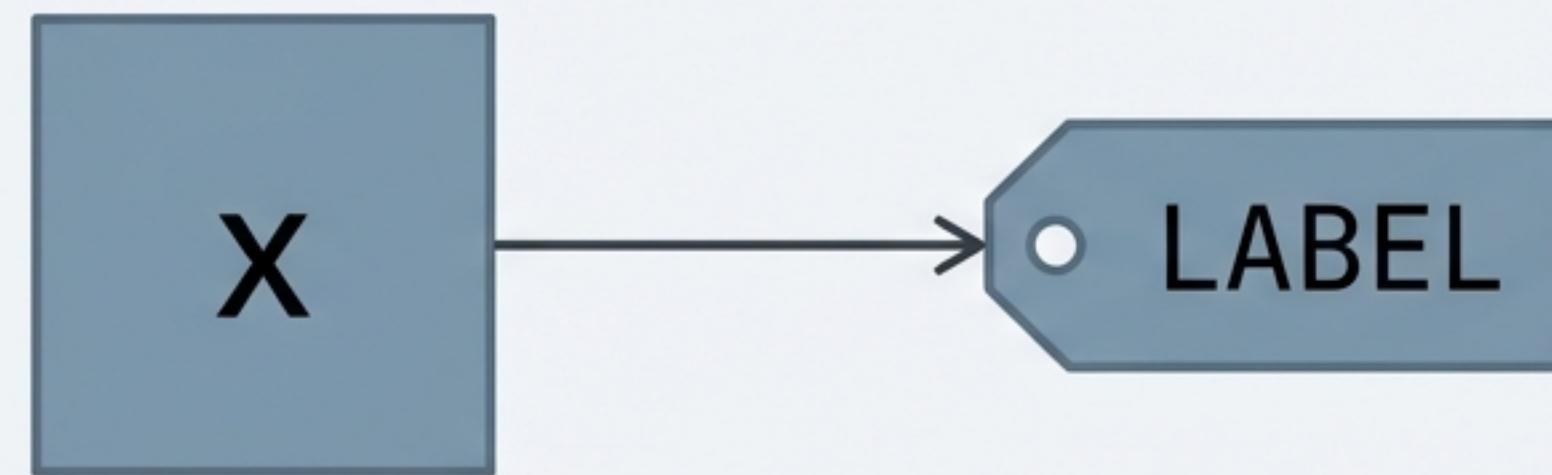


Les Bases de Python : Variables et Types

Du calcul simple à la programmation reproductible



1. Les Variables

Comment sauvegarder des valeurs (le stockage).

2. Les Types de Données

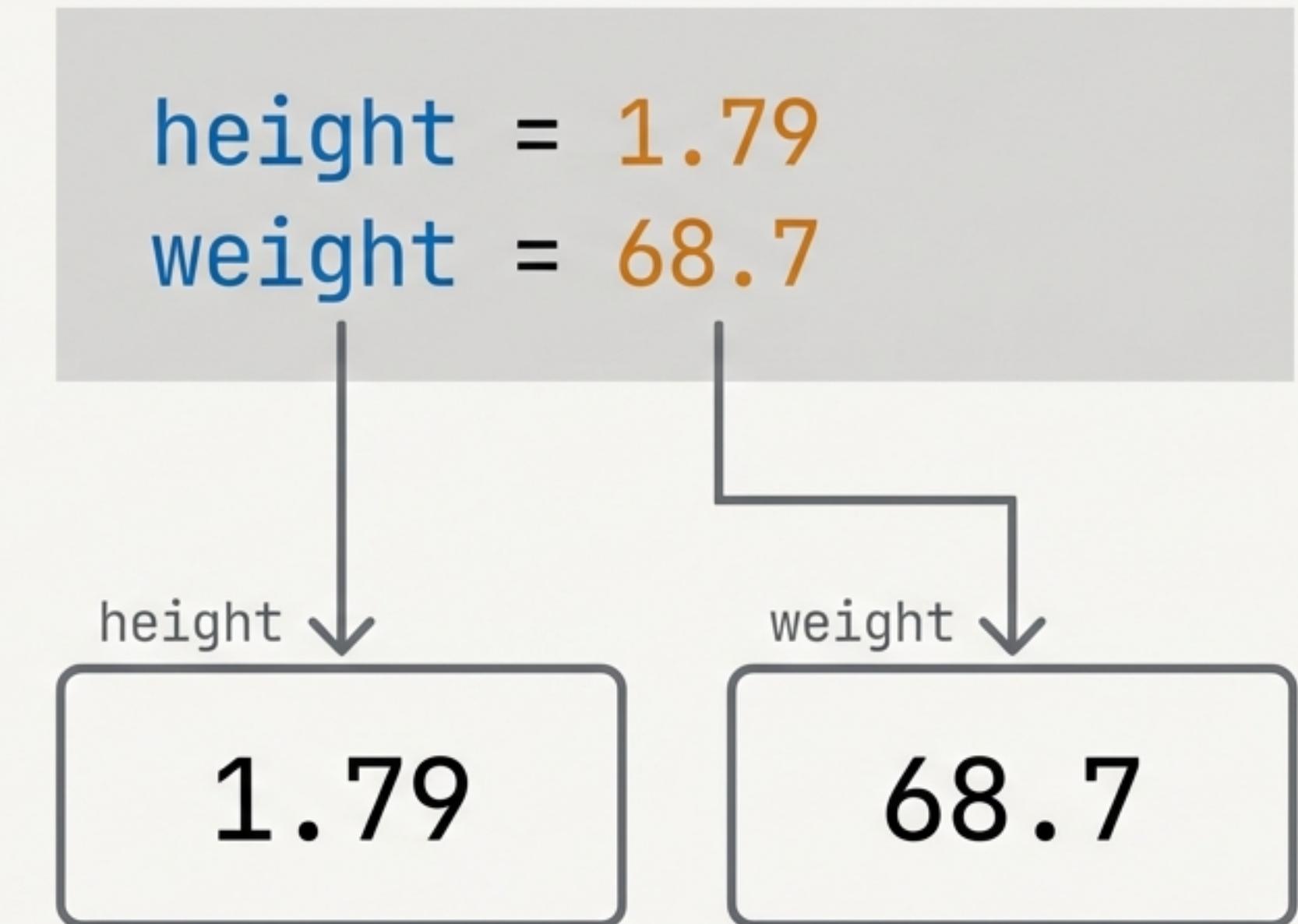
Comment Python interprète ces valeurs (le contexte).

Python est un excellent calculateur, mais pour résoudre des problèmes complexes, il faut savoir stocker l'information.

Stocker une valeur dans une variable

Pour conserver une valeur au fil du code, on la définit via un nom de variable.

- L'assignation se fait avec le signe égal (=).
- Les noms sont sensibles à la casse (majuscules/minuscules).



Ici, `height` stocke la valeur 1.79 et `weight` stocke 68.7.

Utiliser les variables pour le calcul

Une fois déclarée, l'appel du nom de la variable permet à Python de récupérer sa valeur pour effectuer des calculs.

Exemple : Calcul de l'Indice de Masse Corporelle (IMC / BMI).

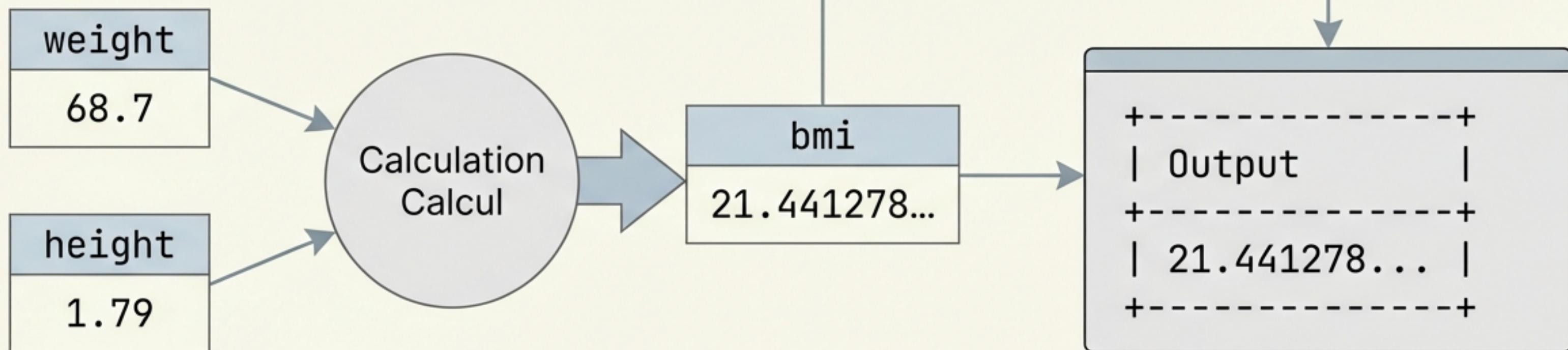
Formule : Poids / Taille²

```
68.7 / 1.79 ** 2  
# Ou en utilisant les variables :  
weight / height ** 2
```

```
+-----+  
| Résultat |  
+-----+  
| 21.441278... |  
+-----+
```

Capturer le résultat

Au lieu d'afficher simplement le résultat, nous pouvons le stocker dans une nouvelle variable nommée bmi. Cette variable contient désormais la valeur calculée.

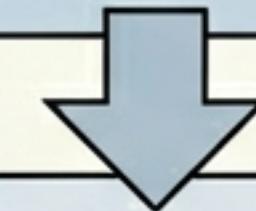


La puissance de la reproductibilité

L'utilisation de variables rend le code reproductible. Si les données d'entrée changent, il suffit de modifier la déclaration initiale et de relancer le script.

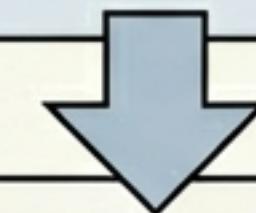
1. Modifier la déclaration

```
weight = 70.0
```



2. Relancer le calcul

```
bmi = weight / height ** 2
```



3. Mise à jour automatique

```
bmi
```

Inspection des types de données : Le Float

En Python, chaque valeur possède un type spécifique. La fonction `type()` permet de l'identifier.

Le résultat de notre calcul IMC est un float.

```
type(bmi)
```



Float (Nombre à virgule flottante) :
Représente un nombre réel
contenant une partie entière et une
partie fractionnaire.

```
+-----+  
| Output |  
+-----+  
| <float> |  
+-----+
```

Les nombres entiers (Integer)

Pour les nombres sans partie fractionnaire, Python utilise le type int.

```
exemple_entier = 4  
type(exemple_entier)
```

Float

4.0

Int

4

```
+-----+  
| Output |  
+-----+  
| <int> |  
+-----+
```

Représenter du texte : Les Strings

Pour manipuler du texte, Python utilise le type str (chaîne de caractères).

- On peut utiliser des guillemets doubles ("") ou simples ('').

```
x = "body mass index"  
y = 'this works too'  
type(x)
```

```
+-----+  
| Output |  
+-----+  
| <str> |  
+-----+
```

x



La logique binaire : Les Booléens

Le type bool ne peut prendre que deux valeurs : True (Vrai) ou False (Faux).

- Analogie : Oui / Non.
- Utilité : Essentiel pour effectuer des opérations de filtrage sur les données.

```
z = True  
type(z)
```

-----+	
	Output
-----+	
	<bool>
-----+	



Le comportement dépend du type

Les opérateurs agissent différemment selon le type de données. L'opérateur + additionne les nombres mais 'colle' les textes.

Addition d'entiers (Math)

```
# Addition d'entiers  
2 + 3
```



-> Résultat : 5

Addition de strings (Concaténation)

```
# Addition de strings  
"ab" + "cd"
```



-> Résultat : "abcd"

Python adapte le comportement du code au contexte du type de donnée.

Récapitulatif des types de données

Type	Description	Exemple
float	Nombre réel (à virgule)	1.79, 68.7
int	Nombre entier	4, 100
str	Texte (String)	"Python", 'Code'
bool	Logique (Vrai/Faux)	True, False

Maîtriser ces types est la première étape pour structurer efficacement vos données.