Chapitre A

Bien démarrer

1 - Installation de Python sur votre machine

a) Téléchargement

Vous trouverez sur le site du livre : http ://prototheque.free.fr/ellipses/ en tapant le code INSTALLATION , un tutoriel vidéo présentant l'installation de la distribution EduPython qui sera utilisée dans ce livre. Pour installer EduPython, il suffit d'aller dans la rubrique téléchargement du site officiel : **http ://edupython.tuxfamily.org/** puis de suivre les instructions.

La distribution EduPython utilise l'excellent éditeur PyScripter qui ne fonctionne malheureusement que sous windows. Si vous utilisez un autre système d'exploitation, vous n'aurez qu'à utiliser n'importe quelle distribution Python sous Python 3.x. Par exemple, la distribution PYZO disponible sur le site : **http ://www.pyzo.org/** est un produit très complet et multiplateforme. Dans tous les cas, les programmes présentés dans ce livre ne dépendent pas du système d'exploitation que vous utilisez, c'est là tout l'intérêt de Python.

b) EduPython, AmiensPython, Python 2.x, Python 3.x ? ? ?

Expliquons ce qui se cache derrière ces mots....

Il existe sur le marché actuel deux versions majeures de Python : Python 2 (la dernière étant la 2.7) et Python 3 (la version actuelle est la 3.4). Le passage de Python 2 à Python 3 a induit quelques changements majeurs (par exemple la définition de la division n'est plus la même) rendant malheureusement incompatibles les programmes de la version 2 à la version

3. A l'heure actuelle Python 2 est encore assez utilisé, mais n'est plus mis à jour. Ce livre traite donc de Python 3, le langage qui est développé à ce jour.

En ce qui concerne AmiensPython, EduPython et Pyzo dont nous avons déjà parlé, il s'agit de distributions Python (un package comportant Python, un éditeur de programmes, ...) permettant d'utiliser Python.

Avec l'arrivée de l'algorithmique dans l'enseignement des mathématiques au lycée, un groupe de travail de l'académie d'Amiens constitué de 4 enseignants, Agnès BARAQUIN, François PREDINAS, Julien POLLET et moi-même sous la direction de Ludovic LEGRY, IA-IPR, a produit une distribution clé en main, portable et francisée pour l'usage de Python en classe. Cette distribution porte le nom d' AMIENSPYTHON , elle fonctionne sous Python 2.7. Une brochure à usage pédagogique a été réalisée comportant de nombreux exemples d'application.

Avec l'arrivée de Python dans les programmes du supérieur, il nous a semblé pertinent de proposer une version d'AmiensPython fonctionnant sous Python 3. Edupython a donc été créé comme un outil permettant de programmer sous Python 3.2.

2 - A la découverte de la console Python

a) Interface EduPython

Le logiciel PyScripter fait office d'éditeur pour EduPython. Si vous n'avez pas pu installer Edupython, les autres éditeurs ont un fonctionnement analogue. L'interface est découpée en plusieurs zones, en particulier :

- La zone de script qui permet de saisir des programmes plus longs et de les exécuter ensuite.
- La console qui permet d'exécuter en direct des commandes Python ; c'est aussi dans cette console que s'afficheront les résultats des programmes.

Dans la zone console de Python, on peut donc saisir des instructions qui seront exécutées immédiatement. Cette saisie se fait après les >>> que l'on appelle des **chevrons**.

Ainsi, on peut déjà, dans un premier temps, utiliser cette console comme une calculatrice. Voici quelques opérations possibles :

Opérateur	Effet
+, -, *	Respectivement la somme, la différence et le produit
a / *b*	Quotient décimal de *a* par *b*
a // *b*	Quotient entier de *a* par *b*
a % *b*	Reste entier dans la division de *a* par *b*
a ** *b*	Résultat de a^b. *Attention, le symbole ^ en Python a une autre signification.*

Sous Python 3.2, la division avec l'opérateur / renvoie donc le quotient d'une division décimale, par exemple `7/2` donne `3.5` alors qu'il donnait `3` avec les versions 2.x de Python. D'ailleurs, si vous utilisiez AmiensPython, cette définition de la division qui nous semblait plus pédagogique avait déjà été anticipée.

Les quelques exemples qui suivent vont vous permettre de prendre la console Python en main en jouant avec les nombres. Vous pouvez utiliser les touches [↑] et [↓] pour vous déplacer dans l'historique de ce que vous avez déjà tapé au fur et à mesure.

EX A1 [▶] *Ecrire deux séquences de calculs (vraiment) différentes pour afficher* `35` *en utilisant exactement 5 fois la touche* [5] *et autant que nécessaire les touches* [+], [-], [*], [/], [%], [(] *et* [)].

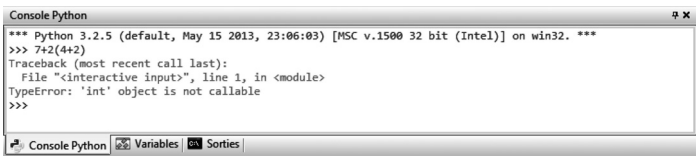
Quelques remarques supplémentaires :

- Le quotient q et le reste r de la division entière sont définis en Python grâce au théorème suivant :

  SI a et b sont deux nombres entiers relatifs avec b non nul
  ALORS il existe un unique couple d'entiers (q, r) vérifiant :

  $$\begin{cases} a = b \times q + r \\ |r| < |b| \\ r \text{ et } b \text{ sont de même signe} \end{cases}$$

  Nous verrons un peu plus loin qu'il faudra être vigilant lorsque l'on utilise des nombres négatifs.
- Le calcul de $a**b$ peut être effectué même lorsque b n'est pas entier. Dans ce cas, si $a > 0$ et $b \in \mathbb{R}$, Python utilise la formule vue en terminale : $a^b = e^{b \ln a}$ (si $a < 0$, c'est abordable dans l'ensemble des nombres complexes mais ne figure pas dans le programme du lycée).
- Le calcul suivant : `7+2(4+2)` n'a pas de sens : Python n'interprète pas les multiplications implicites, il faut donc taper `7+2*(4+2)`.



Si vous connaissez les puissances d'exposant réel, vous pouvez faire l'exercice qui suit :

EX A2 ▶▶▶ *Qu'affichent les calculs suivants ?*

A. `16 ** 0.5`
B. `16 ** 0.25`
C. `27 ** 1 / 3`
D. `27 ** (1 / 3)`

Comme en mathématiques, Python respecte les priorités des opérations. En informatique, ce concept porte le nom de **précédence** des opérateurs. Dans le cas du langage Python, les précédences sont définies ainsi (dans l'ordre croissant) :

+, -	Addition et soustraction
*, /, //, %	Multiplication, division, quotient et reste
$+x$, $-x$	Positif, négatif
**	Exponentiation

EX A3 ▶ *Qu'affichent les calculs suivants ? Après avoir cherché les réponses, ne pas hésiter à tester avec la console pour vérifier.*

A. `7 // 2`
B. `4 % 2`
C. `3 ** 3`
D. `-5 // 2`
E. `-14 // (-3)`
F. `5 ** 0 + 7`
G. `2 ** - 2`
H. `-2 ** 10`
I. `1 - 5 * 2`
J. `5 - 3 / - 2`
K. `- 13 % 3`
L. `13 % - 3`

b) Variables et affectations

L'avantage d'utiliser un langage de programmation pour effectuer des calculs réside dans le fait de pouvoir stocker les informations en mémoire. On peut, lorsque l'on réalise un programme ou un algorithme, utiliser des variables pour stocker les valeurs. Le fait de donner un nom à un calcul, un texte ou un autre objet s'appelle **l'affectation**.

En Algorithmique	On lit ...	En Python
$a \leftarrow 3$	*a* reçoit la valeur 3	`a = 3`

Dans la console Python, lorsque l'on affecte une valeur à une variable, le contenu de celle-ci n'est pas affiché. Il faut taper le nom de la variable pour afficher sa valeur.

Au moment d'une affectation, la quantité à droite du signe = est évaluée et stockée dans la variable de gauche. L'écriture `a = a + 1` a donc un sens en informatique . Cela signifie simplement que la variable `a` augmenté de 1.

Le symbole = joue donc un rôle d'affectation et non d'égalité. D'ailleurs, `b = a * 2` n'est plus vérifiée à la fin de notre exemple.

```
>>> a = 3
>>> b = a * 2
>>> b
6
>>> a + 1
4
>>> a
3
>>> a = a + 1
>>> a
4
>>> b
6
```

Voici donc un petit exercice permettant de travailler le vocabulaire mathématique et les opérations avec Python...

EX A4 *Si n est un entier positif, relier les expressions Python aux expressions françaises correspondantes :*

- `n + 1`
- `n // 10 % 10`
- `n // 10`
- `n // 2 * 2 + 1`
- `n ** 2`
- `n % 10`
- `n * 2`

- *Le chiffre des unités de n*
- *Le premier nombre impair qui suit n*
- *Le chiffre des dizaines de n*
- *Le nombre de dizaines de n*
- *Le carré de n*
- *Le double de n*
- *L'entier qui suit n*

En vous appuyant sur l'exercice précédent, vous pouvez alors tenter celui-ci.

EX A5

1. *Si n est une variable contenant un nombre à 2 chiffres non nuls, écrire une suite d'instructions permettant de stocker dans une variable p le nombre écrit "à l'envers" : si $n = 12$ alors $p = 21$.*
2. *Même question avec un nombre à (exactement) 3 chiffres.*

c) Affectations simultanées

Commençons par un premier exercice pour vous faire une idée.

EX A6 ★ *On considère la suite d'instructions ci-contre*

```
>>> x = 17
>>> y = 32
>>> x = x + y
>>> y = x - y
>>> x = x - y
```

1. *Que contiennent les variables x et y à la fin ?*
2. *Constate-t-on le même phénomène pour des valeurs quelconques de x et y ? (Le démontrer)*
3. *Ecrire une suite d'instructions produisant le même effet, mais sans effectuer de calculs (on pourra utiliser une troisième variable temporaire).*

Python propose une manière simple d'affecter simultanément des variables en utilisant une virgule, les deux premières lignes de l'exercice précédent peuvent être réduites en :

```
x, y = 17, 32
```

Comme pour l'affectation classique, lorsque l'on procède à une affectation simultanée, les quantités de droite sont évaluées dans un premier temps, puis dans un second temps, elles sont stockées dans les variables de gauche. Ainsi les 3 dernières lignes de l'exercice précédent peuvent tout simplement être transformées en :

```
x, y = y, x
```

EX A7 ▶ *Que contiennent les variables à la fin des programmes suivants ?*

1.
```
>>> a, b = 3, 2
>>> a = a - b
>>> b, a = a + 1, 2*b
>>> b = b % 3
>>> a = a - b // 2
```

2.
```
>>> x, y, z = 1, 2, 3
>>> x, y = y, z
>>> y, z = z, x
>>> y = z
>>> z = y
```

EX A8 ★★

1. *Parmi les instructions suivantes de la forme* `x, y = Q1, Q2`*, lesquelles peuvent s'écrire en deux lignes* `x = Q1` *puis* `y = Q2` *sans changer le résultat ?*

   a. `x, y = 17, 32`
   b. `x, y = 5, x - 3`
   c. `x, y = y, x`
   d. `x, y = x + 1, y + 2`

2. *Plus généralement, pour quelles valeurs des nombres* a, b, c *et* d*, peut-on remplacer l'affectation simultanée :*

```
x, y = a * x + b * y, c * x + d * y
```

par :

```
x = a * x + b * y
y = c * x + d * y
```

quelles que soient les valeurs données à **x** *et* **y** *?*

EX A9 ▶ ***La carte vitale.***

En France, le numéro de sécurité sociale présent sur la carte vitale est un numéro "signifiant" (c'est-à-dire non aléatoire) composé de 13 chiffres permettant d'identifier une personne, suivi d'une clé de contrôle de 2 chiffres pour déceler d'éventuelles erreurs de saisie.

Image par Giesesamvitale sur Wikipédia.

Par exemple, le premier chiffre indique le sexe (1 pour un homme, 2 pour une femme), les deux suivants l'année de naissance, suivis de deux chiffres pour indiquer le mois et ainsi de suite. Pour la carte fictive ci-dessus, le code est donc 2690549588157 *et la clé* 80*. Cette clé correspond à la différence entre* 97 *et le reste de la division du code dans la division par* 97*. Réaliser une suite d'instructions qui, connaissant le code* N *à 13 chiffres, calcule la clé de sécurité.*

3 - Les types d'objets

Il existe de nombreux types d'objets en Python, on peut même créer ses propres types d'objets ! En voici quelques-uns que nous allons régulièrement utiliser. Nous y reviendrons plus en détail plus loin dans le livre.

Dans la console (ou plus tard dans un programme), on peut demander de quel type est une variable à l'aide de la fonction **type** comme le montre l'exemple de droite.

```
>>> n = 5
>>> type(n)
<class 'int'>
>>> type(2 ** 2014)
<class 'int'>
>>> type(1/2)
<class 'float'>
>>> L = "Python"
>>> type(L)
<class 'str'>
>>> n < 1
False
>>> type(n < 1)
<class 'bool'>
```