Chapitre 1

Premiers pas en Python

Cours

Le programme de Numérique et Sciences Informatiques de première voie générale stipule qu'un langage de programmation est « nécessaire pour l'écriture des programmes : un langage simple d'usage, interprété, concis, libre et gratuit, multi-plateforme, largement répandu, riche de bibliothèques adaptées et bénéficiant d'une vaste communauté d'auteurs dans le monde éducatif. Au moment de la conception de ce programme, le langage choisi est Python version 3 (ou supérieure). »

Ce même programme indique qu'il n'est pas question de devenir un/une expert/e en Python. Quelques bases doivent néanmoins être maîtrisées et seront exposées dans cet ouvrage. L'environnement de développement en Python choisi dans ce livre est le logiciel gratuit Pyzo.

1 Un peu d'histoire de Python

Python est un langage de programmation dit *interprété*. Il a été développé en 1989 par Guido Von Rossum, à l'Université d'Amsterdam. Guido Von Rossum était un fan de la série Monthy Python's Flying Circus d'où l'orgine du nom de son langage de programmation.

Vocabulaire à connaître

Un langage *interprété* (contrairement à un langage compilé) est retranscrit en langage machine au fur et à mesure de son exécution via un interprète.

Python est devenu un langage de programmation généraliste très utilisé autant dans l'industrie que dans le milieu académique et scolaire. Il est très facile à apprendre, d'où ce choix pour un enseignement au lycée.

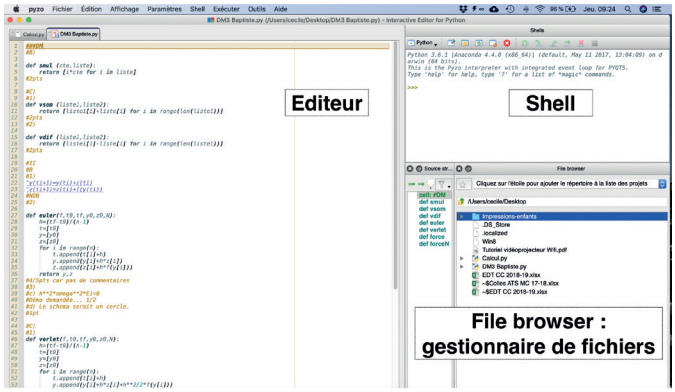
Python est un langage open source (libre de droit et gratuit développé et utilisé par une large communauté d'utilisateurs : soit 300 000 utilisateurs et plus de 500 000 téléchargements par an).

2 L'environnement Pyzo

Les différentes fenêtres de Pyzo

Pyzo va permettre de saisir des programmes (ou scripts) plus ou moins courts en langage Python et d'en visualiser les résultats. Tout d'abord, il faut lancer le logiciel Pyzo : pour cela il suffit de double-cliquer sur l'icône Pyzo (si Pyzo a bien été installé sur votre ordinateur). Les fenêtres les plus utiles pour la suite sont :

- le shell (pour la saisie de scripts très courts et la visualisation de résultats) ;
- l'éditeur (pour l'écriture de programmes plus ou moins longs que l'on peut sauvegarder) ;
- le gestionnaire de fichiers (pour visualiser les fichiers de travail).



Premiers calculs dans le shell

Le shell de Pyzo 4 invite à saisir des instructions avec >>> (ou `In[1]:` dans l'ancienne version du logiciel, à savoir Pyzo 3). On peut alors y taper :

```
>>> 1+1
```

Puis taper sur la touche Enter du clavier. Dans ce cas, le shell renvoie un nombre entier (aussi appelé *integer,* nous reviendrons là-dessus plus loin dans ce chapitre) :

```
2
```

On peut aussi choisir d'effectuer un calcul, par exemple 23/3 :

```
>>> 23/3
```

Une fois la touche Entrée enfoncée, le shell renvoie un nombre réel (appelé nombre *flottant* en Python) :

```
7.666666666666667
```

Les parenthèses s'emploient comme vous le feriez sur une calculatrice. La commande 2 ∗ 3 + 2 donne 8, là où la commande 2 ∗ (3 + 2) donne 10.

Il faut savoir que le calcul d'une puissance est prioritaire sur les autres opérations :

```
>>> 1+3**2
```

Le shell renvoie un nombre entier (aussi appelé *integer*) qui correspond au résultat du calcul $1 + (3^2)$:

```
10
```

Premiers programmes dans l'éditeur

On peut commencer par taper dans l'éditeur :

```
1.  a = 1
2.  b = 2
3.  print("Le calcul de a+b donne", a+b)
```

Remarque

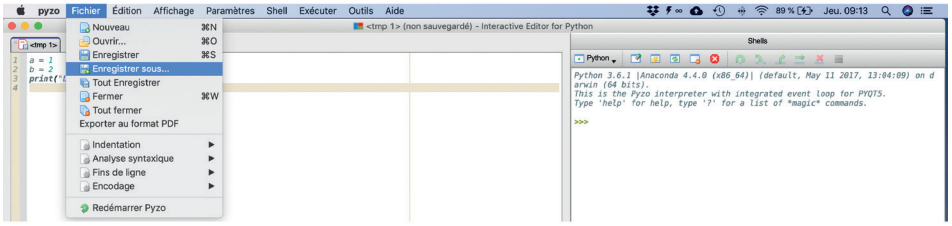
Les numéros 1 à 3 sont ici générés automatiquement par l'éditeur Pyzo dans la colonne de gauche (il ne faut pas les saisir à la main vous-même) et correspondent au numéro de chaque ligne.

Vocabulaire à connaître

Dans ce programme, on effectue une *affectation* des *variables a* et *b* (cela signifie que l'on *stocke* l'entier 1 dans la variable a, et 2 dans la variable b). On demande ensuite un *affichage* du résultat grâce à la fonction `print()`. Le mot `print` est un mot réservé à Pyzo, il s'affiche en bleu.

La phrase de commentaire "Le calcul de a+b donne" est appelée une *chaîne de caractères*, elle s'affiche en bleu clair.

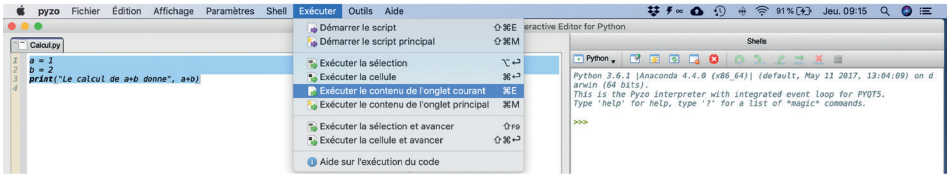
Une fois ces lignes tapées dans l'éditeur, on enregistre le programme (aussi appelé *script*) créé sous le nom *Calcul.py* (par exemple) avant de l'exécuter :

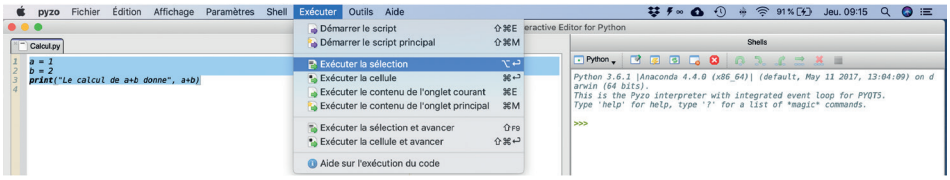


Remarque

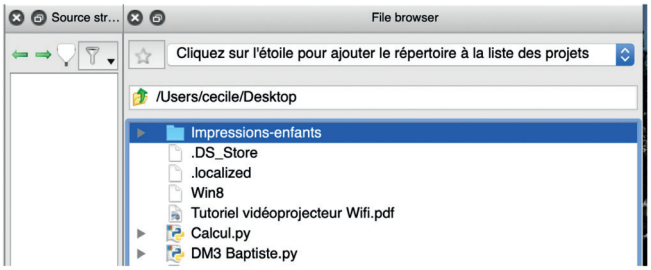
À noter que .py est l'extension qui correspond à un fichier Pyzo.

Une fois le fichier enregistré, on peut exécuter le programme. Il existe deux méthodes assez pratiques pour exécuter du code sous Pyzo. On peut exécuter tout un fichier . py (raccourci clavier : Ctrl+E) ou bien une partie d'un fichier : ce qui correspond à *exécuter le contenu de l'onglet courant* ou *exécuter la sélection*.





Une fois le fichier Calcul.py enregistré, on peut vérifier qu'il figure bien dans le répertoire *.../Info* par exemple :



Une bonne pratique en programmation est de *commenter* ses programmes. Le programme Calcul.py agrémenté de commentaires donne par exemple :

```
###Mon premier programme
#Affectations
a = 1
b = 2

#Calcul et affichage du résultat
print("Le calcul de a+b donne", a+b)
```

Remarque

Un commentaire est donc ajouté grâce à un ou plusieurs #. Aussi, tout ce qui suit un # sur une même ligne, n'est pas exécuté par Pyzo et s'affiche en vert dans l'éditeur.

3 Quelques types de base : les integers et les flottants

Les nombres en python peuvent être de différents *types*. Pour tester le type d'un nombre ou d'une variable, il suffit d'utiliser la commande `type( )`.

Les integers

On se place dans le shell (pour plus de simplicité) et on tape par exemple :

```
>>> type(3)
```

Le shell renvoie `'int'` pour *integer* (nombre *entier* en anglais) :

```
<class 'int'>
```

Remarque

On obtient le même résultat avec :

```
>>> type(-3)
```

Donc les entiers naturels et entiers relatifs sont des *integers* en Python.

Les flottants

On peut aussi tester le type d'un nombre réel ou le type du résultat d'un calcul qui donne lieu à un nombre réel :

```
>>> type(3.0)

>>> type(20/3)
```

Ces deux instructions renvoient chacune dans le shell le type *flottant* (float) qui correspond à un nombre réel :

```
<class 'float'>
```

4 La bibliothèque math et ses fonctions

La bibliothèque *math* contient des nombres préenregistrés comme le nombre π mais aussi des fonctions comme les fonctions trigonométriques (celles-ci seront définies et développées en classe de Terminale).

En mathématique	En Python
exp(x)	exp(x)
ln(x)	ln(x)
sin(x)	sin(x)
cos(x)	cos(x)
π	pi
$\sqrt{x}$	sqrt(x)

Cette bibliothèque peut être utilisée dans le shell mais l'utilisation de l'*éditeur* est plus simple dans le cadre de l'écriture d'un programme. Pour cela, on peut d'abord importer les fonctions (et nombres) souhaités de la bibliothèque *math*. On tape par exemple dans l'*éditeur* :

```
1. ###Fonctions mathématiques de la bibliothèque math
2. from math import cos, pi #on importe la fonction cosinus et le nombre
   pi exclusivement
3. print("cos(pi/2) =",cos(pi/2))
```

On exécute ce programme avec Ctrl+E (par exemple) et ceci donne dans le shell :

```
cos(pi/2) = 6.123233995736766e-17
```

Remarque

Le résultat du calcul $\cos\frac{\pi}{2}$ donne en réalité 0 sur votre calculatrice. Ceci nous permet de faire remarquer que la manipulation de nombres flottants entraîne *une erreur numérique* (ici l'ordre de grandeur de l'erreur est $1{,}0 \times 10^{-17}$). Il faudra en tenir compte par la suite.

Par exemple, si on compare le résultat du calcul de $\cos\frac{\pi}{2}$ à 0 avec :

```
4. cos(pi/2) == 0 # == est un test de comparaison
```

On obtient dans le shell :

```
False
```

Pyzo indique que le résultat de cette comparaison semble faux et que "$\cos\frac{\pi}{2} \neq 0$", ce qui n'a pas de sens d'un point de vue mathématique.

5 Les variables en Python

Affectation d'une ou plusieurs variables

Dans le programme Calcul.py, on a créé deux variables `a` et `b` et donc *déclaré* `a` et `b`. Pour cela, à chacune de ces variables, on a *affecté* une valeur.

```
1. a = 1
2. b = 2
```

Remarque

L'affectation d'une variable peut être effectuée dans le shell ou dans l'éditeur.

Vocabulaire à connaître

On parle de déclaration de variable lorsque celle-ci apparaît pour la première fois dans un programme et qu'une valeur lui est affectée.

Pour *déclarer* deux variables `a` et `b` en une seule ligne, on peut taper :

```
>>> a,b = 1,2
```

La valeur 1 est affectée à la variable `a` et la valeur 2 est affectée à la variable `b`. Ou bien si les variables doivent être égales, on peut taper :

```
>>> a,b = 2
```

La valeur 2 est alors affectée à chacune des deux variables `a` et `b`.

Conseil

Attention à ne pas confondre l'affectation = avec ==

```
1. #Une affectation de variable:
2. a = 1
3. #Une comparaison de valeur:
4. a == 1
```

On peut aussi déclarer une variable en lui affectant un nombre écrit en notation scientifique. Par exemple si $c = 1{,}2 \times 10^{-3}$:

```
>>> c = 1.2E-3
```

Ou encore :

```
>>> c = 1.2e-3
```

Remarque

La virgule se note . en Python.