

CHAPITRE 2

LE CONCEPT DE VARIABLE

Définition d'une variable

2

- En programmation, une variable est un espace mémoire de l'ordinateur qui a été désigné de manière précise.
- Celui-ci sert à stocker certaines valeurs dont nous avons besoin dans notre programme.
- D'un autre point de vue, une variable correspond à la donnée de deux éléments :
 - ▣ un identifiant : le nom de la variable
 - ▣ une valeur : la valeur de la variable

Définition d'une variable

3

□ Exemple :

```
>>> x = 3
```

- Dans ce cas, on définit une variable dont l'identifiant est x et la valeur est 3.
- La variable est en fait un espace mémoire précis de l'ordinateur. On peut retrouver l'adresse de cet espace mémoire :

```
>>> id(x)
```

```
4297546624
```

Définition d'une variable

4

- L'identifiant correspond à un lien d'accès vers la case mémoire no 4297546624.
- La valeur 3 de la variable correspond au contenu de la case mémoire no 4297546624.
- Par abus de langage, on confondra la variable avec son nom. On parlera alors de la variable x dont la valeur est 3.

Définition d'une variable

5

- Contrairement au cas des mathématiques, en programmation, la valeur d'une variable peut (justement) varier au cours de l'exécution d'un programme.

```
>>> x = 2
```

```
>>> x
```

```
2
```

```
>>> x = "bonjour"
```

```
>>> x
```

```
'bonjour'
```

```
>>>
```

Affectation d'une variable

6

- En Python, le nom d'une variable doit commencer par une lettre et ne comportera jamais d'espace.

```
>>> ma_variable = 2
```

```
>>> ma_variable
```

```
2
```

```
>>> x1749 = 3.5
```

```
>>> lma_varibale = 3
```

```
File "<stdin>", line 1
```

```
    lma_variable = 3
```

```
        ^
```

```
SyntaxError: invalid syntax
```

Affectation d'une variable

7

- L'affectation directe consiste à assigner un certain nom de variable à une certaine valeur littérale (numérique, textuelle ou booléenne).
- La syntaxe de cette instruction d'affectation est la suivante:

`<variable> = <littéral>`

- Dans ce cas, le signe « = » n'a aucune valeur arithmétique. Il signifie « prend la valeur... ».
- Le processus d'affectation s'effectue de droite à gauche!

Affectation d'une variable

8

```
>>> x = 2
>>> y = 3.5
>>> x = y
>>> x
3.5
>>> y
3.5
>>>
```

- En effet, nous avons affecté à x la valeur de y.
- Donc c'est la valeur de x qui a changé.

Affectation d'une variable

9

```
>>> x = 2
>>> y = "bonjour"
>>> y = x
>>> x
2
>>> y
2
>>>
```

- En effet, nous avons affecté à y la valeur de x.
- Donc c'est la valeur de y qui a changé.

Affectation d'une variable

10

- L'affectation simple consiste à assigner un certain nom de variable à une certaine expression.
- La syntaxe de cette instruction d'affectation est la suivante:

`<variable> = <expression>`

- Comme avant, le signe « = » n'a aucune valeur arithmétique. Il signifie « prend la valeur... ».
- Le processus d'affectation s'effectue de droite à gauche...

Affectation d'une variable

11

```
>>> x = (2.2 ** 3) / 5
>>> x
2.1296000000000001
>>> longueur = 2.5
>>> largeur = 3.5
>>> surface = longueur * largeur
>>> surface
8.75
>>> jour = "2 juin"
>>> annee = "2002"
>>> date = "le " + jour + " " + annee
>>> date
'le 2 juin 2002'
```



Affectation d'une variable

12

- L'affectation composée consiste à assigner un certain nom de variable à une certaine expression comportant le nom de la variable elle-même.
- La syntaxe de cette instruction d'affectation est la suivante:

```
<variable> = <expression_avec_variable>
```
- Ce processus est très utilisé en programmation.
- Comme avant, le processus d'affectation s'effectue de droite à gauche...

Affectation d'une variable

13

```
>>> x = 0
```

```
>>> x = x + 1
```

```
>>> x
```

```
1
```

```
>>>
```

- Il faut comprendre : la nouvelle valeur de x correspond à l'ancienne valeur incrémentée de 1.
- On écrit aussi cela : $x += 1$

Affectation d'une variable

14

```
>>> x = 1
```

```
>>> x = x * 3          # on peut aussi écrire x *= 3
```

```
>>> x
```

```
3
```

```
>>> x = x ** 4         # on peut aussi écrire x **= 4
```

```
>>> x
```

```
81
```

```
>>> x = x / 2          # on peut aussi écrire x /= 2
```

```
>>> x
```

```
40.5
```

```
>>> x = (x * x) / (x - 1)
```

```
>>> x
```

```
41.5253164556962
```

Affectation d'une variable

15

```
>>> chaine = "Tex"
>>> chaine = chaine + chaine
>>> chaine
'TexTex'
>>> chaine = chaine * 3 # on peut écrire chaine *= 3
>>> chaine
'TexTexTexTexTexTex'
>>>
```

Affectation de variables

16

- L'affectation simultanée consiste à assigner plusieurs noms de variables à plusieurs valeurs ou expressions.

- La syntaxe de cette instruction d'affectation est la suivante:

`<var_1>, ..., <var_k> = <exp_1>, ..., <exp_k>`

- Les variables et expressions correspondantes sont séparées par des virgules.

Affectation de variables

17

```
>>> x, y, z = 2.5, "bonjour", True
```

```
>>> x, y, z
```

```
2.5 bonjour True
```

```
>>> prenom, nom = "Alma", "C."
```

```
>>> "Bonjour " + prenom + " " + nom + " " + "!"*3
```

```
Bonjour Alma C. !!!
```

Compilation de fichiers Python

18

- ❑ Désormais, nous allons arrêter de coder en mode purement interactif, via l'invite de commandes.
- ❑ Nous allons créer programmes Python qui se trouvent dans des fichiers, afin de pouvoir les sauvegarder et les réutiliser plus tard.
- ❑ Pour cela, utilisez un éditeur de type Notepad++.
Une fois votre code écrit, sauvegardez votre fichier au format « .py », qui correspond à du code Python.

Compilation de fichiers Python

19

- Une fois votre fichier sauvegardé, avec votre invite de commande, placez-vous dans le répertoire dans lequel se trouve votre fichier.
- Pour compiler et exécuter votre programme Python, taper la commande

```
python nom_fichier.py
```

- Votre programme Python s'exécute alors dans votre invite de commandes.

Instructions d'entrée-sortie

20

- Pour implémenter une « instruction d'entrée », i.e., une capture d'information qui va de l'utilisateur vers l'application, on utilise les fonctions :

`input(<message>)` et `raw_input(<message>)`

- En mode interactif, ces fonctions affichent dans l'invite de commandes la chaîne de caractères `<message>`, attendent que l'utilisateur tape quelque chose à la suite de `<message>` (suivi de « Enter »), et retournent ce quelque chose.

Instructions d'entrée-sortie

21

- En Python2, la fonction `raw_input()` retourne toujours une chaîne de caractères.
- La fonction `input()` peut deviner le type de ce qui est entré par l'utilisateur.
 - ▣ Si c'est un nombre, elle retourne un nombre
 - ▣ Si c'est une chaîne, elle retourne une chaîne
 - ▣ Si c'est une liste, un tuple ou un dico, elle retourne une liste, un tuple ou un dico (cf. chapitres suivants)

Instructions d'entrée-sortie

22

- En Python 3, la fonction `raw_input()` n'existe plus, seule la fonction `input()` demeure.
- Et cette fonction `input()` convertit tout en chaîne de caractères.

Instructions d'entrée-sortie

23

- En mode interactif, ces fonctions ne sont pas très intéressantes...

```
>>> input("Bonjour: ")
```

```
Bonjour: merci
```

```
'merci'
```

```
>>>
```

```
>>> input("Ceci est un message")
```

```
Ceci est un messageallo
```

```
'allo'
```

```
>>>
```

Instructions d'entrée-sortie

24

- En mode interactif, ces fonctions ne sont pas très intéressantes...

```
>>> input("Quel est votre âge? ")
```

```
Quel est votre âge? 18
```

```
'18'
```

```
>>>
```

```
>>> input("Entrer un nombre à virgule: ")
```

```
Entrer un nombre à virgule: 2.34
```

```
'2.34'
```

```
>>>
```


Instructions d'entrée-sortie

25

- En mode non interactif, ces « instructions d'entrée » deviennent intéressantes...
- Dans ce cas, il convient de stocker – dans une variable – l'information qui va de l'utilisateur vers l'application, grâce aux instructions :

```
var = input(<message>)
```

```
var = raw_input(<message>)
```

- Ainsi, Python affiche dans l'invite de commandes la chaîne de caractères <message>, attend que l'utilisateur tape quelque chose à la suite de <message> (suivi de « Enter »), et stocke ce quelque chose dans la variable var.

Instructions d'entrée-sortie

26

Programme

```
nom = raw_input("Quel est votre nom? ")  
prenom = raw_input("Quel est votre prénom? ")  
print(prenom + " " + nom)
```

Exécution

Quel est votre nom? C.

Quel est votre prénom? Dan

Dan C.

Instructions d'entrée-sortie

27

Programme

```
age = input("Quel est votre âge? ")
print("Vous avez " + age + " ans. ")
print("Dans " + str(100 - int(age)) + \
      " ans, vous aurez 100 ans.")
```

Exécution

Quel est votre âge? 92

Vous avez 92 ans.

Dans 8 ans, vous aurez 100 ans.

Instructions d'entrée-sortie

28

- Dans l'exemple précédent, on a effectué le calcul $100 - \text{age}$.
- Puis, on a utilisé la « fonction de transtypage » `str()` qui convertit son argument en une chaîne de caractères (string), afin de pouvoir l'imprimer.
- Cette question des types de données fait l'objet du chapitre suivant.

Instructions d'entrée-sortie

29

- L'« instruction de sortie », i.e., qui implémente une transmission d'information qui va de l'ordinateur vers l'utilisateur, est maintenant déjà connue, et est implémentée, en Python2, par la fonction :

```
print <message>
```

- Ou alors en Python 3 :

```
print(<message>)
```