

### ➤ Principaux types de données

En python, contrairement à la plupart des autres langages, il n'est pas nécessaire de définir un type de variable avant de l'utiliser, mais ce type existe et a son importance ! Syntaxe : `type(expression)`

- *int* Nombre entier de taille quelconque (contrairement aux autres langages !)
- *float* Nombre à virgule flottante compris dans  $[-10^{308}; -10^{-323}] \cup [10^{-323}; 10^{308}]$ .
- *complex* Nombre complexe (attention, *i* est noté *j* et un coefficient doit être présent devant *j*)
- *bool* Booléen, uniquement deux valeurs possibles : *True* ou *False* (attention à la majuscule)
- *str* Chaîne de caractères Unicode entre guillemets `"..."` ou `'...'`
- *tuple* Liste non modifiable de données de mêmes types ou variés, entre parenthèses (...)
- *list* Liste modifiable de données de mêmes types ou variés, entre crochets [...]
- *set* Ensemble non ordonné de données de mêmes types ou variés entre accolades {...}
- *dict* Dictionnaire, entre accolades {...} Exemple : `{'maths' : 7, 'phys' : 6}`

Attention : Pour le choix des noms de variables, ne pas commencer par un chiffre et différencier les minuscules des majuscules

### ➤ Conversions

Une expression peut être converti d'un type de données vers un autre pour différents usages, on peut utiliser le nom du type que l'on veut obtenir : Exemple : `int("12")` ou `str(5)`. Mais il y en a d'autres :

- `bin(entier)` Pour convertir un entier du système décimal vers le système binaire.
- `oct(entier)` Pour convertir un entier du système décimal vers le système octal.
- `hex(entier)` Pour convertir un entier du système décimal vers le système hexadécimal.
- `ord(caractère)` Pour obtenir l'entier Unicode d'un caractère placé entre guillemets.
- `chr(entier)` Pour obtenir le caractère codé par l'entier Unicode donné.
- `eval(expression)` Permet d'évaluer une expression numérique placée entre guillemets.
- `chaine.split()` Permet de convertir une chaîne de caractères en une liste en coupant par défaut à chaque espace, mais on peut préciser une autre sépateur...
- `expression.join(liste)` Permet de convertir une liste en un chaîne de caractère en intercalant une expression entre chaque valeur de la liste (attention, il faut une liste de *str*)

Remarque : Pour passer d'une base quelconque vers le système décimal, utiliser `int(chaine, base = entier)`

### ➤ Principaux opérateurs

<b>Opérateurs logiques</b> (agit sur les booléens)		<b>Opérateurs de comparaisons</b> (renvoie un booléen)		<b>Opérateurs mathématiques</b>	
or	ou inclusif	<	inférieur strictement	+	addition
and	et	>	supérieur strictement	-	soustraction
not	opposé	<=	inférieur ou égal	*	multiplication
		>=	supérieur ou égal	/	division (float)
	<b>Affectations</b>	==	test d'égalité	**	exposant
=	prend la valeur	!=	différent de	//	division entière
		in	est dans	%	reste de la division euclidienne (modulo)

Remarque : On peut aussi faire des affectations multiples : `a = b = 12` ou parallèles : `a, b = 12, 15`.