Chap. 1 Introduction à la programmation Python Thème 1

Variables et affectation

Sommaire du chapitre 1

Dans un programme, les variables sont indispensables : elles permettent de stocker provisoirement (dans la mémoire de l'ordinateur) des valeurs. On peut alors y accéder, effectuer des opérations ou éventuellement les modifier.

I Variable

Les données d'un programme sont rangées dans des **variables**. Une variable est l'association entre un **nom** (un identifiant) et une **valeur** (d'un certain type). Chaque variable Python a un **nom**, qui ne doit pas contenir d'espace ni de caractères autres que des lettres, des chiffres et le blanc souligné (`_`). Le nom ne doit pas être un mot clé de Python.

Un nom de variable ne peut pas commencer par un nombre : `2pi` sera rejeté par Python comme incorrect. Les majuscules sont différentes des minuscules : la variable `Taille` n'est pas la même que la variable `taille` ou la variable `TAILLE`.

II Affectation

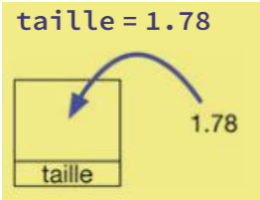
Pour stocker une valeur dans une variable, on utilise une instruction appelée **affectation**, notée avec le signe `=` (c'est-à-dire d'associer son nom à une valeur). Par exemple :

```
1  taille = 1.78
```

La variable `taille` contient un nombre à virgule flottante (type `float`).

Remarques

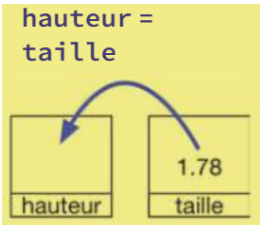
Ce signe égal ne traduit pas une égalité au sens mathématique. Il faut se représenter chaque variable comme une boîte étiquetée par son nom. L'affectation = copie la valeur de l'expression à droite du signe égal dans la boîte de la variable à gauche du signe égal. La valeur de la variable peut être modifiée à un autre endroit du programme par une autre affectation de cette variable.



Lorsque la variable apparaît ailleurs qu'à gauche d'une affectation, elle représente sa **valeur** :

```
hauteur = taille
```

Cette instruction affecte la valeur contenue dans la variable `taille` dans la variable `hauteur`. On copie le contenu de la boîte avec l'étiquette **taille**, et on remplace le contenu de la boîte avec l'étiquette **hauteur** par cette copie.



Donc si l'on change la valeur de `taille` par une nouvelle affectation, cela ne change pas la valeur de `hauteur`. A l'issue du programme suivant :

```
taille = 1.78
hauteur = taille
taille = 1.60
print(hauteur)
print(taille)
```

`hauteur` vaut 1.78 et `taille` vaut 1.60.

Pour bien avoir en tête que l'affectation n'est pas une égalité mathématiques, il est bon de se représenter le signe = plutôt comme une flèche vers la gauche ←, qui montre que l'on transfère une valeur vers une variable :

```
taille ← 1.78
hauteur ← taille
taille ← 1.60
```

Remarques

— En Python, si on souhaite afficher les deux nombres sur une même ligne, on peut remplacer les deux instructions `print` par une seule, en séparant les deux éléments à afficher par une virgule.

```
taille = 1.78
hauteur = taille
taille = 1.60
print(hauteur, taille)
```

Les deux éléments sont affichés sur la même ligne et séparés par un caractère espace.

— Plus généralement, on peut utiliser `print` avec un nombre arbitraire d'éléments, qui peuvent être des nombres ou des chaînes de caractères.

```
taille = 1.78
age = 16
print("Cet élève a",age,"ans et mesure",taille,"m.")
```

L'exécution de ce programme affiche le message suivant :
`Cet élève a 16 ans et mesure 1.78 m`

III Interaction avec l'utilisateur

Pour permettre l'interaction du programme avec l'utilisateur, par exemple la saisie de la valeur d'une variable, il faut procéder en deux temps : d'abord utiliser l'instruction `input` pour récupérer des caractères tapés au clavier par l'utilisateur, puis utiliser l'instruction `int` pour convertir cette chaîne de caractères en un nombre entier.

```
s = input("Entrez un nombre :")
a = int(s)
print("le nombre suivant est, a + 1")
```

L'instruction `input` interrompt l'exécution du programme et attend que l'utilisateur saisisse des caractères au clavier. La saisie se termine lorsqu'il appuie sur la touche `Entrée`. Ici, la suite des caractères saisis par l'utilisateur est stockées dans une variable `s`. Puis, la variable `a` reçoit le nombre représenté par cette suite de caractères.

Remarques

Les deux premières instructions du précédent programme peuvent être composées en une seule de la manière suivante :

```
a = int(input("Entrez un nombre :"))
print("le nombre suivant est, a + 1")
```

L'effet obtenu est quasiment identique : cette version composée récupère de même une saisie de l'utilisateur sous la forme d'une chaîne de caractères et la convertit en un nombre entier, le nombre obtenu étant stocké dans la variable `a`. Dans cette dernière version cependant la chaîne de caractères intermédiaires n'est pas stockée dans une variable et n'est donc plus accessible par ailleurs.