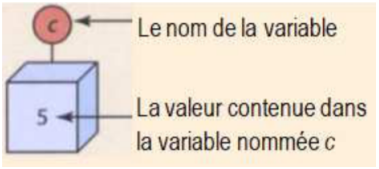
1 Variables et affectations

Définition 1.1 (Variables) Une *variable* est désignée par un nom et fait référence à un emplacement dans la mémoire de l'ordinateur. Elle contient une *valeur* qui peut évoluer au cours de l'exécution de l'algorithme.



Autrement dit, une variable informatique peut-être vue comme une boîte avec un nom (une étiquette) qui peut contenir différentes valeurs. Au fur et à mesure de l'exécution d'un algorithme, le contenu de la variable est susceptible d'être modifié.

Définition 1.2 (Affectation) Lorsque l'on demande de stocker une valeur dans une variable, on dit que l'on procède à une *affectation*.

Exemple 1.3 *A prend la valeur 7* s'écrit en algorithmique $A \leftarrow 7$ et se traduit en `Python` par `A = 7`.

Une instruction d'affectation comme $y \leftarrow x + 5$ est exécutée de la façon suivante :

- évaluer l'expression $x + 5$ en ajoutant 5 à la valeur contenue dans la variable x,
- mettre le résultat dans la variable y.

Dans une affectation, le membre de gauche reçoit la valeur du membre de droite.

Pour affecter une valeur à une variable, on utilise le symbole =.

- Le nom d'une variable est une succession de caractères alphanumériques ne contenant aucun espace, aucun caractère spécial sauf l'underscore _ et ne commençant jamais par un nombre (exemple : `2im` n'est pas un nom valide, mais `im2` est correct). Il peut contenir des majuscules ou des minuscules. Idéalement le nom d'une variable doit être court et explicite. Attention cependant aux mots clés que l'on ne peut pas utiliser comme nom de variable (par exemple : `int` ne peut pas être un nom de variable). De plus il vaut mieux éviter d'utiliser de caractères accentués.
- La variable affectée prend le type de la valeur que l'on affecte. Par exemple, si l'on affecte `3` à `x`, la variable `x` est de type `int`.

On distingue plusieurs méthodes d'affectation : les affectations simples, les affectations augmentées et les affectations multiples.

1.1 Différentes méthodes d'affectation

1.1.1 Les affectations simples

Pour pouvoir affecter une valeur à une variable, on utilise la syntaxe suivante :

```
nom_variable = valeur
```

Exemple 1.4 L'instruction `a = 3.5` affecte à la variable `a` la valeur `3.5`. Elle est alors du type `float`, c'est un nombre flottant. L'instruction `a = "abc"` affecte à la variable `a` la valeur `"abc"`. Elle est alors du type `str`, c'est une chaîne de caractères.

1.1.2 Les affectations augmentées

En `Python` on peut modifier la valeur d'une variable en procédant à une affectation augmentée. Il s'agit simplement d'un raccourci d'écriture, par exemple l'instruction `a = a + 5` permettant d'ajouter 5 à la valeur contenue dans la variable `a` peut s'écrire plus simplement `a += 5`. Attention à ne pas mettre d'espace entre l'opérateur et le symbole d'affectation.

Exemple 1.5

```
>>> a = 5
>>> a *= 3
>>> a
15
```

```
>>> a = 3.1
>>> a -= 1.5
>>> a
1.6
```

```
>>> a = 15
>>> a /= 2
>>> a
7.5
```

```
>>> a = "abc"
>>> a += "d"
>>> a
"abcd"
```

Remarque 1.6 Les opérateurs + et $\star$ n'agissent pas de la même façon selon le type de données qu'ils manipulent. Par exemple, quand il s'agit d'une chaîne de caractère comme dans l'exemple ci-dessus, l'opérateur + concatène les données. Il faut donc avoir conscience du type de données que l'on manipule quand on programme (voir partie 6).

1.1.3 Les affectations multiples

On peut également procéder à une affectation simultanée de plusieurs variables. Pour se faire, on utilise la syntaxe suivante :

```
nom_variable_1,nom_variable_2,...,nom_variable_n = liste de n valeurs
```

Les n valeurs sont affectées de gauche à droite aux variables déclarées à gauche de l'égalité. La liste de ces n valeurs peut être écrite sous l'une des formes suivantes :

- ⋆ directement et séparées par des virgules : `val_1,val_2,...,val_n`,
- ⋆ avec un tuple : `(val_1,val_2,...,val_n)`,
- ⋆ avec une liste : `[val_1,val_2,...,val_n]`.

Exemple 1.7 Considérons les instructions suivantes :

```
a,b = 1,"essai" # affecte à a la valeur 1 et à b la valeur "essai"
a,b,c = (1,3,-5) # affecte à a la valeur 1, à b la valeur 3 et à c la valeur -5
a,b = ["essai",True] # affecte à a la valeur "essai" et à b la valeur True
```

L'affectation simultanée permet également d'échanger des valeurs facilement. Ainsi, la syntaxe

```
a,b = b,a
```

permet d'échanger les valeurs des variables `a` et `b`.

1.2 Exercices

Exercice 1.8 On considère le programme `Python` suivant :

```
a = 7
b = 8
c = a + b
a = c + 1
b = a - 3
```

Compléter le tableau ci-dessous en indiquant, pour chaque ligne du programme, le contenu de chacune des trois variables `a`, `b` et `c` :

Contenu des variables	a	b	c
Ligne 1			
Ligne 2			
Ligne 3			
Ligne 4			
Ligne 5			

Une fois le tableau complété, consulter la solution via l'application en ligne Python Tutor en cliquant ici.

Exercice 1.9 Compléter le Notebook *NSI Première Partie 1 Chapitre 2 Variables et affectation*.

Exercice 1.10 (QCM)

1. On saisit l'instruction `a = 3`
   Parmi les affirmations suivantes, lesquelles sont vraies ?
   (a) `a` est une variable contenant la valeur `3`
   (b) c'est une erreur : `a` n'existe pas
   (c) on affecte la valeur `3` à la variable `a`
   (d) on teste si le contenu de `a` est bien égal à `3`

2. On saisit les instructions suivantes :

```
a = 3
b = a + 1
a = -2
```

   Parmi les affirmations suivantes, lesquelles sont vraies à l'issue de ces instructions ?
   (a) la variable `b` contient la valeur `-1`
   (b) la variable `b` contient la valeur `4`
   (c) la variable `a` contient la valeur `3`
   (d) la variable `a` contient la valeur `-2`

3. On saisit les instructions suivantes :

```
a = 4
b = a + 2
a = a + 1
del(a)
```

   Parmi les affirmations suivantes, lesquelles sont vraies à l'issue de ces instructions ?
   (a) la variable `b` contient la valeur `5`
   (b) il y a une erreur : `a` n'existe pas
   (c) la variable `a` n'existe pas
   (d) la variable `b` n'existe pas

4. On saisit les instructions suivantes :

```
a = 5
b = 2
a = a + b
b = a - b
a = a - b
```

   Parmi les affirmations suivantes, lesquelles sont vraies à l'issue de ces instructions ?
   (a) les valeurs de `a` et `b` sont échangées
   (b) `b` contient la valeur `5`
   (c) `a` contient la valeur `2`
   (d) les valeurs de `a` et `b` sont inchangées

5. On saisit les instructions suivantes :

```
a = 4
b = a + 2
del(a)
a = a + 1
```

   Parmi les affirmations suivantes, lesquelles sont vraies à l'issue de ces instructions ?
   (a) il y a une erreur
   (b) la variable `b` n'existe pas
   (c) la variable `b` contient la valeur `6`
   (d) la variable `a` contient la valeur `5`

2 Les types de données

Les informations étant binaires (une suite de 0 et de 1) dans un ordinateur, il faut que celui-ci soit capable de savoir s'il est en train de manipuler, par exemple, des nombres ou du texte. Pour ce faire, les données manipulées par l'ordinateur sont **typées**, c'est-à-dire qu'elles doivent indiquer à l'ordinateur comment il doit les interpréter (en tant que nombre, en tant que texte, etc.) afin que celui-ci utilise le bon « décodeur ».
En Python, contrairement à d'autres langages de programmation, il n'y a pas de déclaration de type à faire par l'utilisateur : celui-ci est attribué automatiquement au moment de la création de la donnée.

Exemple 2.1

- `A = 17` : A est du type `int` pour *integer* c'est-à-dire un nombre entier ;
- `A = 17.0` : A est du type `float` ou nombre flottant c'est-à-dire un nombre décimal ;
- `A = 'bonjour'` : A est du type `str` pour *string* c'est-à-dire une chaîne de caractères ;
- `A = True` : A est du type `bool` c'est-à-dire une donnée booléenne (soit vrai, soit faux) ;
- `A = [1, 2.5, 7, 'a']` : A est du type `list` c'est-à-dire une liste ;
- `A = (1, "abc", True)` : A est du type `tuple` c'est-à-dire un tuple (ou n-uplet).

Les types que nous utiliserons le plus souvent sont les suivants :

nombre entier	nombre flottant	chaîne de caractères	booléen	vide	range	liste	tuple
`int`	`float`	`str`	`bool`	`NoneType`	`range`	`list`	`tuple`

Remarque 2.2 Pour les nombres à « virgule », on utilise le point `.` pour indiquer la virgule. On écrit donc `6.4` et non pas `6,4`.

Pour obtenir le type d'une donnée ou d'une variable, on utilise la fonction `type()`.

Exemple 2.3 L'instruction `type(5)` renvoie `<class 'int'>`, l'instruction `type(a)` quand la variable `a` contient la valeur `4.5` renvoie `<class 'float'>`.

2.1 Le transtypage

Pour pouvoir modifier certains types de données ou pouvoir appliquer certaines fonctions ou méthodes, il est parfois nécessaire de passer d'un type à un autre. On procède alors à un *transtypage*. Pour ce faire, il suffit d'appliquer à la donnée que l'on souhaite transtyper la fonction donnant le nouveau type.

Exemple 2.4

- Si `x` est un flottant, l'instruction `int(x)` transtype `x` en un entier (en le tronquant à l'unité). Par exemple, `int(7.2)` renvoie `7`.
- Si `n` est un entier, l'instruction `float(n)` transtype `n` en un flottant. Par exemple, `float(5)` renvoie `5.0`.

Certains transtypages sont toujours valides. Par exemple, l'instruction `str(x)` renvoie toujours une chaîne de caractères, quelque soit le type de la donnée `x`. En revanche, certains transtypages peuvent ne pas fonctionner suivant le type de données utilisées.

Exemple 2.5 L'instruction `int("3")` renvoie `3` et donc permet de transtyper la chaîne de caractères `"3"` en un entier. En revanche, l'instruction `int("3.4")` renvoie une erreur.

2.2 Opérations sur les types

Savoir quel type de donnée est manipulé à un instant donné est extrêmement important. En effet, suivant le type manipulé, on pourra utiliser des opérations et des fonctions déjà implémentées. Bien sûr, ces opérations et ces fonctions vont dépendre du type de la donnée : on ne manipule pas de la même façon des nombres, des tableaux (assimilés au lycée à des listes) et des chaînes de caractères.

Les listes et les chaînes de caractères étant abordées dans des chapitres spécifiques, nous nous intéressons ici uniquement aux opérations algébriques et aux fonctions mathématiques.

2.2.1 Opérations algébriques

addition	soustraction	multiplication	division	puissance $n^{ème}$ de x
`+`	`-`	`*`	`/`	`x**n`

Remarque 2.6 Attention, les opérations doivent toujours être écrites. Par exemple, on écrit `2*x` et non pas `2x`.

2.2.2 Opérations avec les nombres entiers

Soient `a` et `b` deux nombres entiers avec $b \neq 0$. Il existe un unique couple d'entiers naturels (q, r) tels que $a = b \times q + r$ avec $0 \leqslant r < b$. Les nombres q et r s'appellent respectivement *le quotient* et *le reste de la division euclidienne de a par b*.

quotient de `a` par `b`	**reste de `a` par `b`**
`a//b`	`a%b`

Exemple 2.7 L'instruction `5//2` renvoie `2`, le quotient de la division euclidienne de 5 par 2 et l'instruction `5%2` renvoie `1`, le reste de la division euclidienne de 5 par 2.

2.2.3 Fonctions mathématiques

Arrondi et valeur absolue	`round(x,n)` : renvoie une valeur arrondie de `x` avec `n` décimales `abs(x)` : renvoie la valeur absolue de `x`

Fonctions arithmétiques (bibliothèque `math`)	`factorial(n)` : renvoie la factorielle de l'entier naturel `n`
	`floor(x)` : renvoie la partie entière (inférieure) de `x`
	`trunc(x)` : renvoie la troncature de `x` à l'unité

Exponentielle et logarithmes (bibliothèque `math`)	`exp(x)` : renvoie l'exponentielle de `x`
	`log(x)` : renvoie le logarithme népérien de `x`
	`log10(x)` : renvoie le logarithme décimal de `x`
	`sqrt(x)` : renvoie la racine carrée de `x`

Trigonométrie (bibliothèque `math`)	`cos(x)` : renvoie le cosinus de `x` (donné en radian) `sin(x)` : renvoie le sinus de `x` (donné en radian) `tan(x)` : renvoie la tangente de `x` (donné en radian)
Nombres aléatoires (bibliothèque `random`)	`random()` : renvoie un nombre pseudo-aléatoire dans $[0; 1[$ `uniform(a,b)` : renvoie un nombre pseudo-aléatoire dans $[a; b[$ `randint(a,b)` : renvoie un nombre entier aléatoire entre deux entiers `a` et `b` inclus

Exemple 2.8 L'instruction `abs(-5.3)` renvoie 5.3, la distance à zéro du nombre $-5{,}3$.
L'instruction `round(5/3)` renvoie `2` (arrondi à l'entier) et l'instruction `round(5/3,3)` renvoie `1.667` (arrondi au millième).

Pour importer une bibliothèque, on peut utiliser l'une ou l'autre des instructions suivantes :

```
from nom_bibliothèque import *
```

```
from nom_bibliothèque import nomFonction1, nomFonction2, ...
```

```
import nom_bibliothèque
```

Exemple 2.9 Les instructions suivantes permettent de calculer la racine carrée de 2 à l'aide de la bibliothèque `math` :

```
>>> from math import *
>>> sqrt(2)
1.4142135623730951
```

```
>>> from math import sqrt
>>> sqrt(2)
1.4142135623730951
```

```
>>> import math
>>> math.sqrt(2)
1.4142135623730951
```

2.3 Exercices

Exercice 2.10 Compléter le Notebook *NSI Première Partie 1 Chapitre 2 Types*.

Exercice 2.11 (QCM)

1. On saisit l'instruction `a = -4`
   Quel est le type de la donnée contenue dans la variable `a` ?
   (a) `bool`
   (b) `int`
   (c) `float`
   (d) `list`

2. On saisit les instructions suivantes :

```
from math import *
a = sqrt(2)
b = a - 5
a = int(round(b,0))
```

   Parmi les affirmations suivantes, lesquelles sont vraies à l'issue de ces instructions ?
   (a) il y a une erreur : `sqrt` ne veut rien dire
   (b) `a` contient la valeur `0`
   (c) `b` est de type `int`
   (d) `a` est de type `int`

3. On saisit les instructions suivantes :

```
a = 3
b = a / 2
a = b*4
```

   Parmi les affirmations suivantes, lesquelles sont vraies à l'issue de ces instructions ?
   (a) `a` est de type `int` et `b` de type `float`
   (b) `a` est de type `float` et `b` est de type `int`
   (c) `a` et `b` sont de type `float`
   (d) `a` et `b` sont de type `int`

4. On saisit les instructions suivantes :

```
from random import *
a = randint(1,6)
b = floor(6*random()) + 1
```

   Parmi les affirmations suivantes, lesquelles sont vraies à l'issue de ces instructions ?
   (a) `a` et `b` contiennent tous les deux des nombres entiers aléatoires entre 1 et 6
   (b) `a` et `b` peuvent avoir des valeurs identiques
   (c) les deux instructions permettent de simuler le lancer d'un dé équilibré à 6 faces
   (d) `a` et `b` sont de type `int`

5. On saisit les instructions suivantes :

```
a = 5
a = a**2
a = a // 2
a = a / 3
```

   Parmi les affirmations suivantes, lesquelles sont vraies à l'issue de ces instructions ?
   (a) `a` est de type `float`
   (b) `a` contient la valeur `4`
   (c) `a` est de type `int`
   (d) `a` contient la valeur `4.0`