Algorithmie

Renaud Costadoat
Lycée Dorian

DORIAN

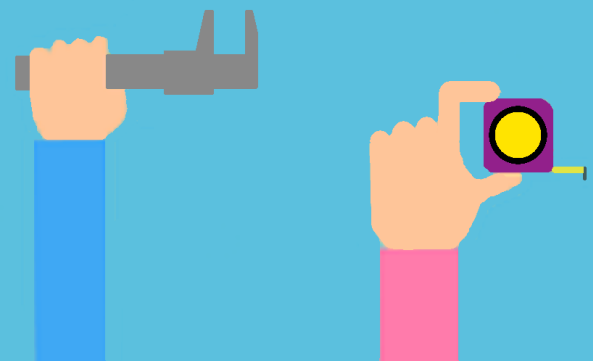

Table des matières

Algorithme

Definition

Un **algorithme** est une procédure permettant de résoudre un problème, écrite de façon suffisamment détaillée pour pouvoir être suivie sans posséder de compétence particulière ni même comprendre le problème que l'on est en train de résoudre.

Definition

Un **programme** est la traduction d'un algorithme dans un langage particulier, à la fois interprétable par la machine et compréhensible par l'homme. Il est constitué d'un assemblage d'instructions, regroupées dans un fichier texte appelé le code source du programme.

Remarque

En toutes circonstances, il faudra s'assurer que :

- le résultat de l'algorithme est conforme au résultat attendu ;
- l'algorithme soit documenté.

Exemple des algorithmes

Objectif: Déterminer la valeur de π par dichotomie.

On sait que :

- $sin(x) > 0$ pour $x < pi$,
- $sin(x) <= 0$ pour $x >= pi$.

Ainsi un algorithme permet de calculer Pi:

1. Soient $a = 3$, $b = 4$,
2. Soit m le milieu de a et b,
3. Si $sin(m) > 0$, $a = m$ sinon $b = m$,
4. Recommencer à partir de l'étape 2, tant que la précision n'est pas celle souhaitée.

Sémantique

Opérateurs

Chaque type de donnée admet un jeu d'opérateurs adaptés :

- opérateurs arithmétiques : $+$, $-$, $*$, $/$, *div*, *mod*, ?, $**$,
- opérateurs relationnels : $==$, $\neq$, $>$, $<$, $\leq$,
- opérateurs logiques : *négation*, *ou*, *et*.

Instruction

Il existe aussi des opérateurs d'instruction comme l'opérateur d'affectation : $\leftarrow$.

Definition

Une **expression** est un groupe de nombres, constantes, variables liées par des opérateurs. Elles sont évaluées pour donner un résultat.

Une séquence d'**instructions** est appelé bloc d'instructions. Lors de l'exécution du programme, les instructions s'exécutent les unes après les autres.

Table des matières

Les fonctions

Definition

Les fonctions permettent :

- d'automatiser des tâches répétitives,
- d'ajouter de la clarté à un algorithme,
- d'utiliser des portions de code dans un autre algorithme.

Une analogie avec les mathématiques reviendrait à dire que: $f : x \rightarrow y$ s'écrit

```
def f(x):
    return y
```

Lors de l'exécution d'un programme, il faut parfois répéter un grand nombre de fois une instruction. Ainsi, il peut être pratique de créer une fonction qui permet de limiter l'écriture.

```
>>>def moyenne(a,b):
    m=(b+a)/2.
    return m
>>>a=(moyenne(40,2))
>>>print a
   21.0
```

Variables locales/globales

Definition

Visibilité : Une variable globale est définie en dehors de toute fonction, une variable locale est définie dans une fonction et masque toute autre variable du même nom.

Durée de vie : Une variable globale existe durant l'exécution du programme, une variable locale existe durant l'exécution de la fonction.

Par défaut, dans un langage interprété, les variables sont locales à un bloc.

```
a=4
def f(x):
    a=3
    b=2
    res = a*x+b
    return res
y=f(a)
```

```
>>> print a
4
>>> print y
14
>>> print b
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'b' is not defined
```

Documentation des fonctions

Il faut absolument commenter les fonctions mises en place dans un programme afin de pouvoir revenir sur un code plus tard.

```
def f(x):
    # x est l'abscisse de la droite déterminée par la droite f
    # la réponse à la fonction est y=a*x+b
    a=3
    # a est le coefficient directeur de la droite
    b=2
    # b est l'ordonnée à l'origine de la droite
    res = a*x+b
    return res
```

Import de méthodes et de fonctions

Par défaut, Python ne permet que de réaliser des opérations élémentaires (opérations mathématiques élémentaires, comparaisons, boucles etc.).

Il existe par ailleurs un grand nombre de bibliothèques permettant d'utiliser beaucoup plus de fonctions permettant de faciliter votre travail.

```
import math # Import de toutes les methodes de la bibliotheque math
math.sqrt(2) # Permet d'utiliser la methode sqrt de math
from math import sqrt # Import de la methode sqrt de math
import os # Import de la bibliotheque os permettant
            de realiser des operations systemes
```

Exemples d'algorithmes

Dans l'exemple de l'algorithme du calcul de π, il est nécessaire d'arrêter le calcul au bout d'un certain temps.

Exemple de boucle **tant que**, ou **while**.

```
while math.sin(m) > 10**-10 or math.sin(m) < -10**-10:
   # tant que le sinus est plus petit que +- 10^-10
```

Exemple d'instruction fonctionnelle **si**, ou **if**.

```
if math.sin(m) > 0:
   # action à effectuer si sin(m)>0
else:
   # action à effectuer si sin(m)<0
```

Exemples d'algorithmes

Exemple de boucle d'instructions itératives, ou **for**.

```
a,n=1,8
for i in range(1,n+1):
   a=a*i # réaliser une action n fois
```

Il est parfois nécessaire de mettre en place des systèmes de gestion des erreurs qui permettent de contrôler les données d'entrée des fonctions.

```
p=math.pi
if (p > a and p < b) or (p < a and p > b):
    print "Faire le calcul"
else:
    print "Impossible de trouver pi entre %s et %s" % (a,b)
```