Bases d'algorithmique

Christophe ROSSIGNOL*

Année scolaire 2021/2022

Table des matières

Liste des algorithmes

En préliminaire à ce chapitre, voir le TD n°1 d'initiation à l'algorithmique.
Pour toute question relative au langage de programmation Python, on pourra se référer aux pages 22-23[Magnard]

1 Un peu de vocabulaire

1.1 Qu'est-ce qu'un algorithme ?

Définition : Un algorithme est une suite finie d'opérations élémentaires, à appliquer dans un ordre déterminé, à des données. Sa réalisation permet de résoudre un problème donné.

Exemples : suivre une recette de cuisine, suivre un plan, faire une division euclidienne à la main sont des exemples d'algorithme.

Remarques : Un algorithme doit être lisible de tous. Son intérêt, c'est d'être codé dans un langage informatique afin qu'une machine (ordinateur, calculatrice, etc.) puisse l'exécuter rapidement et efficacement.

Exemple : L'algorithme 1 permet de tracer un carré de côté 150 pixels.
Sa traduction en Python est donné dans l'algorithme 2.

Algorithme 1 Carré de côté 150 pixels

```
Effacer l'écran
Baisser le stylo
Avancer de 150 pixels
Tourner de 90 degrés vers la gauche
Avancer de 150 pixels
Tourner de 90 degrés vers la gauche
Avancer de 150 pixels
Tourner de 90 degrés vers la gauche
Avancer de 150 pixels
```

Algorithme 2 Programme Python permettant de tracer un carré de côté 150 pixels

```
from turtle import *
reset()
down()
forward(150)
left(90)
forward(150)
left(90)
forward(150)
left(90)
forward(150)
mainloop()
```

Questions flash : Exercices 30, 31, 32, 33, 34 page 30 [1] [Magnard]

1.2 Variable, affectation

Activité : Activité 1 page 14 [2] [Magnard]

Définitions :
- Lors de l'exécution d'un algorithme, on va avoir besoin de stocker des données, voire des résultats. Pour cela, on utilise des variables.
  On attribue un nom à chaque variable.
- Affecter une valeur à une variable, c'est remplacer le contenu de la variable par cette valeur (qui peut être le résultat d'un calcul).

1. Utilisation de la bibliothèque `turtle` sous Python.
2. Afficher et affecter des valeurs.

Remarques :

1. Une variable est comme une boîte, repérée par un nom, qui va contenir une information. Pour utiliser le contenu de cette boîte, il suffit de l'appeler par son nom.
2. Dans l'écriture d'un algorithme, l'affectation d'une variable est désigné par le symbole $\longleftarrow$.
   En Python, l'affectation d'une variable se fait grâce au signe $=$.

Exemple : L'algorithme 3 et le programme Python correspondant (Algorithme 4), permettent d'affecter des valeurs aux variables x et y.

Algorithme 3 Affectation de variable

$x \longleftarrow 5$
$y \longleftarrow 12$
$x \longleftarrow 3x + 2y$
$y \longleftarrow 5y - 12x$
`Afficher` x
`Afficher` y

Algorithme 4 Programme Python d'affectations de variables

```
x = 5
y =12
x = 3*x+2*y
y = 5*y-12*x
print("x=",x)
print("y=",y)
```

Si l'on veut comprendre ce que fait un algorithme sans le programmer, on suit « à la main » les instructions l'une après l'autre. Il peut alors être utile de mettre les différentes valeurs affectées à une variable dans un tableau :

x	5	$3 \times 5 + 2 \times 12 = 39$
y	12	$5 \times 12 - 12 \times 39 = -408$

Remarques :

1. En python, l'instruction `print(<nom de variable>)` permet d'afficher à l'écran la valeur d'une variable.
2. Il est possible de demander à l'utilisateur de l'algorithme de choisir la valeur d'une variable. On dit que cette variable est saisie. L'instruction Python permettant la saisie est `input()`.
   Elle s'utilise de la façon suivante :

$$\underbrace{\texttt{a}}_{\text{nom de la variable}} = \texttt{input}\left(\underbrace{\texttt{"entrer la valeur de a : "}}_{\text{texte à afficher à l'écran}}\right)$$

Exercices : 3, 4 page 24 [3] – 39, 40, 41 page 31 [4] [Magnard]

1.3 Différents types de variables

Activités : Activité 2 page 14 [5] et Activité 3 page 15 [6] [Magnard]

Définitions : Dans un algorithme ou un programme Python, les variables ont un type qui définit la nature des valeurs que cette variable peut contenir.
Les trois principaux types de variables sont :
— Les entiers, lorsque les valeurs prises par la variable ne sont que des nombres entiers (éventuellement relatifs) ;
— Les flottants, lorsque la valeurs prises par la variable sont des nombres réels ;
— Les chaînes de caractères, lorsque les valeurs prises par la variable sont des mots ou des phrases.

3. Affectations de variables.
4. Premiers algorithmes.
5. Comprendre les variables de type numérique.
6. Comprendre les variables de type textuel.

Remarques :

1. En Python :
   — le type entier est noté `int` (pour *integer*) ;
   — le type flottant est noté `float` (pour *floating-point*) ;
   — le type chaîne de caractères est noté `str` (pour *string*).
2. Les commandes `int()`, `float()`, `str()` permettent de changer le type d'une variable. Ceci peut être utile lors de la saisie d'une variable.
   Par exemple, la commande `a=float(input("saisir une valeur : ")` permet de s'assurer que la variable a sera bien un flottant, même si l'utilisateur a rentré une valeur entière.

Exercices : 1, 2 page 24 et 36 page 30 [7] [Magnard]

2 Quelques structures importantes

2.1 L'instruction conditionnelle

Activité : Activité 4 page 15 [8] [Magnard]

Définition : La résolution des certains problèmes nécessite la mise en place d'un test pour savoir si l'on doit effectuer une tâche.
Si la condition est remplie alors on effectue la (ou les) tâches, sinon on effectue (éventuellement) une autre (ou des autres) tâches.
Dans un algorithme, on code la structure du « Si... Alors.. Sinon » sous la forme suivante :

```
Si condition Alors
    Tâche 1
    Tâche 2
    ...
Sinon
    Tâche 1bis
    Tâche 2bis
    ...
Fin Si
```

Remarques : 1. Il est important de respecter les espaces laissés au début de chaque ligne, appelés indentations, car ils permettent de savoir quel bloc d'instructions fait partie du test..

2. Le « Sinon » n'est pas obligatoire. S'il n'est pas présent, aucune tâche ne sera effectué si la condition n'est pas remplie.
3. En Python, une instruction conditionnelle se code de la façon suivante :

```
if condition :
    Tâche 1
    Tâche 2
    ...
else :
    Tâche 1bis
    Tâche 2bis
    ...
```

L'indentation en début de ligne est obtenue grâce à la touche *Tabulation* du clavier. Il ne faut pas oublier les : après la condition du *if* et après le *else*.

Exemples : 1. L'algorithme 5 et le programme Python associé (algorithme 6) permet de déterminer si un triangle ABC est rectangle en C.

7. Type d'une variable.
8. Programmer les instructions conditionnelles.

Algorithme 5 Triangle rectangle en C

```
AB ⟵ valeur saisie
AC ⟵ valeur saisie
BC ⟵ valeur saisie
x ⟵ AB²
y ⟵ AC²+BC²
Si x = y Alors
   Afficher « Le triangle ABC est rectangle en C »
Sinon
   Afficher « Le triangle ABC n'est pas rectangle en C »
Fin Si
```

Algorithme 6 Programme Python :Triangle rectangle en C

```
from math import *
AB = float(input("Entrer la valeur de AB : "))
AC = float(input("Entrer la valeur de AC : "))
BC = float(input("Entrer la valeur de BC : "))
x = AB**2
y = AC**2+BC**2
if x == y :
   print("Le triangle ABC est rectangle en C")
else :
   print("Le triangle ABC n'est pas rectangle en C")
```

— L'instruction `from math import *` permet de charger la bibliothèque mathématique de Python, qui contient plus de fonctions mathématiques, notamment la mise au carré : `AB**2` signifie AB^2.
— En Python, le test d'égalité se fait en utilisant ==. Le signe = est réservé aux affectations de variables.

2. L'algorithme 7 et le programme Python de l'algorithme 8 simule un jeu de pile ou face avec une pièce non truquée. « Pile » est représenté par le nombre 0 et « Face » par le nombre 1.

Algorithme 7 Jeu de Pile ou Face

```
choix ⟵ valeur saisie
tirage ⟵ nombre au hasard choisit dans l'ensemble {0 ; 1}
Si choix = tirage Alors
   Afficher « Gagné ! »
Sinon
   Afficher « Perdu ! »
Fin Si
```

Algorithme 8 Programme Python : jeu de Pile ou face

```
from random import *
choix = input()
tirage = randint(0,1)
if choix == tirage :
   print("Gagne !")
else :
   print("Perdu !")
```

— L'instruction `from random import *` permet de charger la bibliothèque random de Python, qui permet des tirages aléatoires de nombres.
— L'instruction `randint(a,b)` tire un nombre entier au hasard entre a et b.

Exercice : Transformer cet algorithme pour simuler le jet d'un dé à 6 faces, puis le jet de deux dés à 6 faces dont on fait la somme.

3. L'algorithme 9 et le programme Python associé (algorithme 10) permettent de calculer l'image d'un réel x par la fonction $f : x \to \sqrt{x}$ en respectant son ensemble de définition.

Algorithme 9 Image par la fonction racine carrée

```
x ⟵valeur saisie
Si x ≥ 0 Alors
    y ← √x
    Afficher y
Sinon
    Afficher « La valeur choisie n'est pas dans l'ensemble de définition »
Fin Si
```

Algorithme 10 Programme Python : Image par la fonction racine carrée

```
from math import *
x = float(input("Entrer la valeur de x : "))
if x >= 0 :
    y = sqrt(x)
    print("f(",x,")",y)
else :
    print("La valeur choisie n'est pas dans l'ensemble de définition")
```

— La commande Python `sqrt()` signifie *squareroot* et calcule la racine carrée.

Exercices : 6 page 25 et 43, 44, 45, 46, 47 page 31 [9] – 7 page 25 et49, 50 page 31 [10][Magnard]

2.2 La boucle bornée

Activité : *TD n°2 d'initiation à l'algorithmique.*

Définition : Lorsque l'on doit répéter un nombre de fois *connu à l'avance* la même tâche, on utilise une boucle bornée de la forme « Pour.. allant de... à ».

Dans un algorithme, cette structure est codée de la façon suivante :

```
Pour variable allant de valeur_depart à valeur_fin faire
    tâche 1
    tâche 2
    ...
Fin pour
```

La variable utilisée dans la boucle est appelée compteur. À chaque passage dans la boucle, sa valeur est automatiquement augmentée de 1.

Remarque : En Python, une boucle bornée se code de la façon suivante :

```
for i in range(a,b) :
```

9. Comprendre une instruction conditionnelle.
10. Écrire une instruction conditionnelle.

```
tâche 1
tâche 2
...
```

Attention ! Dans ce cas, la valeur de fin du compteur i sera $b-1$, pas b. Sous python, la dernière boucle n'est pas exécutée.

Exemples :

1. L'algorithme 11 permet de tracer un carré de côté 150 pixels en utilisant un boucle bornée. Sa traduction en Python est donné dans l'algorithme 12.

Algorithme 11 Carré de côté 150 pixels, version 2

```
Effacer l'écran
Baisser le stylo
Pour i allant de 1 à 4 faire

   Avancer de 150 pixels
   Tourner de 90 degrés vers la gauche

FinPour
```

Algorithme 12 Programme Python permettant de tracer un carré de côté 150 pixels, version 2

```
from turtle import *
reset()
for i in range (1,3) :

   forward(150)
   left(90)

mainloop()
```

2. L'algorithme 13 (et le programme Python de l'algorithme 14) affiche la table de multiplication (de 0 à 10) d'un nombre entier donné.

Algorithme 13 Table de multiplication

```
n ⟵ valeur saisie
Pour i allant de 0 à 10 faire

   m ⟵ n × i
   Afficher n ,« x », i, « = », m

Fin Pour
```

Algorithme 14 Programme Python : table de multiplication

```
n = int(input())
for i in range (0,11) :

   m = n*i
   print(n," x ",i," = ",m)
```

3. L'algorithme 15 affiche la somme de tous les entiers jusqu'à un entier donné. Sa traduction en Python est donné dans l'algorithme 16.

Exercices : 10 page 26 et 51, 52 page 31 [11] – 12, 13 page 26 et 54, 55, 56 page 32 [12] [Magnard]

11. Comprendre un algorithme avec une boucle bornée.
12. Écrire un algorithme avec une boucle bornée.

Algorithme 15 Calcul d'une somme d'entiers

```
n ⟵ valeur saisie
S ⟵ 0
Pour i allant de 1 à n faire
    S reçoit S + i
Fin Pour
Afficher S
```

Algorithme 16 Programme Python : calcul d'une somme d'entiers

```
n = int(input())
S = 0
for i in range (0,n+1) :
    S = S + i
print(S)
```

2.3 Fonctions

Activité : *TD n°3 d'initiation à l'algorithmique.*

Définition : Pour diverses raisons (de lisibilité ou pour éviter la répétitions d'instructions par exemple), il peut être utile d'écrire un bloc d'instruction sous la forme d'une fonction. Ce bloc d'instruction ne sera exécuté que s'il est appelé par la suite, en utilisant le nom de la fonction.
Une fonction possède généralement des paramètres, et renvoie généralement une valeur de retour.
Dans un algorithme, cette structure est codée de la façon suivante :

```
Fonction nom(param1,param2,...)
    tâche 1
    tâche 2
    ...
    Retourner variable
```

Remarque : En Python, une fonction se code de la façon suivante :

```
function nom(param1,param2,...) :
    tâche 1
    tâche 2
    ...
    return variable
```

Exemples :

1. L'algorithme 17 permet de calculer l'aire d'un triangle, en utilisant une fonction.
   Sa traduction en Python est donné dans l'algorithme 18.

Algorithme 17 Aire d'un triangle

```
Fonction aire(base,hauteur)
    a ⟵ (base×hauteur)/2
    Retourner a
b ⟵valeur saisie
h ⟵valeur saisie
A ⟵ aire(b,h)
Afficher A
```

2. L'algorithme permet de tracer plusieurs carrés à la suite les uns des autres, en utilisant une fonction permettant de tracer un carré.
   Sa traduction en Python est donné dans l'algorithme 20.

Algorithme 18 Programme Python : aire d'un triangle

```
function aire(base,hauteur) :
    a = (base×hauteur)/2
    return a
b = float(input("Entrer la longueur de la base : ")
h = float(input("Entrer la longueur de la hauteur : ")
A = aire(b,h)
print("L'aire du triangle est ",A)
```

Algorithme 19 Suites de carré

```
Fonction carré(coté)
    baisser le stylo
    Pour i allant de 1 à 4 faire
        Avancer de coté pixels
        Tourner de 90 degrés vers la gauche
    FinPour
effacer l'écran
carré(150)
lever le stylo
avancer(150)
carré(200)
lever le stylo
avancer(200)
```

Algorithme 20 Programme Python : suite de carrés

```
from turtle import *
function carre(cote) :
    down()
    for i in range (1,3) :
        forward(cote)
        left(90)
reset()
carre(150)
up()
forward(150)
carre(200)
up()
forward(200)
mainloop()
```

Exercices : 22 page 29 et 63, 64 page 32 [13] – 25, 26 page 29 ; 66 page 32 et 70, 73, 75, 76, 77 page 33 [14] [Magnard]

2.4 La boucle non bornée

Activité : *TD n°4 d'initiation à l'algorithmique.*

Définition : Dans l'exécution d'un algorithme, on peut être amené à réaliser plusieurs fois la même tâche, sans savoir à priori combien de fois cette tâche doit être réalisée. On répète alors les instructions tant qu'une condition est remplie.

On utilise alors une boucle non bornée qui est codée de la façon suivante dans un algorithme :

```
Tant que condition  faire
    tâche 1
    tâche 2
    ...
Fin Tantque
```

Remarque : En Python, une fonction se code de la façon suivante :

```
while condition :
    tâche 1
    tâche 2
    ...
```

Exemple : L'algorithme 21 (et le programme Python donné à l'algorithme 22) affiche la plus petite puissance de 2 supérieure à 10 000.

Algorithme 21 Puissance de 2 supérieure à 10 000

```
puissance ⟵1
Tant que puissance ≤ 10000 faire
    puissance ⟵ puissance × 2
Fin Tantque
Afficher(puissance)
```

Algorithme 22 Programme Python : Puissance de 2 supérieure à 10 000

```
puissance = 1
while puissance <= 10000 :
    puissance = puissance*2
print(puissance)
```

Exercices : 15, 16, 17 page 27 et 59, 60 page 32 [15] – 19, 20, 21 page 28 et 62 page 32 [16] – 78,page 33 [17] [Magnard]

Exercices de synthèse : 81 page 34 [18] – 83 page 34 [19] – 85 page 34 [20] [Magnard]

13. Comprendre une fonction simple.
14. Utiliser des fonctions simples.
15. Comprendre un algorithme avec une boucle non bornée.
16. Écrire un algorithme avec une boucle non bornée.
17. Fonctions et boucles non bornées.
18. Monte-Carlo.
19. Moyennes.
20. Suite de Syracuse.

Références

[Magnard] Maths 2de, Magnard, 2019

2, 3, 4, 6, 7, 10