Chapitre 1

Algorithmique et programmation

L'algorithmique et la programmation deviennent des parties importantes des mathématiques mais aussi de la société civile et du quotidien. Tous vos appareils connectés sont gérés par des algorithmes et l'avenir semble aller tout droit vers le "tout algorithme". Les voitures autonomes, les téléphones, des tablettes, des objets connectés, les moteurs de recherches sont des outils qui vont très certainement devenir incontournables. Les métiers de demain devraient aussi devenir des métiers de programmeur. Il est donc important de se former très tôt à la programmation.

Synthèse de cours

1. Initiation à l'algorithmique

Définition 1 – Un algorithme

Un algorithme est une suite finie d'instructions permettant de résoudre un problème donné.

Exemples

• Dans la vie courante : une recette de cuisine, un mode d'emploi pour construire un meuble, un trajet sur une carte routière, une notice d'utilisation, etc.

• Dans le domaine des mathématiques : déterminer le PGCD de deux nombres (algorithme d'*Euclide*), résoudre une équation par approche de la solution, calculer une longueur, calculer une image, déterminer à partir de quel moment une expression algébrique dépasse une valeur, etc.

Utilités

Les algorithmes peuvent permettre d'effectuer des tâches répétées mais également de résoudre des problèmes plus facilement "qu'à la main". Tous les objets de la vie moderne sont gérés par des algorithmes. Les ordinateurs, les logiciels, Internet, les moteurs de recherches, les calculatrices, les téléphones portables, les objets connectés, etc. Il est donc important d'apprendre à les comprendre et à les construire.

Un algorithme doit être compréhensible pour tout le monde. Il est ensuite traduit dans un programme dans un langage informatique afin qu'une machine puisse l'exécuter simplement et avec efficacité.

Définition 2 – Un programme

Un programme est la traduction d'un algorithme dans un langage adapté à la machine utilisée.

Les langages les plus courants sont Python, C, C++, Java, Php, Pascal, Lisp.

Ce chapitre n'abordera que le langage Python comme il est conseillé dans les programmes du lycée. Pour traduire un algorithme en langage Python, on utilisera l'annexe de ce livre qui donne les principales instructions en Python. Il est fortement conseillé de lire cette annexe dès maintenant pour pouvoir traduire les algorithmes en langage Python le plus vite possible.

Le programme de première pour la spécialité mathématique étant très exigeant pour la partie Python, il faut rapidement travailler ce chapitre et vous familiariser avec ce langage.

Propriété 1 – Structure d'un algorithme

▷ Un algorithme se présente souvent sous la forme suivante :

Nom de l'algorithme	
Déclaration des variables	
	Listes, noms et description des variables que l'on va utiliser dans l'algorithme.
Initialisation	
	Si besoin, lorsque l'on doit donner une valeur initiale à une variable.
Traitement	
	Liste des instructions pour répondre au problème.
Sortie	
	Affichage du (ou des) résultat(s) attendu(s).

Exemple
L'instruction $X \leftarrow a$ signifie que la valeur de la variable X reçoit la valeur de la variable qui se nomme a.

Algorithme 1	
Déclaration des variables	
	X, Y : des nombres réels.
Initialisation	
	Saisir la valeur de X.
Traitement	
	$Y \leftarrow X - 2$ $Y \leftarrow Y \times Y$ $Y \leftarrow 3 \times Y$ $Y \leftarrow 4 - Y$
Sortie	
	Afficher la valeur de Y.

L'algorithme 1 permet de calculer l'image d'un réel x par la fonction

$$f : x \mapsto 4 - 3(x-2)^2$$

L'instruction $Y \leftarrow X - 2$ signifie que la variable Y prend la valeur de la variable $X - 2$.
L'algorithme peut être traduit de cette façon :

$$X \to X - 2 \to (X-2)^2 \to 3(X-2)^2 \to 4 - 3(X-2)^2$$

Programme Python
Lorsque vous voyez # cela signifie que c'est un commentaire qui sert juste à expliquer une instruction mais qui ne rentrera pas en compte dans l'exécution du programme.

```
#importe les fonctions de la bibilothèque math
from math import *
x=float(input("Donner la valeur de x: "))
Resultat=x-2
Resultat=Resultat**2
Resultat=3*Resultat
Resultat=4-Resultat
print(Resultat)
```

Mais on peut aussi le faire en définissant une fonction.
Une fonction en Python est de la forme :

```
def NomFonction(liste des Variables):
    # Instructions permettant de déterminer un résultat.
    return Resultat
```

Par exemple pour écrire $f(x) = 3x^2 + 5x - 1$ en Python on écrit :

```
from math import*
def f(x):
    ImageDeX=3*x**2+5*x-1
    return ImageDeX
```

Traduction de l'exemple précédent en fonction Python :

```
from math import *
def Fonction01(x):
# def permet de définir une fonction
# avec son nom et ses variables.
# Les : permettent au programme de
# définir le début de la fonction
# et de mettre des décalages aux lignes suivantes.
    Resultat=x-2
    Resultat=Resultat**2
    Resultat=3* Resultat
    Resultat=4-Resultat
    return Resultat
```

2. Tests et boucles

Les tests

Pour résoudre certains problèmes il est important, dans certains cas, de faire des tests pour savoir si l'on doit effectuer une instruction ou pas. Par exemple, si on doit calculer un inverse, il faut tester si le nombre est différent de 0 ou pas. Si on veut calculer la racine carrée d'un réel, il faut tester si le nombre est positif ou pas. Si on veut savoir si un nombre est pair, il faut tester si le reste par la division euclidienne par 2 est nul ou pas, etc.

Définition 3 – Instruction conditionnelle

Effectuer un test, revient à écrire **une instruction conditionnelle**. L'instruction conditionnelle effectue des instructions à condition qu'un test soit validé.

Dans un algorithme, on code l'instruction conditionnelle de la façon suivante :

Si **condition validée**	
Alors	
	instruction 01
	instruction 02
	instruction 03
Sinon	
	instruction 04
	instruction 05
	instruction 06
Fin du si	

Exemple

Algorithme 2	Ecart entre entiers
Déclaration des variables	X, Y, Ecart : des nombres entiers
Initialisation	Saisir la valeur de X
	Saisir la valeur de Y
Traitement	
Si $X \leq Y$	
Alors	Ecart $\leftarrow Y - X$
Sinon	Ecart $\leftarrow X - Y$
Fin du Si	
Sortie	Afficher la valeur "Ecart".

Programme Python

```
x=float(input("Donner la valeur de x: "))
y=float(input("Donner la valeur de y: "))
if x<=y:
    Ecart=y-x
else:
    Ecart=x-y
print(Ecart)
```

Mais on peut aussi le faire en définissant une fonction

```
def Ecart(x,y):
    if x<=y:
        return y-x
    else:
        return x-y
```

Les boucles

Définition 4 – Boucle itérative

Lorsque l'on doit répéter une instruction un nombre de fois connu à l'avance, on utilise **une boucle itérative**.

Dans un algorithme, une boucle itérative est codée de la façon suivante :

Pour variable allant de	la valeur Début à la valeur Fin
Faire	
	instruction 01
	instruction 02
	instruction 03
Fin du Pour	

La variable utilisée dans la boucle "Pour" est appelée un "compteur" et à chaque étape sa valeur est automatiquement augmentée de 1.

Exemple

Algorithme 3	Somme des 100 premiers entiers
Déclaration des variables	
	S, I : des nombres entiers.
Initialisation	
	$S \leftarrow 0$
Traitement	
Pour I allant de 1 à 100	Faire
	$S \leftarrow S + I$
Fin du Pour	
Sortie	
	Afficher la valeur de S.

Programme Python

```
from math import *
Somme=0
for i in range (1,101):
# Dans une boucle for en Python
# il faut faire varier le compteur
# à un rang de plus que la valeur voulue.
# Pour i variant de 1 à 100 se traduit par
# for i in range (1,101)
    Somme=Somme+i
print(Somme)
```

Mais on peut aussi le faire en définissant une fonction

```
from math import *
def Somme():
    S=0
    for i in range(1,101):
        S=S+i
    return S
```

Définition 5 – Boucle conditionnelle

Lorsque l'on doit répéter une instruction sans connaître le nombre d'itérations, on utilise **une boucle conditionnelle**. La boucle est répétée tant que la condition indiquée est vérifiée.

Dans un algorithme une boucle conditionnelle est codée de la façon suivante :

Tant que condition vérifiée	
Faire	
	instruction 01
	instruction 02
	instruction 03
Fin du Tant que	

On utilise régulièrement des compteurs à l'intérieur de la boucle conditonnelle, mais il faut faire attention à bien l'initialiser et à l'incrémenter à l'intérieur de la boucle.

Exemple

Algorithme 4	Somme des 100 premiers entiers
Déclaration des variables	
	S, I : des nombres entiers.
Initialisation	
	$S \leftarrow 0$
	$I \leftarrow 0$
Traitement	
Tant que $I \leq 100$	Faire
	$S \leftarrow S + I$
	$I \leftarrow I + 1$
Fin du Tant que	
Sortie	
	Afficher la valeur de S.

Programme Python

```
from math import *
Somme=0
```