Les types natifs

Avec Python (et en programmation en général), il existe plusieurs types natifs qui nous permettent de représenter toutes sortes de données. Parmi ces types natifs, les chaînes de caractères, les nombres et les booléens sont à la base de presque tout.

Les chaînes de caractères

Les chaînes de caractères permettent de représenter du texte. Elles sont définies par des guillemets simples ou doubles.

Attention avec les guillemets, vous pouvez avoir de drôle de surprises si vous utilisez des apostrophes dans votre texte.

Pour régler ce problème, vous pouvez alterner entre les guillemets simples ou doubles, ou utiliser un anti-slash pour 'échapper' le guillemet qui pose problème :

`"Bonjour, je m'appelle Thibault"` → on alterne les guillemets doubles et simples (l'apostrophe)

`'Bonjour, je m\'appelle Thibault'` → on utilise l'anti-slash pour 'échapper' le guillemet simple (l'apostrophe)

Les nombres

Les nombres sont définis en deux catégories principales : les nombres entiers et les nombres décimaux (il existe d'autres types de nombres mais qui sont réservés à des cas très précis que nous ne verrons donc pas pour l'instant).

Ces nombres permettent de représenter tout type de données numériques, par exemple :

- Le solde de votre compte en banque
- Votre âge
- Le nombre d'habitants à Paris (ou à Marseille 👀)
- Les 5 étoiles que vous allez donner à cette formation incroyable (n'est-ce pas 🥹 ?)

Les booléens

Les booléens sont probablement le type natif qui vous laisse le plus sur votre faim pour le moment. En effet, en l'état ils ne semblent pas très utiles. Pourtant, c'est la base de toute l'informatique moderne ! Avec les booléens, on peut représenter l'univers au complet (ou presque...).

En effet, en informatique on n'aime pas les compromis 😎

Les choses sont beaucoup plus simples que dans la vraie vie : les ordinateurs voient les choses en noir et blanc.

On utilisera beaucoup les booléens par la suite car ils vont permettre à nos scripts de prendre des décisions !

En fonction du résultat d'une expression, qui retournera `True` ou `False`, vous pourrez ainsi orienter votre programme dans la direction souhaitée.

Les constructeurs de types natifs

Python est un langage à la syntaxe extrêmement épurée. C'est ce qui fait sa force et la raison pour laquelle il est tant apprécié !

Quand on définit un booléen, un nombre ou une chaîne de caractères, pas besoin donc de spécifier le type de la variable à créer. Python est suffisamment intelligent pour le déduire lui-même (grâce par exemple aux guillemets pour les chaînes de caractères).

Il existe cependant des classes qui nous permettent de créer ces objets :

- `str`
- `int`
- `float`
- `bool`

Ces classes ne sont pas très utiles pour créer des objets.

Mais elles sont très pratiques pour *convertir* un objet d'un type à un autre !

Raison pour laquelle vous les retrouverez souvent sous le nom de *'fonction de conversion'*.

Ainsi, on peut convertir une chaîne de caractères en nombre entier ou vice-versa :

- `str(5)` → `"5"`
- `int("5")` → `5`

C'est une opération que l'on a souvent besoin de faire car il arrive que l'on récupère des variables dans un type qui n'est pas celui que l'on souhaite.

Par exemple, si vous récupérer un nombre à l'intérieur d'un fichier, il est possible qu'il vous soit retourné sous forme de chaîne de caractères. Mais pour l'additionner avec un autre nombre, vous serez obligé de convertir cette chaîne de caractères en nombre entier, ce que vous pourrez faire avec la fonction `int`.