Chapitre A

Prendre un bon départ

1 - Avant-propos

Un ordinateur sait très bien faire deux choses :

- Calculer, d'où le nom *computer* en anglais que l'on pourrait traduire par calculateur.
- Placer des données en mémoire, ce qui a donné le nom *ordinateur* en français dont l'origine quasi divine vient de « *celui qui met de l'ordre* ».

Excepté cela, un ordinateur ne sait pas faire grand-chose sans qu'on le lui ait appris... c'est le but de cet ouvrage. Nous allons découvrir la programmation autour du thème des jeux en suivant la progression que j'ai mise en place avec des élèves de seconde lors d'un enseignement appelé "2ISN" durant l'année 2011-2012.

Le jeu est ici un moyen et non l'objectif. Il s'agit là de débuter dans la programmation, vous ne trouverez d'ailleurs pas de techniques très poussées d'algorithmes d'optimisation de l'affichage ou d'algorithmes complexes d'intelligence artificielle.

Certaines notions, que je n'ai pas forcément eu le temps d'aborder avec les élèves, faute de temps, seront approfondies dans ce livre. Les élèves ont découvert le langage Python durant cinq mois, puis en équipe ont travaillé sur un projet de jeu durant le reste de l'année, complétant leurs connaissances au fur et à mesure des besoins. Le rendu du travail est visible à l'adresse : http://prototheque.free.fr/2ISN ou en saisissant le code `2ISN` sur le site du livre...

D'ailleurs vous trouverez régulièrement tout au long du livre des codes que vous pourrez saisir sur le site

http://prototheque.free.fr/ellipses

et ainsi obtenir des compléments (extraits de programmes, fichiers ressources : images, sons...). Si vous disposez d'un téléphone portable, vous pouvez scanner ce code flash pour atteindre la page directement.

A la fin de chaque chapitre vous trouverez les corrections des exercices en intégralité ou de manière partielle. N'hésitez pas à les lire même si vous avez réussi l'exercice. C'est souvent l'occasion d'apporter un certain nombre de nouvelles informations. Dans tous les cas, vous pourrez retrouver sur le site les programmes avec de nombreux commentaires.

2 - Remerciements

De nombreuses personnes ont rendu ce livre possible, à commencer par les premières personnes avec qui j'ai appris à programmer : Jean-Claude PLANTEGENEST, mon instituteur de CM2 avec qui j'ai réalisé en classe mes premiers programmes complexes en LOGO et Pierre MALLEJAC qui m'a aidé à faire mes premiers pas dans la programmation. A mon tour, j'ai voulu partager avec mes élèves ma passion de la programmation.

J'espère que vous aurez autant de plaisir à lire ce livre que j'en ai eu à l'écrire. Merci à mon épouse et mes enfants pour leur patience témoignée face aux heures que j'ai passées derrière l'ordinateur, à mes parents qui ont relu le livre dans son intégralité apprenant par la même occasion les bases de la programmation en essayant de réaliser les exercices !

Un grand bravo et merci aux élèves qui ont travaillé sur les projets durant une année, ainsi qu'aux élèves de seconde A qui ont réalisé de nombreux graphismes. D'autres graphismes proviennent du web, merci aux artistes pour leurs autorisations à placer leurs œuvres dans ce livre en particulier à Guillaume GREU auteur de nombreux dessins.

Enfin, un grand merci à Catherine DECAYEUX ainsi qu'à mes collègues et amis Guillaume MIANNAY et Fatima ESTEVENS pour leur relecture attentive et leurs précieux conseils de pédagogues avertis.

Ce livre est une réécriture d'une première édition pour Python 2. Les programmes et commentaires ont été mis à jour pour Python 3.

3 - Quelques algorithmes

Commençons par présenter les bases de la programmation via quelques algorithmes assez simples en langage naturel.

En programmation, on va heureusement pouvoir accéder facilement aux emplacements de la mémoire en leur donnant des noms, que l'on appelle variables. Nous allons donc pouvoir affecter des valeurs à ces variables. Nous noterons ← cette action **d'affectation** dans les algorithmes.

Un **algorithme** (nom qui vient du mathématicien Al-Khawarizmi) est une suite finie d'opérations ou d'instructions permettant de décrire un processus. Commençons donc par étudier quelques algorithmes assez simples en langage naturel, cela nous permettra de comprendre la manière de fonctionner d'un ordinateur.

Exemple :

EXEMPLE 1 :
$a \leftarrow 2$
$m \leftarrow 5$
$a \leftarrow m + a$
$a \leftarrow a + 1$
$m \leftarrow a \times m$
$m \leftarrow 2 \times m$

Ligne	a	m
1	2	.
2	2	5
3	7	5
4	8	5
5	8	40
6	8	80

Remarquons que quand on réalise une affectation, la machine calcule le membre de droite et l'affecte à la variable de gauche. Ainsi l'instruction $a \leftarrow a + 1$ revient à dire que a augmente de 1.

A la fin de ce programme, a vaut donc 8 et m vaut 80. Mais comment fera-t-on pour voir les valeurs finales des variables a et m si on ne demande pas leur affichage ? Pensez-y : si on demande d'effectuer un calcul, l'ordinateur calcule...et c'est tout !

Maintenant que vous avez compris l'essentiel, il est temps de pratiquer. Les exercices qui vont vous être proposés sont progressifs, n'hésitez pas à revenir sur les pages précédentes en cas de doute.

EX A1 Que valent les variables a et b à la fin du programme ?

$a \leftarrow 0$
$b \leftarrow 1$
$a \leftarrow a + b$
$b \leftarrow a \times b$
$a \leftarrow a - 2$

EX A2 1. Qu'affiche ce programme pour $a = 7$ et $b = 21$?
2. Qu'affiche ce programme pour $a = 13$ et $b = 3$?
3. Plus généralement, si a et b sont deux nombres décimaux. Que fait ce programme ?

```
DEMANDER a et b
SI a < b ALORS
  | AFFICHER b
SINON
  | AFFICHER a
FINSI
```

EX A3 1. Qu'affiche le programme si on entre 14 pour a et 3 pour b.
2. Plus généralement, si a et b sont deux entiers strictement positifs. Que fait ce programme ?

```
DEMANDER a et b
TANT QUE b ⩽ a
  | a ← a − b
FIN TANT QUE
AFFICHER a
```

EX A4 1. Qu'affiche ce programme ?
2. Que se passe-t-il si l'on répète 40 fois au lieu de 4 fois ?

```
X ← 18
REPETER 4 fois
  | SI X est un multiple de 2 ALORS
  |   | X ← (X + 10)/2
  | FINSI
  | SI le reste de X ÷ 3 vaut 2 ALORS
  |   | X ← (X + 1)/3
  | FINSI
FIN REPETER
AFFICHER X
```

4 - Solutions des exercices

Retour sur l'exercice A1 En détaillant étape par étape :

```
a ← 0
b ← 1
a ← a + b
b ← a × b
a ← a − 2
```

Ligne	a	b
1	0	.
2	0	1
3	1	1
4	1	1
5	-1	1

A la fin du programme $a = -1$ et $b = 1$.

Retour sur l'exercice A2 1. Si on entre $a = 7$ et $b = 21$, Comme $7 < 21$, on affiche b soit 21.
2. Si on entre $a = 13$ et $b = 3$, Comme 13 n'est pas inférieur à 3, on affiche a soit 13.
3. Dans le cas général, lorsqu'on entre deux nombres, le programme affiche le plus grand des deux.

Retour sur l'exercice A3
1. On pourrait aussi présenter le processus sous forme d'un tableau : Si on entre $a = 14$ et $b = 3$, le programme affiche 2.

a	b	$b \leqslant a$?
14	3	OUI
11	3	OUI
8	3	OUI
5	3	OUI
2	3	NON

2. Plus généralement, le programme retire la valeur a tant que $b \leqslant a$, c'est-à-dire tant que $a - b \geqslant 0$. Ce qui revient à dire que l'on affiche le reste de la division euclidienne de a par b. En effet dans l'exemple précédent, on a retiré 4 fois 3, donc $14 = 4 \times 3 + 2$.

Retour sur l'exercice A4 Encore une fois nous allons présenter le cheminement dans un tableau :

X	X multiple de 2 ?	X	Reste de X ÷ 3 = 2 ?	X
18	OUI	14	OUI	5
5	NON	5	OUI	2
2	OUI	6	NON	6
6	OUI	8	OUI	3

Si on continue à répéter les boucles, il ne se passe plus rien puisque :

- 3 n'est pas multiple de 2.
- Le reste de la division de 3 par 3 vaut 0.

Chapitre B

Premiers pas en Python

Il existe de très nombreux langages de programmation. Parmi eux, le langage Python. Ce langage, de par son mode d'écriture très structuré, est un très bon outil pédagogique pour commencer à programmer car (dans un premier temps) les programmes vont fortement ressembler à nos algorithmes écrits en anglais.

Le Python est un langage très apprécié des scientifiques de par sa puissance de calcul et les nombreuses bibliothèques qu'il contient. En tant que professeur de mathématiques, cette dualité simplicité-puissance m'a séduit.

1 - Installation de Python

Il existe une multitude de distributions Python, de la plus simple à la plus évoluée. Pour l'enseignement des mathématiques et de l'ISN au lycée, une distribution initiée par l'académie d'Amiens a été réalisée : EDUPYTHON que vous pouvez télécharger à l'adresse.

http://edupython.tuxfamily.org/

Il s'agit là d'une distribution clé en main pour commencer sans stress, d'autant que cette version est portable et peut donc être installée sur une machine (sous windows) ou sur une clé USB pour plus de mobilité. L'avantage donc de cette version est qu'elle vous permettra de réaliser tous les programmes de ce livre. En particulier, Python n'ayant pas été conçu pour réaliser des jeux, nous avons embarqué aussi dans cette distribution toutes les bibliothèques nécessaires.

Les exemples de ce livre sont traités en Python 3.2 avec cette distribution mais vous pouvez utiliser un autre environnement de programmation. Attention cependant si vous décidez d'utiliser une solution utilisant Python 2.x, des différences existent entre Python 2 et Python 3.

2 - Premiers programmes

Le plus simple pour commencer est de faire tourner quelques programmes pour prendre en main l'interface. Saisissez et exécutez les programmes suivants :

Programme 1 :

```
age=eval(input("Entrez votre age"))
if age < 12 :
    print ("Ce jeu est déconseillé aux moins de 12 ans !")
else :
    print ("Début de la partie")
```

Programme 2 :

```
# La table des 9...
table = 0
while table <= 90 :
    print (table)
    table = table + 9
```

EX B1 Après avoir exécuté tous ces programmes, devinez les significations des instructions **`print, eval input, if, else, while`**.

3 - Fonctions Python

Voici donc quelques fonctions déjà étudiées précédemment :

Commande	Effet
`print`(″*texte*″)	Affiche le mot ″texte″.
`print`(*t*)	Affiche la valeur de la variable *t*.

Commande	Effet
v=**eval(input(***texte***))**	Affiche la phrase *texte* et attend que le joueur entre une valeur numérique qui sera stockée dans la variable numérique *v*.
if *cond* **:** BLOC	Effectue tout le BLOC d'instructions dès lors que la condition *cond* est vérifiée.
if *cond* **:** BLOC1 **else :** BLOC2	Effectue tout le BLOC1 d'instructions dès lors que la condition *cond* est vérifiée. Dans le cas contraire, c'est le BLOC2 qui est exécuté.
for *i* **in range(**N**) :** BLOC	Répète N fois le BLOC d'instructions. La variable *i* prend successivement les valeurs de 0 à N – 1.
while *cond* **:** BLOC	Répète le BLOC d'instructions tant que la condition *cond* est vérifiée.

Précisons quelques points :

- N'oubliez pas le **eval** avant le **input** lorsque vous voulez demander au joueur d'entrer un nombre, sinon votre réponse sera traitée comme du texte, et vous risquez d'avoir des surprises quand vous lui demanderez de le multiplier par 2. Essayez par exemple ce programme :

```
a = input('Entrez un nombre à 2 chiffres')
print (5*a)
```

- En Python, c'est la mise en forme du programme qui détermine son architecture. Pour déclarer des blocs d'instructions, on les décale (on dit qu'on réalise des indentations). Voici deux programmes qui se ressemblent :

```
#Programme 1 :
n = eval(input("Entrez votre chiffre porte-bonheur"))
if n == 7 :
    print ("7, comme 60% des personnes")
    print ("c'est un bon choix")
```