Les types de base en Python

① MÉMO

Les quatre types fondamentaux

`int` Entier relatif, précision illimitée en Python : `42`, `-7`, `0`.
`float` Nombre à virgule flottante (norme IEEE 754, précision finie) : `3.14`, `-0.5`, `1e-3`.
`str` Chaîne de caractères, entre guillemets simples ou doubles : `'hello'`, `"bonjour"`.
`bool` Booléen, deux valeurs : `True` et `False`.

Conversions de type (transtypage)

On convertit un type en un autre avec les fonctions éponymes :
- `int("42")` renvoie l'entier `42` ;
- `float(3)` renvoie `3.0` ;
- `str(42)` renvoie la chaîne `"42"` ;
- `bool(0)` renvoie `False` (tout nombre non nul donne `True`).

Opérateurs arithmétiques

`+`, `-`, `*` Addition, soustraction, multiplication.
`/` Division réelle : `7 / 2` donne `3.5`.
`//` Division entière (quotient) : `7 // 2` donne `3`.
`%` Modulo (reste) : `7 % 2` donne `1`.
`**` Puissance : `2 ** 10` donne `1024`.

Pièges fréquents

- `0.1 + 0.2` ne vaut pas exactement `0.3` (erreur d'arrondi des flottants) ;
- `"3" + "4"` donne `"34"` (concaténation, pas addition) ;
- `type(7 / 1)` renvoie `float` : la division `/` renvoie toujours un flottant ;
- `int("3.5")` provoque une erreur : il faut d'abord passer par `float`.

② EXEMPLES

Programme 1 · Vérifier le type d'une variable

```
x = 42
print(type(x))        # <class 'int'>

y = 42.0
print(type(y))        # <class 'float'>
```

Programme 2 · Division entière et modulo

```python
heures = 137 // 60    # 2 heures
minutes = 137 % 60    # 17 minutes
print(f"{heures}h{minutes}min")  # 2h17min
```

Programme 3 · Piège des flottants

```python
print(0.1 + 0.2 == 0.3)          # False !
print(abs(0.1 + 0.2 - 0.3) < 1e-9)  # True (comparaison correcte)
```

Programme 4 · Transtypage

```python
age = input("Âge : ")   # input renvoie toujours un str
age = int(age)           # conversion nécessaire pour calculer
print(f"Dans 10 ans : {age + 10}")
```

Programme 5 · Booléens et valeurs "falsy"

```python
print(bool(0))       # False
print(bool(""))      # False  (chaîne vide)
print(bool([]))      # False  (liste vide)
print(bool("0"))     # True   (chaîne non vide !)
```

③ EXERCICES

Exercice 1 *Types et transtypage* **(3 points)** Donner le type et la valeur de chaque expression :

1. `17 // 5`
2. `17 / 5`
3. `"17" + "5"`
4. `int("17") + int("5")`
5. `bool("")`
6. `float(True)`

Exercice 2 *Extraction de chiffres* **(3 points)** Écrire un programme qui demande un entier positif à trois chiffres et affiche séparément le chiffre des centaines, des dizaines et des unités.

Exemple : pour `427`, afficher `4`, `2`, `7`.

Exercice 3 *Conversion de durée* **(2 points)** Écrire une fonction `convertir(secondes)` qui prend un nombre entier de secondes et renvoie un tuple `(heures, minutes, secondes)`.

Exemple : `convertir(3725)` renvoie `(1, 2, 5)`.

Exercice 4 *Flottants et comparaison* **(2 points)**

1. Expliquer pourquoi `0.1 + 0.2 == 0.3` renvoie `False` en Python.

2. Écrire une fonction `sont_egaux(a, b, epsilon=1e-9)` qui compare deux flottants avec une tolérance.

SOLUTIONS DES EXERCICES

Corrigé de l'exercice 1.

1. `3` de type `int` (quotient de la division entière).
2. `3.4` de type `float` (la division `/` renvoie toujours un flottant).
3. `"175"` de type `str` (concaténation de deux chaînes).
4. `22` de type `int` (on additionne deux entiers après conversion).
5. `False` de type `bool` (une chaîne vide est considérée comme fausse).
6. `1.0` de type `float` (`True` est converti en `1` puis en flottant).

Corrigé de l'exercice 2.

```
n = int(input("Entier à 3 chiffres : "))
centaines = n // 100
dizaines = (n % 100) // 10
unites = n % 10
print(f"Centaines : {centaines}")
print(f"Dizaines : {dizaines}")
print(f"Unités : {unites}")
```

Principe : la division entière par 100 isole les centaines. Le modulo 100 donne les deux derniers chiffres, puis la division entière par 10 isole les dizaines. Le modulo 10 donne les unités.

Corrigé de l'exercice 3.

```
def convertir(secondes):
    h = secondes // 3600
    m = (secondes % 3600) // 60
    s = secondes % 60
    return (h, m, s)
```

Vérification : `convertir(3725)` donne :

- `h = 3725 // 3600 = 1`;
- `m = 125 // 60 = 2`;
- `s = 125 % 60 = 5`, soit `(1, 2, 5)`.

On vérifie : $1 \times 3600 + 2 \times 60 + 5 = 3725$. ✓

Corrigé de l'exercice 4.

1. Les nombres `0.1` et `0.2` ne sont pas représentables exactement en binaire (norme IEEE 754). Leur somme vaut `0.30000000000000004`, ce qui diffère de la représentation de `0.3`.
2.

```
def sont_egaux(a, b, epsilon=1e-9):
    return abs(a - b) < epsilon
```

   On compare la valeur absolue de la différence à un seuil ε suffisamment petit.