Chap. 1 Introduction à la programmation Python Thème 1

Expressions

Sommaire du chapitre 1

Une **expression** est le résultat d'un calcul effectuer par le programme. Une expression permet de calculer une valeur (et donc son type) en combinant entre elles d'autres valeurs avec des **opérateurs**. Les valeurs utilisées peuvent être des constantes (nombres, chaînes de caractères), des valeurs stockées dans des variables (`taille`), ou des résultats de fonctions (`sin(x)`). Les opérateurs les plus courants sont les opérateurs arithmétiques, logiques, et de manipulation de chaînes.

I Opérateurs arithmétiques

On peut calculer des formules mathématiques avec des opérations arithmétiques usuelles : addition `+`, soustraction `-`, multiplication `*`, division `/`.
La puissance s'écrit `**` : `5**4` vaut `625`.
La division entière s'écrit `//` et son reste `%` : `3/2` vaut `1.5` mais `3//2` vaut `1`

```
1 72 // 60    # vaut 1
2 72 % 60     # vaut 12
```

La division euclidienne de 72 par 60 a pour quotient 1 et pour reste 12. Donc 72 minutes correspondent à 1 heure et 12 minutes.

Remarques

Le caractère `#` introduit un **commentaire** qui se poursuit jusqu'à la fin de la ligne.
Les commentaires permettent d'expliquer et de documenter le code. Ils sont ignorés par l'ordinateur, mais très utiles au lecteur humain.

Comme en mathématiques, les opérateurs arithmétiques sont groupés par priorité : `+` et `-` ont la plus faible priorité, `*`, `/`, `//` et `%` la priorité intermédiaire, et `**` la plus haute priorité. Les parenthèses permettent de grouper les calculs : `3 + 5 // 3 ** 4` vaut `3`, et `(((3 + 5) // 3) ** 4` vaut `16`.

En plus des opérateurs classiques, le langage Python inclut de nombreuses fonctions mathématiques.
En important la bibliothèque `math`, on a accès à des fonctions telles que l'arrondi (`round(x)`), la racine carrée (`sqrt(x)`), les fonctions trigonométriques comme `cos(x)` et `sin(x)` ; logarithmiques comme `log(x)` et `exp(x)`, les constantes π (`pi`) et e (`e`), ou encore les conversions entre degrés et radians (`degrees(r)`, `radians(d)`).

Les opérateurs arithmétiques et les fonctions mathématiques s'appliquent à des nombres et à des variables contenant des nombres.
Par exemple :

```
from math import *                  # importer la bibliothèque "math"
delta = b**2 - 4*a*c                # calculer le discriminant
x1 = (-b + sqrt(delta)) / (2*a) # première racine
x2 = (-b - sqrt(delta)) / (2*a) # deuxième racine
```

Remarques

En Python, `type(expression)` donne le type de l'expression :

```
print(type(3 + 5))      # vaut <class 'int'>
print(type(3 + 5.5))    # vaut <class 'float'>
```

II Opérateurs logiques

Les comparaisons entre expressions numériques donnent des résultats booléens, c'est-à-dire `True` ou `False`. Le test d'égalité s'écrit `==` (avec deux fois le signe `=`), l'inégalité s'écrit `!=` et les autres comparaisons numériques sont `<`, `<=`, `>`, `>=`.
Les comparaisons ont une priorité plus faible que les opérateurs arithmétiques :

```
3 + 4 < 6          # vaut False, car 7 > 6
1 + 2 == 6 / 2  # vaut True
```

On peut combiner des expressions à valeur booléenne avec `or`, `and` et `not` :

```
x < 0 or x > 100     # vaut False si x est dans [0,100]
y >= 0 and y <= 100 # vaut True si y est dans [0,100]
0 <= z <= 100        # autre façon d'exprimer l'inégalité précédente
```

```
4 not majeur          # vaut False si majeur est True, et inversement
```

Remarques

Priorité croissante des opérateurs dans les expressions :

- `or`
- `and`
- `not`
- `in`, `not in`
- `==`, `!=`, `<`, `<=`, `>`, `>=`
- `+`, `-`
- `*`, `/`, `//`, `%`
- `**`
- `(...)`

III Opérateurs sur les chaînes de caractères

L'opérateur `+` permet de concaténer deux chaînes de caractères, et l'opérateur `*` de répéter une chaîne un nombre entier de fois :

```
1 'bon' + 'jour' # vaut 'bonjour'
2 'ha' * 3       # vaut 'hahaha'
```

Les opérateurs de comparaison comparent les chaînes selon l'ordre alphabétique. Cependant les majuscules sont différentes des minuscules, et toutes les lettres majuscules précèdent les lettres minuscules : `'A' < 'Z' < 'a' < 'z'`. On peut également tester si une chaîne apparait dans une autre chaîne avec l'opérateur `in`, ou si elle n'apparait pas avec l'opérateur `not in`.

```
1 'bon' < 'jour'      # vaut True
2 'ha' == 'HA'        # vaut False
3 'ou' in 'jour'      # vaut True
4 'on' not in 'Bon'   # vaut False
```

Une fonction utile pour manipuler les chaînes de caractères est la fonction `len(s)` qui retourne la longueur de la chaîne `s` : `len('NSI') == 3` est vrai.

Remarques

La fonction `print` affiche du texte : `print("Bonjour !")`.
La tortue Python utiliser quant à elle la fonction `write` : `write("Hello !")`