Lycée Marcelin BERTHELOT
BCPST 1C – Mme MOREL

Variables et notion de type

Une **variable** est une donnée ou un résultat dont la valeur peut changer au cours d'une exécution ou à chaque exécution.
Elle correspond à un **emplacement mémoire** et peut être considérée comme un objet possédant
- une valeur,
- un identifiant (une adresse),
- un type.

De façon simplifiée, l'interpréteur crée un nouvel objet et le garde en mémoire. Chaque fois que nous taperons 128, Python retrouve sa valeur stockée à l'aide de son identifiant.
ATTENTION! L'identifiant change d'une session à l'autre.
Le **type** d'une variable est caractérisé par l'ensemble des valeurs qu'elle peut prendre et par les opérations permises sur ces valeurs.
En Python, on utilise essentiellement:

- entiers (relatifs) dont le type est int.
- réels (appelés flottants) dont le type est float.
- valeur de vérité: boolean.
- chaîne de caractères: str (pour string)

1 Entiers, type integer: int

∗ **Représentation des nombres en mémoire:**
1 octet= 8 bits, où bit= 0 ou 1.
L'ordinateur "compte" en base 2, explications:

Donc le chiffre 12 est représenté en mémoire par le mot 1100 (importance de l'ordre!!)
La plupart des machines actuelles utilise une représentation de 8 octets= 64 bits.

∗ **Limites de la représentation:**

Plus généralement, avec des mots en n bits, on peut représenter les relatifs entre -2^{n-1} et $2^{n-1}-1$.
Tous les calculs ne dépassant pas ces limites sont très rapides. Lorsqu'on dépasse cette limite, Python convertit (code et décode), mais nous n'avons pas à nous soucier de ce travail supplémentaire.

∗ **Opérations usuelles sur les entiers:**

Opération mathématique	instruction Python
addition	$+$
soustraction	$-$
multiplication	$*$
division	pas de division dans $\mathbb{Z}$
puissance	$**$

Remarques 1 :
(1) ATTENTION ! Pas de division dans $\mathbb{Z}$!!
(2) *Quotient et reste de la division euclidienne:*

	instruction Python
quotient	//
reste	%

(3) la division euclidienne par ..... renvoie un message d'erreur.
(4) Eviter la division euclidienne pour des nombres négatifs car Python ne donne pas tout à fait la même valeur qu'en mathématique.

Remarques 2 L'ordre des opérations en Python est le même que celui en mathématique:
(1) $*$, //, %, $+$, $-$: de gauche à droite.

(2) $**$: de droite à gauche.

2 Nombres réels, flottants, de type float

∗ **Limites de la représentation:**
Python ne manipule que des nombres avec un nombre fini de chiffres après la virgule. Par conséquent, tous les nombres dont l'écriture décimale admet une infinité de chiffres ne sont connus que par approximation.
De plus, des erreurs d'arrondi peuvent provenir d'une approximation dans la décomposition binaire d'un nombre...

Remarques 3 la virgule est un point (notation anglo-saxonne)

∗ **Notation scientifique:**
Pour 1×10^{-7}, Python écrit: $1\,e-07$, où e indique une puissance de 10 et non le nombre de Neper e.

∗ **Opérations usuelles sur les flottants:**

Opération mathématique	instruction Python
addition	$+$
soustraction	$-$
multiplication	$*$
division	$/$
puissance	$**$

Remarques 4 Apparaît maintenant la division. Elle peut s'appliquer aux entiers, mais attention, elle renvoie toujours un flottant (même quand le résultat est entier).

Remarques 5 Attention à l'ordre des opérations...

3 Choix et conversion de type

∗ **Choix:**
Il faut savoir choisir le type approprié en fonction du problème posé pour représenter des nombres.
Pour du calcul exact, il faut utiliser les entiers, une grandeur physique est du type float.

∗ **Conversion:**

- *int* → *float:* ajouter .0 ou utiliser la fonction `float()`

- *float* → *int:* la fonction `int()` tronque la partie décimale après la virgule et a donc pour résultat un entier.