Décimal, binaire et hexadécimal avec Python

Il faut bien distinguer le type d'une variable en Python et les systèmes de numération en bases 10, 2 et 16.
Types Python utilisés ici :

- `int` (pour « integer », nombre entier relatif, comme 42, 0, -666, ...)
- `str` (pour « string », chaine de caractères, "hello", "42", "", ...)

Systèmes de numération utilisés ici :

- décimal, système positionel en base 10 : 123 est cent-vingt-trois
- binaire, en base deux : $(101)_2$ désigne 5 ;
- hexadécimal, en base seize : $(A7)_{16}$ désigne 167.

int littéraux

Si on ne tape que des chiffres pour une valeur dans une expression, Python évalue ceci en un nombre entier écrit en base 10. Si on préfixe par `0b` ou `0B`, Python essaiera de lire un nombre entier écrit en binaire. Idem avec `0x` ou `0X` pour l'hexadécimal. Python présentera la valeur en décimal. Exemples :

```
>>> 123              >>> 0b101            >>> 0xA7
123                  5                    167
>>> 0b2
SyntaxError: invalid digit '2' in binary literal
```

On peut donc déja faire des conversions sans utiliser de fonction !

str littéraux

Python offre plusieurs possibilités pour taper une chaine de caractères : guillemets doubles (`"`), guillements simples (l'apostrophe droite `'`) ou les triples guillemets (`"""` ou `'''`) pour pouvoir taper des guillemets et des retours à la ligne.

Conversions en Python

La fonction `int` accepte un ou deux arguments.
Le premier est la chaine de caractère à décoder,
le second est la base à utiliser (par ex 2, 10 ou 16, et 10 si rien n'est indiqué).

Les fonctions `bin` et `hex` acceptent un seul argument de type entier (`int`).
Le résultat renvoyé par ces fonctions est une chaine de caractères.

Exemples de conversions avec les fonctions int, bin et hex

Repérez bien les types des valeurs Python et la base utilisée pour les écrire, il y a de tout ! Entiers littéraux dans les trois bases, chaines de caractères littérales dans les trois bases.

```
>>> bin(5)          >>> int("101", 2)    >>> hex(0b1111)
'0b101'             5                    '0xf'

>>> hex(167)        >>> int("A7", 16)    >>> bin(0xf)
'0xa7'              167                  '0b1111'

                    >>> int("0b11", 2)
                    3

                    >>> int("0xA", 16)
                    10
```

Quelques remarques :

- classement de gauche à droite : du plus facile au plus difficile à mettre en place avec Python. Pour les conversions « à la main », c'est le contraire !
- `int` ignorera le préfixe `0b` si la base est 2, idem avec `0x` en base 16.

Dernière remarque

Attention à ne pas confondre le mot « décimal » utilisé pour décrire la base 10, et le mot « décimal » qui décrit un nombre qui peut s'écrire exactement en utilisant un nombre fini de chiffres après la virgule en écriture décimale positionnelle.