant le signe =. Tester si une égalité est vraie ou fausse se fait en utilisant ==.

2. Convention Pep8 : `https://pep8.org`

Exemple 1

```
>>> a = 13
>>> print(a)
13
>>> a == 13
True
```

Python permet les affectations multiples :

Exemple 2

```
>>> x, y = 1, 2
>>> x
1
>>> y
2
```

III. 3. Modification du contenu d'une variable

Pour multiplier par 2 le contenu de la variable a, on peut écrire `a = 2 * a`. Le contenu de la variable a est alors remplacé.

Exemple 3

```
>>> a = 13
>>> a = 2 * a
>>> a
26
```

IV. Les types de base

Python est un langage à typage dynamique. Cela signifie que lorsqu'une variable est affectée, l'interpréteur trouvera automatiquement son type sans que l'utilisateur soit contraint de le préciser. Pour connaître le type d'une donnée ou le type de la valeur d'une variable, il suffit d'utiliser la fonction `type()`.

Exemple 4

```
>>> type(15)
int
>>> a = "toto"
>>> type(a)
str
```

IV. 1. Le type int (entier)

Ce type est utilisé pour stocker un entier, en anglais integer. Pour cette raison, on appelle ce type `int`. Les entiers peuvent être positifs ou négatifs.

Exemple 5

```
>>> type(128)
int
>>> a = - 256
>>> type(a)
int
```

IV. 2. Le type float (flottant)

Ce type est utilisé pour stocker des nombres à virgule flottante, désignés en anglais par l'expression floating point numbers. Pour cette raison, on appelle ce type : `float`. En français, on parle de flottant. Le séparateur décimal est le point (notation anglo-saxonne).

On peut déclarer une variable en utilisant la notation scientifique. Par exemple, pour saisir $1,2 \times 10^{-7}$, on écrira : `1.2e-7`.

Exemple 6

```
>>> a = 14.5
>>> type(a)
float

>>> a = 11.
>>> type(a)
float

>>> a = 3.25e7
>>> type(a)
float
```

IV. 3. Le type str (chaîne de caractères)

Une donnée de type `str` est une suite quelconque de caractères délimitée soit par des apostrophes (simple quotes), soit par des guillemets (double quotes). `str` est l'abréviation de string, qui veut dire chaîne en français.

Exemple 7

```
>>> a = 'Bonjour'
>>> type(a)
str
>>> b = "Bonsoir"
>>> type(b)
str
```

IV. 4. Le changement de type

Dans Python, il est possible de passer une variable d'un type à l'autre, **à condition que le contenu de la variable soit compatible avec le nouveau type**.

Exemple 8

```
>>> a=13
>>> type(a)
int
>>> a = str(a)      # on change le type de la variable a. Elle devient « str »
>>> a
'13'                      # notez les simples quotes
>>> type(a)
str
```

L'opération inverse est possible aussi. On peut passer d'un caractère à un entier :

Exemple 9

```
>>> a='13'
>>> type(a)
str
>>> a = int(a) # on change le type de la variable a. Elle devient « int »
>>> a
13
>>> type(a)
int
```

Par contre, on ne peut pas changer le caractère 'a' en un entier :

Exemple 10

```
>>> a = "truc"
>>> a = int(a)
Traceback (most recent call last):

File "<ipython-input-6-8941b20ea7e0>", line 1, in <module>
a = int(a)

ValueError: invalid literal for int() with base 10: 'truc'
```

IV. 5. Le type bool (booléen)

Un booléen est une variable qui prend deux états : soit l'état vrai (`True`), soit l'état faux (`False`).

Exemple 11

```
>>> type(True)
bool
```

Les booléens permettent de tester si une expression est vraie ou fausse. On ne peut pas faire

d'opération élémentaire avec des booléens.

Exemple 12

```
>>> 3 > 2
True
>>> 3 < 2
False
```

Activité 1.
On a entré les instructions suivantes dans une console Python :

```
>>> a = 23
>>> b = 12.5
>>> c = 'True'
>>> d = '- 4'
>>> e = 23.5e-5
>>> f = '-12e6'
>>> g = True
```

1. On souhaite connaître le type de la variable a. Quelle instruction faut-il saisir ?

2. Déterminez le type de chacune de ces variables.

3. On souhaite que la variable a soit du type `str`. Quel est le code à saisir ?

4. Quelles sont les variables dont on peut modifier le type en `int` ?

V. Les opérations élémentaires

Les opérations élémentaires sont résumées dans le tableau suivant :

Opération	**Symbole**	**Exemple**
addition	+	5 + 2 renvoie 7
soustraction	-	5 - 2 renvoie 3
multiplication	*	5 * 2 renvoie 10
exponentiation (puissance)	**	5 ** 2 renvoie 25
division	/	5 / 2 renvoie 2.5
quotient division entière	//	5 // 2 renvoie 2
reste division entière	%	5 % 2 renvoie 1

Les opérations ne sont possibles que si les types sont compatibles. Par exemple, on peut additionner un entier et un réel, mais on ne peut pas additionner un entier et une chaîne de caractères.

Activité 2.
On souhaite :

- ▷ affecter la valeur 12 à la variable x ;
- ▷ affecter la valeur -3 à la variable y ;
- ▷ effectuer l'addition $x + y$, la soustraction $x - y$, la multiplication $x \times y$ et la division $\frac{x}{y}$, et les stocker dans de nouvelles variables.

1. Indiquez la séquence d'instruction permettant de réaliser cela.

2. Indiquez le contenu de chacune de ces variables, ainsi que leur type

3. Reprendre les mêmes questions si on affecte la valeur 13 à la variable x et la valeur -3 à la variable y.

4. Reprendre les mêmes questions si on affecte la valeur '14' à la variable x et la valeur -3 à la variable y.

VI. Les opérateurs de comparaison

Les opérateurs de comparaison sont résumés dans te tableau suivant :

Comparaison	Symbole	Exemple
égal à	$==$	$5 == 2$ renvoie `False`
différent de	$!=$	$5 != 2$ renvoie `True`
strictement supérieur	$>$	$5 > 2$ renvoie `True`
supérieur ou égal	$>=$	$5 >= 2$ renvoie `True`
strictement inférieur	$<$	$5 < 2$ renvoie `False`
inférieur ou égal	$<=$	$5 <= 2$ renvoie `False`

Activité 3.
On a saisi la séquence d'instruction suivante :

```
>>> a = 3
>>> b = 5
>>> c = 8
>>> d = -2
```

Indiquez l'instruction à saisir et le résultat attendu pour les comparaisons suivantes :

1. a + b est égal à c ;

2. a - b est différent de d ;

3. c / b est inférieur ou égal à a ;

4. (a + b) / c vaut 1.

Je retiens

- ▷ Python est un langage de programmation séquentielle, c'est à dire qu'il exécute les instructions les unes à la suite des autres ;
- ▷ Spyder est un IDE, c'est à dire une interface permettant de saisir le code Python à exécuter, puis de le compiler (vérifier sa syntaxe) et enfin de l'exécuter ;
- ▷ la coloration syntaxique permet d'identifier rapidement les variables, les instructions, les commentaires ... ;
- ▷ l'autocomplétion et les info-bulles permettent de faciliter la saisie ;
- ▷ les commentaires sont insérés après la balise `#` ;
- ▷ les types de base sont : `int`, `float`, `str` et `bool` ;
- ▷ l'instruction `type()` permet de connaître le type d'une variable ;
- ▷ les opérations élémentaires sont : `+`, `-` , `*`, `/`, `**`, `//` et `%` ;
- ▷ les opérateurs de comparaison sont : `==` , `!=` , `<` , `<=` , `>` , `>=`.

VII. Exercices

Exercice 1 (QCM).
Pour chacune des questions, plusieurs propositions sont faites, mais une seule est correcte. Laquelle ?

1. La réponse à l'instruction `1 == 3` est :

- ☐ une variable numérique ;
- ☐ une chaîne de caractères ;
- ☐ un booléen.

2. On souhaite mettre le résultat de l'opération `9 - 3 + (7 - 2) * 4` dans une variable `result`. On saisit ensuite l'instruction `type(result)`. Que va renvoyer la console ?

- ☐ `str` ;
- ☐ `bool` ;
- ☐ `int` ;
- ☐ `float`.

3. Dans un programme Python, si a prend la valeur '8' et b prend la valeur 5, que retourne la console si on saisit l'instruction `a + b` ?

- ☐ `'13'` ;
- ☐ `False` ;
- ☐ `13` ;
- ☐ `TypeError : must be str, not int.`

Exercice 2.

1. Écrivez la suite d'instruction à saisir dans un **notebook** permettant :

- d'affecter le nombre 25×10^{-12} à la variable *taille* ;
- de calculer le triple de ce nombre et de lui ajouter 7 ;
- de multiplier le résultat par le nombre 5 ;
- de soustraire le contenu de la variable *taille* du précédent résultat ;
- de stocker ce résultat final dans la variable *total* ;

2. Exécutez cette séquence d'instruction dans la console, puis saisissez l'instruction affichant le type de la variable *total*.

3. Modifiez le notebook pour qu'il effectue le même calcul lorsque taille prend la valeur 50. Quel sera le type de la variable *total* ?

Exercice 3.
Vous avez vu dans le cours de seconde que la relation permettant d'évaluer l'intensité de la pesanteur g selon l'altitude h est donnée par :

$$\mathrm{g} = G \times \frac{m_{Terre}}{(R_{Terre} + h)^2}$$

où $m_{Terre} = 6,0 \times 10^{24}$ kg, $R_{Terre} = 6\ 371\ 000$ m et $G = 6,67 \times 10^{-11}$. Toutes les distances doivent être exprimées en mètre.

1. Écrivez la suite d'instruction à saisir dans un **notebook** permettant d'évaluer l'intensité de la pesanteur à une altitude de 9 200 m.

2. Cette relation est valable sur toutes les planètes du système solaire. Évaluez l'intensité de la pesanteur sur la planète Mars (dont le rayon est de 3 396 km et la masse de $6,4185 \times 10^{23}$ kg), à la même altitude.

Exercice 4 (À faire dans un notebook).

1. Écrivez un programme Python qui calcule la moyenne arithmétique de trois nombres.
2. Écrivez un programme Python qui calcule la somme, la différence, le produit, le quotient et le reste de ces deux entiers.
3. Écrivez un programme Python qui calcule le volume de la sphère. On rappelle que le volume d'une sphère est donné par la formule : $V = \dfrac{4\pi R^3}{3}$.
4. On considère qu'un horaire s'exprime sous la forme " h m s ". Écrivez un programme qui, pour un horaire donné renvoie en sortie cet horaire exprimé en secondes.
5. Écrivez un programme Python qui échange les valeurs de deux variables en utilisant une variable supplémentaire.