Python les bases

Opérations et instructions de base : print, input

Programme et Instructions

Un **algorithme** est une suite d'étapes à suivre pour atteindre un objectif. On peut l'écrire en langage naturel ou en pseudo code.

Un **programme** informatique est l'implémentation d'un algorithme dans un langage de programmation, il peut être exécuté dans une machine.

Un programme est constitué d'une suite finie d'**instructions**:

```
<instruction 1>
<instruction 2>
    . . .
<instruction n>   `
```

Ces instructions peuvent être de différentes natures, voici quelques exemples dans le langage Python:

Opération en entre données

```
################################
# Opération entre des données #
################################

2 + 2       # addition entre deux nombres entiers
2 * 2       # multiplication entre deux nombres entiers
2 ** 3      # puissance: 2 à la puissance 3

7 / 2       # division décimale entre deux entiers (le résultat n'est plus forcément un entier)
7 // 2      # quotient de la division euclidienne entre deux entiers
7 % 2       # reste de la division euclidienne entre deux entiers

'Python' + "Introduction" # concaténation de deux chaînes de caractères (textes)
'Python' * 3              # concaténation de plusieurs fois la même chaîne de caractères


#################################################
# Appel à des services du système d'exploitation #
#################################################

print("Python Introduction")      # affichage d'un texte sur la sortie standard (l'écran)
print("2 x 2 =", 2 * 2 )          # affichage d'un texte puis du résultat d'une expression arithmétique
input("Donnez votre réponse:")    # récupération d'un texte depuis l'entrée standard (le clavier)
```

Notion de variables

Qu’est-ce qu’une variable ?

On a souvent besoin dans un programme de conserver une donnée en mémoire en vue d'un traitement ultérieur, on utilise pour cela une **variable** qui est représentée par un nom.

```
a = 3
```

Cette instruction est une instruction d'**affectation**: on affecte à la variable nommée ***a*** la valeur ***3***:

En termes simples, une variable est **un conteneur dans lequel tu peux stocker différents types de données**, comme des nombres, des chaînes de caractères ou même des objets plus complexes.

Règles pour nommer les variables Python

Il y a quelques règles importantes à suivre lors de la déclaration de variables en Python :

- les noms de variables ne peuvent contenir que des lettres, des chiffres et des underscores, ou tiret bas (_) ;
- ils doivent commencer par une lettre (majuscule ou minuscule) ou un underscore, mais ne peuvent pas commencer par un chiffre ;
- les noms de variables sont sensibles à la casse, ce qui signifie que nom et Nom seraient deux variables distinctes ;
- il faut éviter d’utiliser des noms de variables déjà réservés par Python, tels que print, if, else, etc.

Les différents types de données des variables Python

Les variables en Python peuvent contenir différents types de données tels que :

- **entiers** (int) : ils représentent les nombres entiers, positifs ou négatifs, comme 42 ou –10 ;
- **décimaux** (float) : ils représentent les nombres à virgule flottante tels que 3.14 ;
- **chaînes de caractères** (str) : elles représentent du texte et doivent être entourées de guillemets simples ou doubles, par exemple, “Bonjour” ou ‘Python’ ;
- **booléens** (bool) : ils représentent des valeurs de vérité, True ou False.
-

Les variables sur Python choisissent automatiquement leur type.

Type	Nature	Exemple	Opération
int (integer)	nombre entier relatif	2 1000000 -3 2020	+ (addition) * (multiplication) / (division décimale) ** (puissance) // (quotient division euclidienne) % (reste division euclidienne)
float (floating point)	nombre décimal relatif	2.3 -1.7245	+ (addition) * (multiplication) / (division décimale)
str (string)	chaîne de caractères (caractère, texte)	"Hello" "2020"	+ (concaténation) * (concaténation multiple)
bool (boolean)	booléen	True False	and (et logique) or (ou logique) not (non logique)

Il est très important de maîtriser la typologie car les opérations et d'autres manipulations ne sont pas les mêmes selon le type de la donnée:

```
# Initialisation des variables (première affectation)
nombre_a = 3
nombre_b = 5
string_a = 'texte a'
string_b = 'texte b'
booleen_a = True
booleen_b = False

# opérations correctes
nombre_a + nombre_b     # addition de deux nombres (résultat: 8)
string_a + string_b     # concaténation de deux textes (résultat: 'texte atexte b')
booleen_a and booleen_b # et logique entre deux booléens (résultat: False)
booleen_a or booleen_b  # ou logique entre deux booléens (résultat: True)
not booleen_a           # non logique d'un booléen (résultat: False)

# opérations incorrectes: l'exécution entraîne une erreur (ou le résultat est inconsistant)
```

On peut connaître le type d'une variable avec la fonction: `type(nom_variable)`

Mais tu peux aussi convertir une donnée d'un type à un autre en utilisant la commande int() ou float() ou bool().

On peut changer le type d'une variable (lorsque c'est possible) avec une opération de convertion (casting):

```
# initialisation des variables(changement de type)
a = 3
print( type( a ) ) # l'instruction affichera int
b = '2.3'
print( type( b ) ) # l'instruction affichera str

# casting
c = str( a )        # la fonction str() convertit le nombre en string
print( type( c ) ) # l'instruction affichera str

d = int( c )        # la fonction int() convertit le texte en nombre entier (si c'est possible !)
print( type( d ) ) # l'instruction affichera int

e = float( b )      # la fonction float() convertit le texte en nombre décimal (si c'est possible !)
print( type( e ) ) # l'instruction affichera float
```