Types simples

I. Généralités

Tout programme informatique nécessite l'utilisation de données. En fonction des besoins, il peut manipuler des entiers (ex: le nombre d'articles achetés dans un magasin), des flottants (ex: le taux de TVA) ou des booléens (ex: Vrai / Faux). Ces types sont dits simples car ils ne contiennent qu'une seule valeur.

II. Variables et constantes

Certaines données sont amenées à changer durant l'exécution d'un programme, dans ce cas, on a affaire à une variable (ex: indice d'une boucle).
Par contre, d'autres données ne changent pas (ex: nombre maximum de tentatives de saisies d'un mot de passe), ce sont des constantes.

III. Entiers

La plupart des langages informatiques possède différents types d'entiers. Python utilise deux types d'entiers: int et long. Les deux types sont signés: ils acceptent des valeurs positives ou négatives. En mathématiques, on dit que ce sont des entiers relatifs.
Les entiers int sont compris entre -2147483648 et 2147483647 (ils sont codés sur 32 bits).
Les entiers longs ont, en théorie, une étendue illimitée. En pratique, les capacités de la machine limitent l'étendue de ces nombres.

Certains langages informatiques nécessitent que l'on déclare les variables avant de les manipuler, ce n'est pas le cas de Python. Elles sont créées lors de leur première affectation.

L'affectation d'une variable consiste à lui donner une valeur. Elle se fait en utilisant la syntaxe nom_variable=valeur_variable. Par exemple, l'instruction nb_repetitions=112 est valide. Les nombres entiers peuvent être saisis en décimal (ex: w=122), en octal (ex: x=0o11), en hexadécimal (ex: y=0x12) ou en binaire (ex: z=0b111).

Il est possible d'utiliser l'ancienne valeur d'une variable pour calculer la nouvelle. Dans ce cas, le nom de la variable se trouve à gauche et à droite du symbole =. A droite du égal, le nom de la variable représente son ancienne valeur. Exemple:

```
>>> a= 5
>>> a=a+1
>>> a
6
```

L'instruction a=a+1 signifie a vaut maintenant son ancienne valeur incrémentée de 1.

Toutes les opérations courantes sur les nombres entiers sont possibles. Voici quelques exemples:

Symbole	Action	Exemple
+	Addition	>>> 2+3 5
-	Soustraction	>>> 2-6 -4
*	Multiplication	>>> 4*5 20
/	Division entière	>>> 8 / 3 2
%	Modulo (reste de la division euclidienne)	>>> 10 % 4 2
**	Puissance	>>> 3**4 81

Il est possible de combiner différents opérateurs en une seule expression. Exemple: nbr=(5*17)%8. Les règles de priorités des opérations sont les mêmes qu'en mathématiques.

Pour comparer deux valeurs, on n'utilise pas le symbole = mais le symbole ==. Voici un exemple de code fonctionnel:
>>> a=10
>>> b=11
>>> a==b
False
La valeur False signifie faux, ce qui est la bonne réponse dans le cas de notre exemple.

En dehors de l'égalité, il est possible d'effectuer d'autres comparaisons. Le tableau ci-dessous en présentent quelques unes.

Symbole	Signification	Résultat si a=5 et b=6
!=	Teste si deux variables sont différentes	>>> a!=b True
<=	Teste si le terme de gauche est inférieur ou égal à celui de droite	>>> a<=b True
<	Teste si le terme de gauche est inférieur à celui de droite	>>> a<b True
>=	Teste si le terme de gauche est supérieur ou égal à celui de droite	>>> a>=b False
>	Teste si le terme de gauche est supérieur à celui de droite	>>> a>b False

IV. Flottants

Les nombres flottants sont des nombres à virgule flottante. Les flottants correspondent plus ou moins aux nombres réels. La différence essentielle est que les flottants ont une précision finie tandis que les nombres réels ont une précision infinie (=nombre infini de chiffres après la virgule).

Toutes les notions étudiées précédemment avec les entiers restent globalement valides.

Pour saisir les nombres flottants, on peut utiliser la notation scientifique. Par exemple, on peut représenter 1445 par 1.445e3. Le symbole e correspond à 10 puissance 3.

V. Booléens

Un booléen est une donnée qui ne prend que deux valeurs: vrai (True) ou faux (False).

L'affectation se fait de la même façon que pour les entiers (ex: resultat=True).

Il est possible de comparer l'égalité ou la différence entre deux variables booléennes en utilisant les opérateurs == et !=.

Il est a noter que False est considéré comme inférieur à True (False<True renvoit True). Cela est dû au fait, qu'en interne, les booléens sont stockés en tant qu'entiers.

Les booléens sont très importants en programmation car ils permettent de prendre des décisions (si une variable ou une expression est vraie alors on peut exécuter une action ou non). Nous reverrons cette notions au cours des chapitres sur les structures conditionnelles et sur les boucles.

Comme en mathématiques, il est possible de combiner des booléens au sein d'expression plus complexes à l'aide d'opérateurs logiques. Les opérateurs les plus courants sont le NON (not), le ET (and) et le OU (or).

Exemple d'instructions exploitant les booléens:

```
>>>a=True                    Attention à la majuscule
>>>b=False
>>>a or b
True
>>>a and b
False
```