Cours

1 Variable et affectation

Définition Une **variable** est un *casier* de mémoire qui contient une information d'un certain **type**. Elle est désignée par un **nom**.

Type dans les algorithmes	Type en langage Python
• Nombre entier • Nombre réel • Booléen • Chaîne de caractères • Liste	• int : nombre entier • float : nombre réel • boolean : booléen • str : chaîne de caractères (*string* en anglais) • list : liste d'éléments

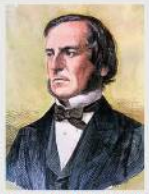
Remarque

Un booléen est une variable ne pouvant prendre que deux états notés Vrai (True ou 1) et Faux (False ou 0).

Définition L'**affectation** d'une variable est l'instruction permettant de « remplir » le *casier* correspondant.

Algorithmes	Langage Python
$X \leftarrow 5$	$X = 5$

Info



George Boole (1815-1864) est le créateur de l'algèbre binaire manipulant des éléments ne pouvant prendre que deux valeurs : 0 ou 1.

Remarque

En Python, la fonction input permet à l'utilisateur d'affecter une valeur à une variable pendant l'exécution du programme.

2 Structure conditionnelle

a Instruction conditionnelle *Si… Alors… Sinon*

Dans un algorithme, il peut être nécessaire d'exécuter un bloc d'instructions uniquement lorsque certaines conditions sont satisfaites.

Syntaxe

```
Si condition 1 Alors
  | bloc 1 d'instructions à exécuter si condition 1 vraie
Sinon Si condition 2 Alors
  | bloc 2 d'instructions à exécuter si condition 2 vraie
Sinon
  | bloc 3 d'instructions à exécuter dans les autres cas
Fin Si
```

Exemple

L'algorithme suivant indique à l'utilisateur s'il est majeur ou mineur, *âge* étant un entier saisi par l'utilisateur.

```
Si âge ⩾ 18 Alors
  | Afficher « Vous êtes majeur. »
Sinon
  | Afficher « Vous êtes mineur. »
Fin Si
```

Remarques

- Un seul bloc d'instructions est exécuté, les autres sont ignorés.
- Les blocs *Sinon* et *Sinon Si* sont facultatifs.

b Syntaxe en langage calculatrice et Python

Casio

```
"AGE"?→A
If A≥18
Then
"vous etes majeur"
Else
"vous etes mineur"
IfEnd
```

TI

```
:Input "AGE?",A
:If A≥18
:Then
:Disp "VOUS ETES
 MAJEUR"
:Else
:Disp "VOUS ETES
 MINEUR"
:End
```

Python

```
1 age=int(input("Quel est votre âge ?"))
2
3 if age>=18:
4     print("Vous êtes majeur.")
5 else:
6     print("Vous êtes mineur.")
```

- Les retours à la ligne sont obligatoires après les mots-clés **Then** et **Else**.
- Pour conclure la structure conditionnelle, on utilise la commande **IfEnd** (Casio) ou **End** (TI).

- Les double-points : marquent le début des blocs d'instructions. Ils sont **indentés** (le texte est décalé). C'est cette indentation qui indique le début et la fin de chaque bloc et en particulier, la fin de la structure conditionnelle.

3 Fonctions

La notion de fonction est très importante en informatique, elle permet de structurer un programme en le décomposant en sous-programmes.

a Qu'est-ce qu'une fonction ?

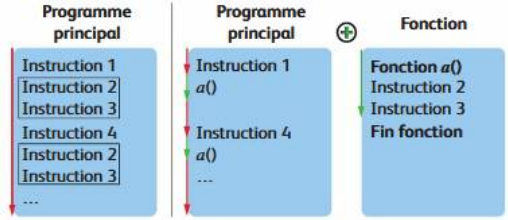
Définition Une **fonction** est une succession d'instructions regroupées dans un bloc que l'on désigne par un nom.

Intérêt des fonctions

- **Lisibilité** : elles rendent le programme principal plus lisible.
- **Répétabilité** : elles permettent d'éviter la redondance du programme. Il est assez courant d'avoir à répéter des blocs d'instructions, on les regroupe alors dans une fonction.
- **Flexibilité** : il est plus rapide de modifier une fonction que de modifier l'ensemble des parties du programme où elle est utilisée.

Remarque

Le schéma ci-dessous illustre la différence entre l'exécution d'un programme sans fonction (à gauche) et d'un programme avec fonction (à droite), où les instructions 2 et 3 sont regroupées dans une fonction *a*.



Exemple

On considère l'algorithme suivant calculant plusieurs volumes de pyramides à base carrée.
Soient c, h et v des variables réelles.

Sans fonction

```
c ← 2
h ← 5
v ← (1/3)*c**2*h
Afficher(v)
c ← 0.5
h ← 10
v ← (1/3)*c**2*h
Afficher(v)
```

Avec fonction

Fonction volume qui prend en paramètre les variables c et h.

```
Fonction volume(c, h)
  | return (1/3)*c**2*h
Fin Fonction
```

Programme principal

```
Afficher(volume(2,5))
Afficher(volume(0.5,10))
```

b Construire une fonction en Python

Le langage Python permet aux programmeurs d'enrichir les bibliothèques de fonctions prédéfinies en créant leurs propres fonctions.

Structure générale d'une fonction en langage Python

```
def nomFonction(paramètres):
    #bloc d'instructions
    ...
    return ...
```

`def` est le mot clé permettant la construction d'une fonction.

Les paramètres (ou *arguments*) sont des variables nécessaires à sa bonne exécution. Certaines fonctions n'admettent aucun paramètre.

`return` est le mot clé qui permet à la fonction de renvoyer un contenu. Il n'est pas obligatoire.

Remarques

- On choisit un nom de fonction en rapport avec ce qu'elle fait pour rendre le programme principal plus intelligible.
- Sous Python, on retrouve le double-point « : » et l'indentation, indiquant le début et la fin de la fonction.
- De nombreuses bibliothèques de fonctions sont disponibles. Il faut au préalable les importer avec le mot clé import.

Exemple

```
from math import sqrt # Importe la fonction racine carrée.
from random import * # Importe toutes les fonctions de random.
import pylab as pl # Définit un raccourci pour utiliser pylab.
```

Exemple

L'algorithme ci-dessus en Pythor devient :

```
def volume(c , h):
    return (1/3)*c**2*h

print(volume(2,5))
print(volume(0.5,10))
```

Info Python

« c**2 » signifie c^2 en langage Python.

Info Python

La commande help(fonction) permet de donner une documentation sur la fonction saisie.

4 Structures itératives : boucles

Les structures d'instructions itératives permettent de répéter un nombre fini de fois un même bloc d'instructions, on parle de **boucles**.

a Boucle bornée (Boucle *Pour* ou *for*)

Lorsque le nombre de répétitions est connu à l'avance, on parle de boucle *bornée* ou encore de boucle *Pour* (*for*).

Syntaxe

Pour *i* allant de n_1 à n_2 **Faire**
| *bloc d'instructions*
Fin Pour

Le **compteur** *i* prend successivement toutes les valeurs entières de n_1 à n_2 : il est automatiquement **incrémenté** d'une unité à chaque passage dans la boucle.

Pour chaque valeur prise par *i*, le bloc d'instructions est exécuté.

Exemple

On dispose initialement de 100 € d'épargne. On verse 10 € le premier mois et ensuite chaque mois 10 € de plus que le mois précédent. On calcule l'épargne disponible au bout de 24 mois.

```
epargne ← 100
Pour i allant de 1 à 24 Faire
    versement ← 10×i
    epargne ← epargne+versement
Fin Pour
```

Syntaxe en langages calculatrice et Python

Casio

```
100→E
For I→I To 24
10I→V
E+V→E
Next
E
```

TI

```
:100→E
:For(I,1,24)
:10I→V
:E+V→E
:End
:Disp E
```

Python

```
epargne=100
for i in range(1,25):
    versement=10*i
    epargne=epargne+versement

print(epargne)
```

Sur Casio, la boucle *For* se termine par l'instruction *Next*. On remarquera la syntaxe particulière pour la définition et l'initialisation du compteur.

`range(1,N+1)` est la liste des entiers consécutifs de 1 à N inclus. Ainsi, `range(1,25)` est la liste des entiers de 1 à 24 inclus.

b Boucle non bornée (Boucle *Tant que* ou *while*)

Lorsque le nombre de répétitions est *a priori* inconnu, mais que l'on connaît un test d'arrêt, on parle de boucle *non bornée* ou encore de boucle *Tant que* (*while*).

Exemple

On reprend l'exemple précédent et on souhaite déterminer à partir de quel mois l'épargne atteindra ou dépassera 1 000 €.

```
epargne ← 100
i ← 0
Tant que epargne<1 000
    i ← i+1
    versement ← 10×i
    epargne ← epargne+versement
Fin Tant que
```

Syntaxe

Tant que *condition* **Faire**
| *bloc d'instructions*
Fin Tant que

Si la *condition* est *vraie*, le bloc d'instructions est exécuté puis la *condition* est réévaluée. Si elle est *fausse*, on sort de la boucle.

Syntaxe en langages calculatrices et Python

Casio

```
100→E
0→I
While E<1000
I+1→I
10I→V
E+V→E
WhileEnd
I
```

TI

```
:100→E
:0→I
:While E<1000
:I+1→I
:10I→V
:E+V→E
:End
:Disp I
```

Python

```
epargne=100
i=0       #compteur initialisé
while epargne<1000:
    i=i+1 #compteur incrémenté
    versement=10*i
    epargne=epargne+versement

print(i)
```

Il faut ici définir et initialiser le compteur *i*. On incrémente ce compteur à chaque passage dans la boucle.

Consoles PYTHON « en ligne »

- https://www.lelivrescolaire.fr/console-python
- https://repl.it/
- http://www.brython.info/tests/editor.html?lang=fr
- http://www.pythontutor.com/visualize.html#mode=edit
- https://robertvandeneynde.be/parascolaire/turtle_test.html

Quelques liens utiles

- https://www.youtube.com/watch?v=pE11tsQo4dA
- https://www.python.org/getit/
- https://vimeo.com/31223457
- https://www.numworks.com/fr/ressources/python/activites/
- https://edupython.tuxfamily.org/
- http://matharavel.tuxfamily.org/TS.spe.2018/peton.pdf
- http://pise.info/algo/enonces4.htm
- https://info.blaisepascal.fr/exercices-python
- http://maths.enseigne.ac-lyon.fr/spip/squelettes/docs_lyon/supports_formations/2017_Python_2nde/formation_upo_python_2nde/accueil_python_2nde.html