Les variables

À retenir pour la suite

Lors de l'exécution d'un programme en Python (et dans de nombreux autres langages, dits « impératifs »), l'ordinateur exécute **séquentiellement** (c'est-à-dire dans l'ordre, de manière successive) les instructions écrites sur chacune des lignes.

Bonne pratique : une ligne = une instruction.

Nous verrons par la suite qu'il est possible de rediriger l'exécution du programme vers une partie antérieure ou postérieure du code grâce aux « structures de contrôle ». Elles sont indispensables pour réaliser un programme complexe (il faudrait sinon écrire toutes les instructions « à la main »).

1 Définitions

- Une variable est un élément qui associe un nom à une valeur. Cette dernière peut être de nature différente : nombre, texte, etc. Une variable est physiquement implantée dans la mémoire de l'ordinateur. Sa valeur peut varier au cours de l'exécution du programme.
- Une variable porte un nom qui commence par une lettre suivie de lettres (non accentuées), de chiffres et d'*underscores* : _
  Attention à ne pas utiliser de mot réservé par Python ! Par exemple, on n'appellera pas une variable `def`.
- Par convention, en Python, on choisit d'écrire les noms de variables en « snake case ».
- L'opération permettant de créer ou de modifier une variable s'appelle « affectation » et se fait grâce au symbole = .

Ne pas confondre

Il ne faut pas confondre l'égalité mathématique avec l'affectation bien que ce soit le même symbole qui est utilisé.

Exemple :

```
1 x = 5
2 x = x + 3
```

La ligne 2 n'aurait aucun sens en mathématiques, il s'agit ici d'affecter une nouvelle valeur à x : la valeur de x précédant l'affectation à laquelle on ajoute trois.
Une variable ne contient qu'une valeur à la fois, les précédentes valeurs sont écrasées au fur et à mesure des affectations.
Pour afficher la valeur d'une variable, on utilise la fonction `print` issue de la bibliothèque standard de Python.

Exemple :

```
1 ma_variable = 11
2 print(ma_variable)
```

Ce script affichera : `11`

Exercice 1

On donne le programme suivant :

```
x = 5
print(x)
x = x + 3
print(x)
```

1. Qu'est-ce qui va s'afficher à l'exécution ?
2. Dans votre éditeur Python, créer un fichier nommé `variables.py`, enregistré dans votre espace de travail. Y copier le programme précédent et exécuter-le. Aviez-vous trouvé la bonne réponse ?

2 Le type d'une variable

Dans un programme informatique, chaque variable appartient à une catégorie donnée, appelée type .
Les variables élémentaires ont des types dits primitifs. En python, il s'agit essentiellement :

- des booléens (`bool`) ;
- des entiers (`int`), positifs ou négatifs, de grandeur arbitraire (spécificité de python) ;
- des flottants (`float`), en double précision (64 bits) ;
- des chaînes de caractères (`str`) ;
- des listes (`list`) et des tuples (`tuple`) ;
- des dictionnaires (`dict`).

Dans de nombreux langages, le type d'une variable est explicitement précisé à sa création (ou déclaration), et il est alors impossible d'en changer le type. Mais, en python, la déclaration du type d'une variable est implicite, ce qui signifie que le langage le reconnaît automatiquement. Il est en outre possible de changer le type d'une variable (ce qui cependant ne fait pas partie des bonnes pratiques).

Exercice 2

La fonction `type` permet d'identifier le type d'une variable.
Quels sont les affichages produits par le script ci-dessous ? Pourquoi ?

```
x = 2020*2
print(x, type(x))

x = 2020.
print(x, type(x))

x = 1+1<=2
print(x, type(x))

x = 2019/2
print(x, type(x))

x = 2019//2
print(x, type(x))

x = 10 % 3
print(x, type(x))

x = "nsi"
print(x, type(x))

x = [2, 4]
print(x, type(x))

x = {"nsi1": 24, "nsi2": 24}
print(x, type(x))

x = (2022, 2023)
print(x, type(x))
```

Outre la clarté du code, l'intérêt (et la nécessité) des types vient du fait qu'un même opérateur peut désigner des opérations différentes : cette possibilité porte le nom de surcharge d'opérateurs.

Par exemple, l'opérateur « `+` » désigne :

- l'addition pour des nombres, et celle-ci n'est pas effectuée de la même manière selon qu'il s'agisse d'entiers ou de flottants ;
- la concaténation pour des chaînes de caractères ;
- l'extension (ou réunion) pour des listes.

Les langages fournissent en général des fonctions permettant de convertir, lorsque c'est « naturellement » possible, une variable d'un type vers un autre. En python, les principales fonctions de conversion (*casting*, en anglais) sont :

- `int` : conversion en entier ;
- `float` : conversion en flottant ;
- `str` : conversion en chaîne de caractères ;
- `list` : conversion en liste.

Exercice 3

Quels sont les affichages produits par le script ci-dessous ? Pour répondre à cette question, vous devez d'abord dérouler le code à la main, sur papier, et donner votre réponse, puis utiliser votre éditeur Python pour l'exécuter et vérifier le résultat précédent.

```
x = int(1.999)
print(x)
s = str(3.14)
t = s+s
print(t)
```

Exercice 4

1. Dans une console Python, taper les lignes suivantes et observer leurs résultats :

   ```
   >>> 5 - 3 - 2
   >>> 1 / 2 / 2
   ```

   Que peut-on en déduire sur le fonctionnement de la soustraction et de la division en Python ?
2. Réécrire les opérations suivantes en explicitant toutes les parenthèses (le résultat est inchangé avant et après) et vérifier les résultats dans votre console :

   (a) `1 + 2 * 3 - 4`
   (b) `1+2 * 4*3`
   (c) `1-a+a*a/2-a*a*a/6+a*a*a*a/24`
3. Réécrire les expressions suivantes en utilisant aussi peu de parenthèses que possible sans changer le résultat :

   (a) `1+(2*(3-4))`
   (b) `(1+2)+((5*3)+4)`
   (c) `(1-((2-3)+4))+(((5-6)+((7-8)/2)))`

Exercice 5

1. Que fait le programme suivant ?

```
a = 5
b = 6
a = b
b = a
```

2. Que fait le programme suivant ?

```
a = 7
b = 8
tmp = a
a = b
b = tmp
```

Exercice 6

On rappelle que la division euclidienne (ou division entière) entre deux nombres entiers naturels `a` et `b` donne un reste `r` et un quotient `q` est telle que : `a = (b * q) + r` (avec `r < b`).

On veut écrire un programme qui calcule le nombre de boîtes de 6 œufs nécessaire au transport d'un certain nombre d'œufs.

1. Questions préalables :
   (a) Quel opérateur permet d'obtenir le quotient d'une division euclidienne (entière) en Python ?
   (b) Quel opérateur permet d'obtenir le reste d'une division euclidienne (entière) en Python ?
   (c) Quel opérateur permet de faire une division décimale en Python ?
2. Un élève propose la solution suivante : `nb_boites = nb_oeufs // 6`
   (a) Pour quelles valeurs de `nb_oeufs` ce programme est-il correct ?
   (b) Un autre élève suggère d'écrire plutôt : `nb_boites = nb_oeufs // 6 + 1` Est-ce correct ? Pourquoi ?
   (c) Proposer une solution correcte.