## **Python: Types et opérations**

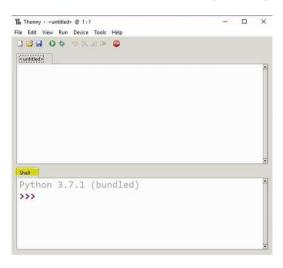
Les documents suivants sont construits de telle manière à ce que vous observiez des exemples simples commentés puis que vous réalisiez les exercices proposés au fur et à mesure.

Il est important de tester vous même les petits morceaux de code proposés dans les exemples avant de passer aux exercices

#### **I Thonny**

Dans cette introduction, vous utiliserez Python en mode interactif via la zone **Shell** du logiciel **Thonny**.

Cette zone permet de tester du code sans avoir à la sauvegarder au préalable.



Une invite de commande aussi appelée **prompt** et symbolisée par 3 chevrons `>>>` vous permet de tester directement du code qui sera immédiatement interprété après l'appui sur la touche `entrée`

# II Les types de base

L'instruction **type** permet d'obtenir le type d'un objet. Dans le shell, tester les types des différents objets suivants :

type(8)	type('Bonjour !') type(True)
type(-5) type(8.0)	type(11de)
type(-5.4)	

Quelles sont les différentes valeurs possibles pour un booléen ? (voir ce <u>lien</u>)

Compléter le tableau suivant avec les exemples précédemment faits :

Exemple ci-dessus	Туре	Nom
		entiers (integer)
		flottant
		chaîne de caractères (string)
		booléen

Le type détermine les propriétés de la valeur (par exemple les opérations qu'elle peut subir) et matérielles (par exemple, la façon dont elle est représentée en mémoire et la place qu'elle occupe)

#### **III Les nombres entiers**

Effectuer, via le shell, les calculs suivants et compléter au fur et à mesure le tableau cidessous :

60 - 7	10 % 3
2 * 3 + 4	11 % 3
(-2+3)*7-1	12 % 3
6/3	13 % 3
1/3	14 % 3
2 ** 2	2 < 7
2 ** 3	2 < 2
2 ** 4	5 >= 5
10 // 3	2 ** 3 ** 2
11 // 3	3/1/3
12 // 3	
13 // 3	
14 // 3	

Opérateur	Résultat obtenu	Type de l'objet obtenu
a ** b		
a // b		
a % b		
a < b		

En Python, les priorités opératoires sont-elles respectées : Oui / Non

En Python, la taille des entiers est limitée : Oui / Non

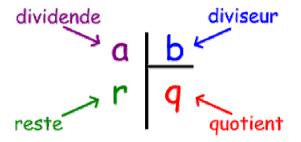
D'après les deux derniers exemples, dans quel sens python effectue-t-il les opérations si les opérateurs sont de même priorité ?

#### Règles de priorités :

Certains opérateurs sont évalués avant les autres. Dans l'ordre de priorité suivant :

- Mise à la puissance ;
- Reste dans la division euclidienne ;
- Multiplications et divisions entières (quotient dans la division euclidienne) ;
- Addition et soustraction

Sur les opérateurs de même priorité, celui qui est le plus à gauche est évalué en premier. Les parenthèses permettent de changer ces priorités.



#### IV Les flottants (ou nombres à virgule flottante)

Les nombres à virgule flottante sont des nombres comportant une partie entière et une partie décimale après la virgule marquée par un point (notation anglo-saxonne).

Entrez dans la console :

Lorsque la partie entière d'un nombre flottant devient trop grande ou que le nombre commence par une grande suite de zéros (y compris la partie entière) : le nombre est donnée en notation scientifique (comme sur vos calculatrices). Ici **e** signifie « *fois dix puissance* ».

Exemple: 1.35e4 correspond au nombre 13 500

On peut d'ailleurs directement écrire les nombres flottants sous cette forme :

Exemple: tapez: 6.02e21

Valeur exacte ou approximation?

Entrez: 0.20000000000000000000000458 0.000000000002458652541255612

0.0000000000245865254125561246545645645645645645645

*Attention* donc : la valeur retournée (et donc éventuellement stockée) par Python peut être une approximation du réel entré.

Enfin, il est intéressant de noter que Python convertit implicitement les entiers en flottants si nécessaire :

Exemple: entrez: 4 + 5.2

Remarque:

Chaque opérateur existe sous forme d'une fonction (cf. documentation officielle : Correspondance entre opérateurs et fonctions.

\_\_ mettre ici paragraphe sur valeur max et min possible avant infini

#### V Chaînes de caractères et opérations

Testez dans le shell les opérations suivantes :

```
'abc' + 'def'
'abc' * 2
'abc' ** 3
'abc' / 2
'abc' // 2
```

L'opérateur + appliqué à deux chaînes de caractères réalise leur **concaténation**.

Peut-on appliquer tous les opérateurs à deux chaînes de caractères ?

# VI Changer le type d'un objet

Dans certains cas, il est nécessaire de changer le type d'un objet.

## Convertir en un entier :

```
Testez:
int('457'):
int(3.145)
int(-2.25)
int('4.25')
int('a')
```

#### Convertir en un flottant :

```
float('135')
float('13.25')
float(45)
float('a')
```

## Convertir en une chaîne de caractères :

```
string(3.1456)
string(12)
```