Les types de base en Python

RÉSUMÉ

Quand on programme, on manipule des **données** : des nombres, du texte, des réponses vrai/faux. Python a besoin de savoir quel *type* de donnée il manipule pour décider comment la traiter : additionner deux nombres n'est pas la même chose que coller deux mots bout à bout. Connaître les types de base, c'est la première étape pour éviter les erreurs les plus courantes et écrire des programmes qui fonctionnent.

① MÉMO

Les quatre types principaux

`int` Nombre entier : `42`, `-7`, `0`.
`float` Nombre à virgule : `3.14`, `-0.5`.
`str` Texte, entre guillemets : `'hello'`, `"bonjour"`.
`bool` Booléen, c'est-à-dire une valeur qui est soit `True` (vrai) soit `False` (faux). Indispensable pour les conditions (`if`) et les comparaisons.
Les booléens sont le résultat de toute comparaison : `3 > 2` donne `True`, `5 == 7` donne `False`.

Conversions de type

On change le type d'une valeur avec :
- `int("42")` donne l'entier `42` ;
- `float(3)` donne `3.0` ;
- `str(42)` donne le texte `"42"`.

Opérateurs arithmétiques

`+`, `-`, `*` Addition, soustraction, multiplication.
`/` Division : `7 / 2` donne `3.5`.
`//` Division entière : `7 // 2` donne `3`.
`%` Modulo (reste) : `7 % 2` donne `1`.
`**` Puissance : `2 ** 3` donne `8`.

Pièges fréquents

- `"3" + "4"` donne `"34"` (concaténation de texte, pas addition).
- `type(7 / 1)` donne `float` : la division `/` donne toujours un nombre à virgule.
- `input()` renvoie toujours du texte : il faut convertir avec `int()` ou `float()` pour calculer.

② EXEMPLES

Programme 1 · Vérifier le type d'une variable

```python
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 · Saisie clavier et conversion

```python
age = input("Quel âge as-tu ? ")   # age est un str
age = int(age)                      # conversion en entier
print(f"Dans 10 ans tu auras {age + 10} ans")
```

③ EXERCICES

Exercice 1 *Types et conversions* Donner le type et la valeur de chaque expression :

1. `17 // 5`
2. `17 / 5`
3. `"17" + "5"`
4. `int("17") + int("5")`

Exercice 2 *Extraction de chiffres* 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* On dispose d'une durée exprimée en secondes. Écrire un programme qui affiche cette durée en heures, minutes et secondes.

Créer trois variables `h`, `m`, `s` pour stocker respectivement le nombre d'heures, de minutes et de secondes, puis afficher le résultat.

Exemple. Pour `3725` secondes, afficher `1h 2min 5s`.

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 `/` donne toujours un nombre à virgule).
3. `"175"` de type `str` (concaténation de deux textes).
4. `22` de type `int` (on additionne deux entiers après conversion).

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.

```
total = 3725
h = total // 3600
m = (total % 3600) // 60
s = total % 60
print(f"{h}h {m}min {s}s")
```

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