Variables

Types de variables

Dans un algorithme ou un programme, les variables considérées ont des types qui définissent la nature des valeurs qu'elles peuvent prendre. Les trois types principaux considérés en classe de Seconde sont :
- les entiers quand les valeurs possibles de la variable sont des nombres entiers.
- les flottants quand les valeurs possibles de la variable sont des nombres réels.
- les chaînes de caractères quand les valeurs possibles de la variable sont des mots.

Exemple

On doit écrire un programme effectuant des statistiques sur des équipes sportives. Dans ce programme, il y aura en particulier trois variables: nom qui correspond au nom de l'équipe, effectif qui correspond à son effectif et moyenne_age qui correspond à sa moyenne d'âge. Les valeurs possibles prises par :
- nom sont des mots: elle est donc de type chaîne de caractères.
- effectif sont des nombres entiers: elle est donc de type entier.
- moyenne_age sont des nombres réels, non entier d'une manière générale : elle est donc de type flottant.

Remarque

Python est un langage informatique dans lequel :
- le type chaîne de caractères se nomme str (pour string),
- le type entier se nomme int (pour integer),
- le type flottant se nomme float (pour floating-point).

Affectation d'une valeur à une variable

Lorsque l'on a affecté une valeur à une variable, on peut remplacer la variable par cette valeur dans les instructions qui suivent (par exemple dans les opérations arithmétiques).
On peut visualiser une boîte portant le nom de la variable dans laquelle on stocke une seule valeur.

Exemple

On considère la suite d'instructions:

En langage naturel	**En python**
a←2	a=2
a←a+1	a=a+1

Les différentes étapes correspondantes sont les suivantes:

Étape 1

Ligne a=2

« a reçoit la valeur 2 » ou « 2 est affecté à a » donc on place la valeur 2 dans la boîte a.

Étape 2

Ligne a=a+1

Le calcul a + 1 est effectué : son résultat est 3 puisque a vaut 2.

Étape 3

Ligne a=a+1

a reçoit la valeur 3 calculée précédemment donc on remplace 2 par 3 dans la boîte a.

Remarque

Plutôt que d'affecter une valeur à une variable, on peut laisser l'utilisateur affecter la valeur de son choix à la variable.

L'algorithme suivant demande la saisie d'un nombre par l'utilisateur puis calcule et affiche le double de ce nombre.

En langage naturel	**En python**
a←Valeur saisie	a=float(input("Saisir une valeur"))
Afficher 2×a	print(2*a)

Instructions conditionnelles

Définition

Dans un algorithme, on est parfois amené à exécuter une ou plusieurs instructions uniquement si une certaine condition est vérifiée, c'est ce que l'on appelle des instructions conditionnelles. Si la condition n'est pas vérifiée, on peut soit exécuter un autre bloc d'instructions, soit ne rien faire. Dans ces deux cas, on exécute ensuite la suite de l'algorithme.

Exemple

L'algorithme suivant permet de simuler un jeu dans lequel on lance un dé et où l'on gagne uniquement si le résultat est 6.

En langage naturel

```
x←entier aléatoire entre 1 et 6
Si x=6
   Afficher "Gagné !"
Sinon
   Afficher "Perdu…"
Fin si
Afficher "À bientôt !"
```

En python

```
import random
x=random.randint(1,6)
if x==6:
    print("Gagné !")
else:
    print("Perdu…")
print("À bientôt !")
```

- Si la valeur de la variable x est 6 alors l'instruction Afficher "Gagné !" est exécutée. Sinon, c'est-à-dire si la valeur de la variable x n'est pas 6, alors l'instruction Afficher "Perdu…" est exécutée.
- Dans les deux cas, l'algorithme se poursuit après Fin si et la dernière instruction (qui n'est pas conditionnelle) Afficher "À bientôt !" est exécutée.

Remarque

- La condition qui s'écrit x = 6 en langage naturel s'écrit x == 6 en Python. En Python, le signe = seul est réservé à l'affectation.
- La condition Sinon est facultative.

Booléen

Il existe un type de variables appelé booléen dont les valeurs ne peuvent être que « Vrai » ou « Faux ». Dans une instruction conditionnelle, le résultat d'une condition est un booléen.

Exemple

Dans l'algorithme en langage naturel précédent, si x vaut 5 alors la condition x = 6 renvoie Faux (x == 6 renvoie False en Python) et si x vaut 6 alors la condition x = 6 renvoie Vrai (x == 6 renvoie True en Python)

Fonctions

Définition

Pour diverses raisons (de lisibilité ou pour éviter des répétitions d'instructions, par exemple), il peut être utile de définir une fonction c'est-à-dire un bloc d'instructions qui ne sera exécuté que s'il est appelé (éventuellement plusieurs fois). Une fonction possède généralement des paramètres et retourne une valeur de retour (mais pas systématiquement, par exemple si elle réalise un affichage)

Exemple

L'indice de masse corporelle (IMC) d'une personne est donné par la formule IMC = $\frac{masse}{taille^2}$ où la masse est en kilogrammes et la taille en mètres. On considère l'algorithme ci-dessous dont on a numéroté les lignes.

En langage naturel

```
1 fonction calculIMC(masse, taille)
2     IMC←masse/taille²
3     Retourner IMC
4
5 IMCjean←calculIMC(60,1.6)
6 massepaul ← 85
7 taillepaul ← 1.80
8 Afficher calculIMC(massepaul, taillepaul)
```

En python

```
def calculIMC(masse, taille):
    IMC=masse/(taille*taille)
    return IMC

IMCjean=calculIMC(60,1.6)
massepaul=85
taillepaul=1.8
print(calculIMC(massepaul, taillepaul))
```

Cet algorithme est constitué :
- d'une fonction appelée calculIMC (des lignes 1 à 3) dont les paramètres sont les variables masse et taille (ligne 1) et dont la valeur de retour est la valeur de IMC (ligne 3) ;
- d'un algorithme principal à partir de la ligne 5.

Dans la ligne 5, IMCjean ← calculIMC(60,1.6) :
- l'algorithme principal appelle la fonction calculIMC en spécifiant que la variable masse (de la fonction) doit prendre la valeur 60 et que la variable taille (de la fonction) doit prendre la valeur 1.6.
- le bloc d'instructions correspondant à la fonction calculIMC est alors exécuté : la variable IMC (de la fonction) reçoit la valeur 60 / $1{,}6^2$ = 23,4375 (ligne 2) et la retourne (ligne 3) ;
- cette valeur 23,4375, de la variable IMC, est retournée dans l'algorithme principal à l'endroit de l'appel de la fonction (ligne 5), c'est-à-dire que l'algorithme principal affecte la valeur 23,4375 à la variable IMCjean (de l'algorithme principal).

De la même manière, à la ligne 8, l'algorithme principal affiche la valeur retournée par la fonction c'est-à-dire 85 / $1{,}8^2$ soit approximativement 26,23

Remarques

- Dans le programme python précédent, taille * taille peut être remplacé par taille**2. D'une manière générale en python, x**n signifie x^n
- Si une fonction python n'a pas de paramètre, on écrit tout de même les parenthèses vides dans sa définition et lors des rappels.

Boucles bornées

Définition

Lorsqu'on veut exécuter un nombre déterminé de fois un même bloc d'instructions, on utilise une boucle bornée, aussi appelée boucle Pour. Ces boucles sont munies d'une variable compteur que l'on peut utiliser dans les instructions.

Exemple

L'algorithme suivant permet de calculer et afficher les 100 premiers nombres pairs strictement positifs puis, quand cela est fait, affiche Terminé.

En langage naturel

```
Pour i allant de 1 à 100
   a←2×i
   Afficher a
Fin pour
Afficher "Terminé"
```

En python

```
for i in range(1,101):
    a=2*i
    print(a)
print("Terminé")
```

Au premier passage dans la boucle, i = 1 donc a = 2, puis i = 2 donc a = 4,…, puis i = 100 donc a = 200. Quand ces instructions ont été exécutées, la boucle Pour est terminée, on exécute alors la suite de l'algorithme c'est-à-dire l'affichage de Terminé.

Remarque

Dans la boucle précédente, la variable i est le compteur. On dit qu'à chaque passage dans la boucle, i est incrémenté de 1 c'est-à-dire augmente de 1.

Boucles non bornées

Définition

Lorsqu'on veut répéter un même bloc d'instructions tant qu'une certaine condition est vérifiée, on utilise une boucle non bornée, aussi appelée boucle Tant que.

Exemple

L'algorithme ci-contre affiche la plus petite puissance de 2 supérieure strictement à 1 000 000.

En langage naturel

```
p ← 1
Tant que p ⩽ 1 000 000
    p ← p×2
Fin tant que
Afficher p
```

En python

```
p=1
while p<=1000000
    p=p*2
print(p)
```

En effet, l'évolution des valeurs de la variable p est la suivante

p	**Condition p ⩽ 1 000 000**
1	Vérifiée
1 x 2 = 2	Vérifiée
2 x 2 = 4	Vérifiée
4 x 2 = 8	Vérifiée
…	…
524288	Vérifiée
1048576	Non vérifiée

Lorsque p = 1048576 on sort de la boucle Tant que et on exécute la suite de l'algorithme c'est-à-dire l'affichage de la valeur de la variable p, soit : 1048576.

Remarque

Comme la condition de la boucle Tant que de l'exemple précédent porte sur la variable p, cette variable doit être initialisée préalablement (c'est-à-dire qu'il faut lui donner une valeur au départ) : ici, on l'a initialisée à $1 = 2^0$, la première puissance de 2.