
MPSI - ITC

Cours 1 - Les types de base en Python

1 - Contact Si vous avez des question entre les cours, vous pouvez m'écrire à l'adresse

Santiago.Bautista@ac-grenoble.fr

2 - Acronyme de la matière ITC est l'acronyme de Informatique Tronc Commun.

3 - Organisation des TP Pour les TP d'ITC, la classe est coupée en 3 groupes, qui passent sur des créneaux différents :

- Lundi de 10h à 11h en L312 avec M. Bautista
- Lundi de 11h à 12h en L312 avec M. Bautista
- Lundi de 12h à 13h en L312 avec M. Légaut

Chaque semaine, à l'issue du dernier créneau de TP, je laisserai dans la salle des MPSI une feuille qui décrit qui doit passer sur quel créneau. Vous pouvez échanger entre vous, de commun accord, et apporter les modifications correspondantes à la feuille, **à condition qu'il y aie au plus 17 personnes par créneau à la fin**. Avant le premier créneau de TP de la semaine, je ramasse la feuille, qui fera foi sur qui a TP quand.

Vous devez, **pro-activement et en avance**, repérer tout conflit avec une colle ou toute difficulté pour manger le midi, et faire les échanges nécessaires pour le résoudre. (Je reste disponible si besoin).

4 - EDT exceptionnel pour la semaine prochaine Pour que la randonnée d'intégration puisse commencer tôt le lundi 7 septembre, le créneau de TP de lundi 7 septembre 12h-13h avec M. Légaut est exceptionnellement reporté à mardi 8 septembre de 13h à 14h.

5 - Matériel nécessaire Pour les TP d'ITC, vous aurez besoin de :

- Une clef USB (pour enregistrer vos fichiers à la fin de l'heure)
- Un cahier (on verra pourquoi)
- De quoi écrire

I Python comme « calculatrice »

6 - contexte En ITC vous apprendrez à utiliser un **langage de programmation** appelé Python. Nous verrons progressivement pendant l'année qu'est-ce qu'un langage de programmation. Pour l'instant, nous allons faire comme si Python était une calculatrice, à laquelle on peut passer des **expressions** pour que Python les évalue vers des **valeurs**.

7 - Exemples En Python,

- l'expression `3 + 2` s'évalue vers la valeur `5`
- l'expression `1 / 2` s'évalue vers la valeur `0.5`
- l'expression `2 < 4` s'évalue vers la valeur `True`
- l'expression `"bon" + "jour"` s'évalue vers la valeur `"bonjour"`

II Différents types

II.1 Le type entier

8 - Utilité En python, le type entier permet de représenter les entiers relatifs

9 - Appellation Python En Python, le type entier s'appelle `int`.

10 - Opérations binaires entre entiers dont le résultat est entier Il existe plusieurs opérations binaires qu'on peut appliquer entre deux expressions entières pour obtenir une expression entière. Par exemple :

Symbole	Opération	Aussi appelée
+	Addition	
-	Soustraction	
*	Multiplication	
//	Quotient de la division euclidienne	Division entière
%	Reste de la division euclidienne	Modulo
**	Exponentiation	Puissance

11 - Exemples En Python,

- l'expression `1 // 2` s'évalue vers `0`
- l'expression `15 % 7` s'évalue vers `1`.
- l'expression `2**3` s'évalue vers `8`

II.2 Le type flottant

12 - Utilité En python, le type flottant permet de représenter **certains** nombres décimaux.

13 - Appellation Python En Python, le type flottant s'appelle `float`.

14 - Autre appellation Les flottants sont aussi appelés **nombres à virgule flottante**.

15 - Division Entre deux entiers, l'opérateur `/`, qui fait la division, produit un résultat de type flottant.

16 - Opérations entre flottants, à résultat flottant Entre flottants on peut utiliser, entre autres, les opérations `+`, `-`, `*` et `**`.

17 - Remarque : Précision limitée sur les flottants Vu que certains nombres ne sont pas représentables par des flottants, les résultats de calculs entre flottants sont approximatifs. Par exemple, en Python, `0.1 + 0.2` s'évalue vers `0.30000000000000004`.

18 - Remarque : non-associativité des flottants Les opérations entre flottants ne sont pas associatives. Par exemple,

`(1 + 2**-53) - 2**-53`

ne s'évalue pas vers la même valeur que

`1 + (2**-53 - 2**-53)`

II.3 Le type booléen

19 - Utilité En Python, le type booléen est utilisé pour représenter les valeurs de vérité (**vrai** ou **faux**).

20 - Appellation Python En Python, le type booléen s'appelle `bool`. Il n'y a que deux valeurs Python ayant ce type :

- la valeur `True` pour représenter **vrai**
- la valeur `False` pour représenter **faux**

21 - Origine du nom Le nom du type booléen est en l'honneur de George Bool (1815 - 1864).

22 - Résultat des comparaisons Le résultat d'une comparaison est de type booléen, quelque soit le type des éléments comparés. En python, il y a plusieurs opérateurs binaires permettant de faire des comparaisons

Opérateur	Opération
<code>==</code>	Test d'égalité
<code><</code>	« strictement plus petit que »
<code><=</code>	« plus petit ou égal »
<code>></code>	« strictement plus grand que »
<code>>=</code>	« plus grand ou égal »
<code>!=</code>	« différent de »

23 - Un seul opérateur unaire sur les booléens : la négation La négation booléenne, qui en Python se note `not`, est telle que

- l'expression `not True` s'évalue vers `False`
- l'expression `not False` s'évalue vers `True`

24 - Notion de table de vérité pour les opérateurs booléens binaires Étant donné le très faible nombre de valeurs différentes dans les booléens, les opérateurs sur les booléens sont souvent décrits en donnant exhaustivement les résultats pour toutes les paires de valeurs possibles

a	b	Conjonction (« et »)	Disjonction (« ou »)
		<code>a and b</code>	<code>a or b</code>
<code>True</code>	<code>True</code>	<code>True</code>	<code>True</code>
<code>True</code>	<code>False</code>	<code>False</code>	<code>True</code>
<code>False</code>	<code>True</code>	<code>False</code>	<code>True</code>
<code>False</code>	<code>False</code>	<code>False</code>	<code>False</code>

II.4 Exercices

25 - Exercice Vers quelle valeur s'évalue l'expression suivante, et quel est son type ?

```
(1 < 2) == (5 == 5)
```

Justifiez votre réponse.

26 - Exercice Vers quelle valeur s'évalue l'expression suivante, et quel est son type ?

```
((2 < 1) or (not(2 < 2))) and (not(2 >= 2) or (-1 < 3))
```

Vous détaillerez les étapes de calcul vous permettant de répondre.

II.5 Le type chaîne de caractères

27 - Utilité Ce type permet de représenter les textes et les bouts de phrase, entre autres.

28 - Appellation Python En Python, ce type est noté `str` qui est l'abréviation du mot anglais *string* (au sens *fil*, *succession*, *chaîne*).

29 - Écrire une chaîne de caractères en Python En Python, on peut délimiter les chaînes de caractères par des simples guillemets, des doubles guillemets ou des triplets de doubles guillemets. Par exemple, `'Bonjour'`, `"Bonjour"` et `"""Bonjour"""` sont 3 façons d'écrire la même chaîne.

30 - Concaténation de deux chaînes de caractères On peut concaténer deux chaînes de caractères avec l'opérateur `+`.

31 - Longueur d'une chaîne de caractères La *longueur* d'une chaîne de caractères est le nombre de caractères qu'elle contient. Par exemple `"Qu'il fait beau!"` est une chaîne de longueur 16. La fonction `len` de Python donne la longueur d'une chaîne de caractères. Par exemple l'expression `len("Qu'il fait beau!")` s'évalue vers `16`.

Le type liste

32 - Utilité Le type liste permet de représenter des *collections* de plusieurs valeurs.

33 - Appellation Python En Python, le type liste s'appelle `list`

34 - Écrire une liste Pour écrire explicitement une liste, on sépare ses différents éléments par des virgules, et on met le tout entre crochets carrés.

35 - Exemple La valeur `[2, 3, 5, 7, 11, 13, 17, 19]` est une liste.

36 - Longueur d'une liste On appelle *longueur* d'une liste son nombre d'éléments ; et on peut utiliser `len` pour l'obtenir.

Par exemple, l'expression `len([2, 3, 5, 7, 11, 13, 17, 19])` s'évalue vers `8`.