

Types de base

Fiche de synthèse

① LE TYPE D'UNE VALEUR

Le **type** d'une valeur détermine les opérations permises : on ne multiplie pas un mot par un mot, on ne met pas un nombre en majuscules. La fonction `type` donne le type d'une valeur ; `print(type(42))` affiche `<class 'int'>`.

Type	Nom	Exemples	Conversion
entier	<code>int</code>	42, -7	<code>int()</code>
flottant	<code>float</code>	3.14, 7.0	<code>float()</code>
chaîne	<code>str</code>	'Bonjour', "42"	<code>str()</code>
booléen	<code>bool</code>	True, False	<code>bool()</code>

C'est la donnée qui est typée, pas la variable : une même variable peut désigner successivement un entier puis une chaîne.

② ENTIERS

Un **entier** est un nombre sans partie décimale. En Python, sa taille est illimitée : `2 ** 100` se calcule exactement.

```
print(10 / 2)      # 5.0 : toujours un flottant
print(10 // 2)     # 5 : division entière
print(7 // 2)      # 3 : quotient de 7 par 2
```

③ FLOTTANTS

Un **flottant** est un nombre à virgule, écrit avec un **point** : 3.14. Le point suffit : 3.0 est un flottant. Un calcul qui mêle un entier et un flottant donne un flottant : `3 + 4.0` vaut 7.0.

Les flottants sont rangés sur 64 bits, avec environ quinze chiffres significatifs : leurs valeurs sont souvent approchées.

```
print(0.1 + 0.2)    # 0.30000000000000004
print(0.1 + 0.2 == 0.3) # False
print(abs(0.1 + 0.2 - 0.3) < 1e-10) # True
```

On ne compare jamais deux flottants avec `==` : on vérifie que leur écart est très petit. `abs(x)` est la valeur absolue de `x`, et `1e-10` s'écrit en notation scientifique pour 10^{-10} .

④ CHAÎNES DE CARACTÈRES

Une **chaîne de caractères** s'écrit entre apostrophes ou entre guillemets : "L'informatique" contient une apostrophe, on l'entoure donc de guillemets.

```
mot = "Python"
print(len(mot))          # 6 caractères
print("Ada" + " " + "Lovelace") # assemblage
print("ha" * 3)          # hahaha
print(42 + 1)            # 43
print("42" + "1")        # 421
```

Assembler deux chaînes s'appelle la **concaténation**. On ne peut pas additionner une chaîne et un nombre : "J'ai " + 17 provoque une erreur `TypeError`. On écrit "J'ai " + `str(17)`, ou plus simplement la f-string `f"J'ai {age} ans"`.

⑤ BOOLÉENS ET NONE

Un **booléen** vaut `True` ou `False`, avec une majuscule. C'est le résultat d'une comparaison : `5 > 3` vaut `True`.

`None` représente l'absence de valeur ; son type est `NoneType`.

0, 0.0, la chaîne vide '' et `None` sont considérés comme faux par `bool` ; toute autre valeur est considérée comme vraie, même la chaîne "False", qui n'est pas vide.

⑥ CONVERSIONS

```
print(int("15"))      # 15
print(float("3.14"))  # 3.14
print(str(42))         # la chaîne '42'
print(int(3.7))        # 3
print(int(-2.9))       # -2
print(round(3.7))      # 4 : arrondi
print(round(3.14159, 2)) # 3.14
```

`int("abc")` et `int("3.14")` provoquent une erreur `ValueError`.

📌 **Calculer avec une saisie.** `input` renvoie toujours une chaîne : on la convertit dès la saisie.

```
age = int(input("Ton âge ? "))
print(f"Dans 10 ans : {age + 10}")
```

📌 À retenir.

- Quatre types de base : `int`, `float`, `str`, `bool`.
- `/` donne toujours un flottant, `//` la division entière.
- On ne compare jamais deux flottants avec `==`.
- Les guillemets font la chaîne : "42" + "1" vaut "421".
- `int` supprime la partie décimale sans arrondir ; `round` arrondit.