Notion de variable en Python

1. Définition

De manière générale, une **variable** est une zone de la mémoire de l'ordinateur dans laquelle une **valeur** est stockée.
Cette variable est définie par un **nom**, alors que pour l'ordinateur, il s'agit d'une zone particulière de la mémoire (appelée adresse).

On peut visualiser une variable comme une boîte dans laquelle est stockée une valeur qui peut être modifiée.

Pour créer une variable, il faut :

- la nommer : on parle de déclaration de variable de la variable
- **lui stocker une première valeur : on parle d'initialisation de la variable**

En Python, déclaration et initialisation se font en même temps :

Testez les instructions suivantes dans le shell :

`In [1]: x=2`
 	
`In [2]: x`
`Out[2]: 2`
 	
`In [3]: 1=y`
`Out[3]:`
	Pouvez-vous prévoir ce que l'on obtient en sortie Out [3] ?

Que s'est-il passé ?

In [1] : x=2 : on a déclaré, puis initialisé la variable x avec la valeur 2.
Python a « deviné » que la variable était un entier (on dit que Python est un langage au typage dynamique.

- Python a réservé l'espace en mémoire pour y accueillir un entier.
- Python a aussi fait en sorte qu'on puisse retrouver la variable sous le nom x.
- Enfin, Python a assigné la valeur 2 à la variable x.

La simple instruction x = 2 a suffi à réaliser les 3 étapes en une fois !

In [2] : x et Out [2] : 2 l'interpréteur nous a permis de connaître le contenu de la variable juste en tapant son nom.

In [3] : 1=y

2. Les types de variables

a) Définition

Le **type** d'une variable correspond à la nature de celle-ci.

Les principaux types dont nous aurons besoin dans un premier temps sont :

- les **entiers** (integers)
- les **nombres décimaux** (flottants)
- les **chaînes de caractères** (strings)
- les **listes**
- les **tableaux** (arrays)
- les **booléens** (true, false).

Les nombres réels dont la représentation décimale est infinie ne peuvent être représentés autrement que par une approximation (sinon, il nous faudrait une capacité de stockage infinie !). Ils seront donc représentés que par des flottants c'est-à-dire des décimaux.

Remarque

II existe de nombreux autres types (par exemple, les nombres complexes, etc.).

Dans l'exemple précédent, nous avons stocké un nombre entier dans la variable x, mais il est tout à fait possible de stocker des décimaux, des chaînes de caractères ou de nombreux autres types de variable.

`In [5]: y=3.14`
 	
`In [6]: y`
`Out[6]: 3.14`
 	
`In [7]: a="bonjour"`
 	
`In [8]: a`
`Out[8]: 'bonjour'`
 	
`In [9]: a='bonjour'`
 	
`In [10]: a`
`Out[10]: 'bonjour'`
 	
`In [11]: a=bonjour`
	Pouvez-vous prévoir ce que l'on obtient en sortie Out [11] ?

b) L'instruction type()

En Python, on dispose de l'instruction **type()** permet de savoir de quel type est une variable.

`In [12]: type(x)` `Out[12]: int`	La variable x contient l'entier 2. Les entiers seront donc représentés par le qualificatif **int** *(pour integer : entier)*
`In [13]: type(y)` `Out[13]: float`	La variable y contient le décimal 3.14. Les décimaux seront donc représentés par le qualificatif **float** *(pour flottant)*
`In [14]: type(a)` `Out[14]: str`	La variable a contient la chaîne de caractère " bonjour ". Les chaînes de caractère seront donc représentés par le qualificatif **str** *(pour string : chaîne)*

3. Nommer une variable

Règles à suivre

Le nom des variables en Python peut être constitué :

- de lettres minuscules (a à z)
- de lettres majuscules (A à Z),
- de nombres (0 à 9)
- du caractère souligné (_).
- Vous ne pouvez pas utiliser d'espace dans un nom de variable.
- ne doit pas débuter par un chiffre
- Absolument éviter d'utiliser un mot « réservé » par Python comme nom de variable (par exemple : print, range, for, from, etc.).

À savoir , Python est sensible à la casse (différencie les minuscules et majuscules) , ce qui signifie que les variables TesT, test ou TEST sont différentes.

4. Opérations sur les variables

a) Opérations sur les types numériques

Prédire les sorties Out[…].

```
In [15]: x=45

In [16]: x+2
Out[16]:

In [17]: x-2
Out[17]:

In [18]: x*3
Out[18]:

In [19]: y=2.5

In [20]: x-y
Out[20]:

In [21]: (x*10)+y
Out[21]:

In [23]: z=8

In [24]: z/x
Out[24]:

In [25]: x=2

In [26]: z/x
Out[26]:
```

De quel type est la variable z en Out[29]?

```
In [27]: y=z/x

In [28]: z=x+y

In [29]: z
Out[29]:
```

b) Opérations sur les chaînes de caractère

Prédire les sorties Out[...], puis les vérifier dans le shell.

```
In [30]: chaine="bonjour"

In [31]: chaine
Out[31]:

In [32]: chaine+"prépa"
Out[32]:

In [33]: chaine*3
Out[33]:

In [34]: mot="2"

In [35]: mot+mot
Out[35]:

In [36]: 4*mot
Out[36]:

In [37]: mot*2.0
```

Commentaires

5. Conversion de type

Nous serons souvent amené à convertir les types, c'est-à-dire passer d'un type numérique à une chaîne de caractères ou vice-versa. Cela se fera avec les fonctions int(), float() et str().

Exemples

```
In [38]: i=3

In [39]: str(i)
Out[39]: '3'

In [40]: i='456'

In [41]: int(i)
Out[41]: 456

In [42]: float(i)
Out[42]: 456.0

In [43]: i='3.1416'

In [44]: float(i)
Out[44]: 3.1416
```

6. Applications

a) Prédire le résultat des instructions suivantes :

(1+2)**3

'da'*4

'da'+3

('pa'+'la')*2

('da'*4)/2

b) Prédire le résultat des instructions suivantes :

- str(4) * int("3")
- int("3") + float("3.2")
- str(3) * float("3.2")
- str(3/4) * 2

c) x=10

y=x

x=15

Après ces instructions, que valent x et y ?

d) x=42

y=10

x=y

y=x

Après ces instructions, que valent x et y ?

e) Taper dans l'éditeur :

```
1 x=(input('donner un entier'))
2 x=x+2
3 print(x)
```

puis exécuter le script.

- A quoi sert l'instruction input() ? De quel type est la variable x à la ligne 1 ?

- Comment comprenez-vous le message que vous renvoie Python ? Comment pourriez-vous contourner ce problème ?

f) complément

Écrire une suite d'instructions permettant d'échanger le contenu des variables x et y.

g) complément bis

Quelle est la valeur des variables a, b et c à la fin du programme suivant ?

```
a = 1
b = 1
c = 1
a = b + c
while b < 5:
    a = a + 1
    c = 2*c
    b = b + 1
b = 3*b
```