Chapitre 1

Introduction

La programmation a été introduite peu à peu dans l'enseignement : avec la spécialité ISN en terminale S (2012), en classes préparatoires (2013), puis dans l'enseignement d'exploration ICN en seconde et l'option ISN en première. Depuis la rentrée 2017, elle est présente dans le programme de mathématiques en seconde. Une simplification de l'écriture des algorithmes est en même temps préconisée : la syntaxe est simplifiée, la déclaration des variables et la gestion des entrées-sorties sont supprimées.

Concernant la syntaxe, le choix est fait ici d'utiliser " = " pour indiquer une affectation plutôt que " $\leftarrow$ " et " == " pour tester une égalité plutôt que " = ", ce qui est pratiqué dans les langages les plus courants comme C ou C++, Java et Python. La confusion entre les symboles " = " et " == " dans l'écriture d'un programme est habituelle chez les débutants et l'utilisation d'une même syntaxe dans les algorithmes et les programmes peut contribuer à éviter plus rapidement ce type d'erreur.

1.1 Algorithmes et programmes

1.1.1 Ecriture d'un algorithme

Un algorithme est une suite d'instructions qui, à partir d'un ensemble donné de valeurs (entrée), produit, après un nombre fini d'étapes, un autre ensemble de valeurs (sortie).

Jusqu'à récemment, les élèves travaillaient essentiellement sur des algorithmes avec papier ou logiciel. Puis ils apprenaient à les programmer sur une calculatrice, toujours à portée de main. Les écritures d'un algorithme et d'un programme étaient alors très semblables, avec des entrées-sorties, un affichage des résultats et sans aucune programmation fonctionnelle.

Il s'agit maintenant de programmer des algorithmes sur ordinateur en langage Python dont l'un des principaux intérêts est de pouvoir être utilisé comme langage

de description d'algorithmes tout en étant très épuré par rapport au langage utilisé par les calculatrices courantes.

Il est aussi recommandé de s'entraîner le plus tôt possible à écrire des fonctions qui sont ensuite utilisées dans une console, d'où l'intérêt d'un langage interprété.

Enfin, le temps passé sur un ordinateur par des élèves au lycée est insuffisant pour acquérir de bonnes pratiques. Il est donc important de les habituer à écrire des algorithmes avec une structure et une syntaxe semblables à celles d'un programme.

Exemple

Etant données les coordonnées de deux points A et B d'abscisses différentes, calculer les nombres m et p de l'équation réduite $y = mx + p$ de la droite (AB).

Version 1

Variables	xA est un nombre réel yA est un nombre réel xB est un nombre réel yB est un nombre réel m est un nombre réel p est un nombre réel
Entrées	Saisir xA Saisir yA Saisir xB Saisir yB
Traitement	m prend la valeur $(yB - yA)/(xB - xA)$ p prend la valeur $yA - m \times xA$
Sortie	Afficher m Afficher p

Version 2

Equation(xA, yA, xB, yB)
$m = (yB - yA)/(xB - xA)$
$p = yA - m \times xA$
renvoyer m et p

Nous voyons sur cet exemple, qui peut sembler un peu extrême mais qui est très courant, les quelques règles à appliquer dans l'écriture d'un algorithme.

La colonne de gauche est supprimée.

La déclaration des variables, parfois longue et fastidieuse est supprimée. Dans les langages de programmation courants, cette déclaration est très souvent associée à l'initialisation, ce couple constituant la définition d'une variable. L'affectation

d'une valeur à une variable est notée avec le signe =, comme nous l'écrivons dans nos programmes en Python.

La gestion des entrées-sorties reste importante. Il s'agit d'utiliser des données entrées au clavier, les lire dans un fichier, afficher des résultats à l'écran ou les écrire dans un fichier. Cette partie est traitée dans le dernier chapitre. Mais elle n'a pas d'intérêt dans l'écriture d'un algorithme particulier, c'est-à-dire le traitement des données qui est la partie à travailler, et peut entraîner une perte de temps inutile au moment de la programmation de cet algorithme.

L'indentation, décalage vers la droite d'une ligne, ou d'un ensemble de lignes, permet de délimiter clairement un bloc d'instructions.

pour ...
    si ...
        ...
    sinon
        ...

Les notations de type "Fin pour" ou "Fin si" par exemple, obligatoires avec les calculatrices avec lesquelles il n'y a pas d'indentation possible, ne sont donc plus nécessaires ici.

Il est possible d'utiliser une instruction qui permet de renvoyer une ou plusieurs valeurs.

Un exemple

Ecrire l'algorithme d'Euclide qui prend en entrée deux entiers naturels non nuls a et b, et produit en sortie leur pgcd d. L'énoncé précise les entrées et sorties, le travail est simplement d'écrire le traitement.

Nous supposons ici que `reste(a,b)` a pour valeur le reste r obtenu par la division euclidienne de a par b : $a = bq + r$ avec $0 \leq r < b$.

Nous écrivons `d!=0` qui signifie d différent de 0.

PGCD(a, b)
$d = 1$
tant que d != 0
    $d =$ reste(a, b)
    $a = b$
    $b = d$
renvoyer a

1.1.2 Ecriture d'un programme

Nous continuons avec l'algorithme de calcul du pgcd de deux entiers naturels. Nous utilisons pour cela l'algorithme d'Euclide.

Avec une calculatrice, un programme pourrait s'écrire sous cette forme :

PROGRAM : PGCD
Input "A = ", A
Input "B = ", B
$1 \rightarrow$ D
While D $\neq 0$
reste(A,B) $\rightarrow$ D
B $\rightarrow$ A
D $\rightarrow$ B
End
Disp A

Avec les calculatrices courantes, les entrées-sorties et l'affichage des résultats sont obligatoires.

Ce programme peut se traduire directement en langage Python :

```
# pgcd(a,b)
a=int(input("a= "))
b=int(input("b= "))
d=1
while d!=0:
    d=a%b
    a=b
    b=d
print(a)
```

Mais sous cette forme, chaque fois que nous voulons obtenir le pgcd de deux nombres, nous devons revenir au programme et le faire interpréter par Python.

Il est beaucoup plus pratique d'écrire une fonction, qui est chargée puis appelée autant de fois que nécessaire avec les valeurs choisies de a et b. Les entrées seront effectuées par l'appel de la fonction avec les paramètres choisis, pour obtenir en sortie la valeur renvoyée par la fonction.

```
def pgcd(a,b):
    d=1
    while d!=0:
        d=a%b
        a=b
        b=d
    return a
```

Cette fonction est chargée dans l'interpréteur et peut être utilisée à volonté :

```
>>> pgcd(15,12)
>>> 3
>>>
>>> pgcd(15,28)
>>> 1
>>>
```

Ce programme est pratiquement identique à l'algorithme du pgcd écrit plus haut. Nous avons écrit notre première fonction.

Note : une fonction est un élément d'un programme. Son écriture dépend du langage de programmation utilisé.

Il est aussi possible d'ajouter après la fonction une gestion des entrées-sorties qui permet d'utiliser le programme dans une console, (donc avec un double-clic sur le fichier). C'est ce qui est pratiqué avec un langage de programmation qui compile le fichier programme en un exécutable.

```
def pgcd(a,b):
    d=1
    while d!=0:
        d=a%b
        a=b
        b=d
    return a

n=int(input("n= "))
p=int(input("p= "))
print(pgcd(n,p))
```

Ce programme peut être utilisé en Python, avec la console, petite fenêtre noire qui s'ouvre après un double-clic sur le nom du fichier, et se ferme dès la fin de l'exécution. (Pour empêcher une fermeture immédiate, il suffit d'ajouter, à la fin de ce programme, une dernière ligne comme `input("?")`).

Mentionnons une difficulté réelle : après avoir pratiqué la programmation sans fonction sur calculatrice, il n'est pas simple de changer d'habitudes et de produire un code correct. On voit parfois des programmes comme celui qui suit.

```
def pgcd(a,b):
    a=int(input("a ? "))
    b=int(input("b ? "))
    d=1
    while d!=0:
        d=a%b
        a=b
        b=d
    print(a)
    return a
```

Bien entendu, ce programme peut être exécuté.

Si on retire les lignes `input...` et `print...`, la fonction est bien celle attendue. Mais si on retire les lignes `def...` et `return...`, l'algorithme est aussi correct et le programme permet d'obtenir le résultat souhaité.

C'est en fait un mixage, un mélange entre un programme sans fonction sur calculatrice, et l'utilisation d'une fonction non maîtrisée. La première ligne peut être pensée comme étant le nom du programme avec la déclaration des variables. La suite est un programme écrit comme avec une calculatrice courante. La ligne `return...` est parfois ajoutée, sans trop savoir pourquoi. En effet, `print` et `return` produisent le même effet dans l'interpréteur Python !

Remarque : il est tout à fait possible d'écrire une fonction sans paramètre.

```
def pgcd():
    a=int(input("a ? "))
    b=int(input("b ? "))
    d=1
    while d!=0:
        d=a%b
        a=b
        b=d
    return a
```

Mais cette fonction ne peut servir qu'à un utilisateur devant sa machine et n'a aucun intérêt en dehors de ce cadre. Elle ne peut pas être utilisée par une autre fonction ou un autre programme, alors que cela constitue l'un des principaux objectifs.

Aborder très tôt l'écriture de fonctions dans un programme est nécessaire. Cela peut aussi être un plus en mathématiques où la notion de fonction, même un peu différente, n'est parfois pas maîtrisée jusqu'en terminale.

Il reste que la gestion des entrées-sorties apporte la convivialité, le dialogue avec la machine, essentiel en seconde pour intéresser des élèves qui ont besoin de dialoguer avec la machine et savent que leurs programmes ne seront jamais utilisés par quelqu'un d'autre ! C'est pourquoi elle est traitée dans le dernier chapitre.

En conclusion, penser, écrire un algorithme est le point central. La traduction dans un langage de programmation est ensuite plus ou moins simple suivant le langage. Elle est presque immédiate en Python, et c'est pourquoi, dans la suite, nous confondrons programme en Python et algorithme.

Ajoutons que le choix parmi les nombreuses possibilités d'écriture est souvent une affaire de goût, d'habitude. Par exemple, les instructions suivantes vont avoir exactement le même effet :

```
if a<=b:
    x=a
else:
    x=b

# ou bien
x=a if a<=b else b

# ou bien
x=[a,b][a>b]
```

1.2 Environnement de développement

Si ce n'est pas déjà fait, il faut commencer par installer la partie logicielle qui permet de travailler avec le langage Python. Une description est donnée au dernier chapitre.

1.2.1 Présentation

Le langage de programmation Python, introduit par Guido von Rossum en 1990, est un langage très répandu dont les principaux avantages sont la facilité d'apprentissage et la simplicité d'utilisation. Il est possible d'écrire du code en Python sur un ordinateur fonctionnant avec Linux, MacOS ou Windows, ainsi que sur un smartphone ou certaines calculatrices.

Un environnement de développement **Idle**, **Pyzo** ou **Spider** est une application qui regroupe en particulier un éditeur de textes et un interpréteur.

Un double-clic sur "Python IDLE" et la fenêtre "Python shell" (l'interpréteur Python) s'ouvre et affiche un message d'information.

```
Python 3.6.3 (v3.6.3:2c5fed8, Oct  3 2017, 18:11:49)
[MSC v.1900 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more
information.
>>>
```

Un double-clic sur "Pyzo" et la fenêtre "Interactive Editor for Python" s'ouvre et affiche plusieurs fenêtres dont celle de l'interpréteur, (le shell) à droite, qui contient par exemple :

```
Python 3.6.1 |Continuum Analytics, Inc.| (default,
May 11 2017, 13:25:24) on Windows (64 bits).
This is the Pyzo interpreter with integrated event
loop for PYQT5.
Type 'help' for help, type '?' for a list of *magic*
commands.

>>>
```

Nous pouvons écrire directement dans cette fenêtre, après l'invite " >>> ", les différentes commandes Python, c'est-à-dire évaluer des expressions ou exécuter des instructions de manière interactive. Il est aussi possible d'écrire dans l'éditeur et d'enregistrer le contenu dans un fichier avec l'extension .py. Avec Idle, utiliser le menu **File** puis **New File** et une deuxième fenêtre s'ouvre. Cette fenêtre est un éditeur de texte. Avec Pyzo, l'éditeur est déjà ouvert, c'est la fenêtre de gauche.

Lorsque le fichier est écrit, on peut le sauvegarder à partir du menu **File** puis **Save As ...** pour choisir le dossier de destination et entrer le nom du fichier. Pour interpréter le fichier, il faut passer par le menu **Run** ou utiliser les raccourcis clavier **F5** avec Idle ou **ctlr+E** avec Pyzo. Il est aussi possible avec Pyzo de changer la langue par le menu **Settings** puis **Select language**.

1.2.2 Utilisation

Dans l'interpréteur, écrire `a='bonjour'` et appuyer sur la touche "entrée" (pour la suite, à la fin de chaque ligne, appuyer sur la touche "entrée").

Ecrire ensuite `b='tout le monde'`, puis à la ligne suivante `a+b`.

Le résultat s'affiche sur la ligne suivante.

```
>>> a='bonjour'
>>> b='tout le monde'
```