Labo d’introduction à l’informatique

Yann Thorimbert

Semaine 1
Variables et expressions
Fonctions et opérateurs
Input et output

Qu'est-ce qu'une variable ?

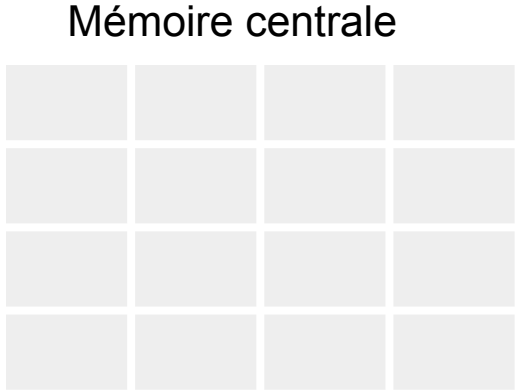
Une variable permet à la machine de mémoriser une valeur.

Qu’est-ce qu’une variable ?

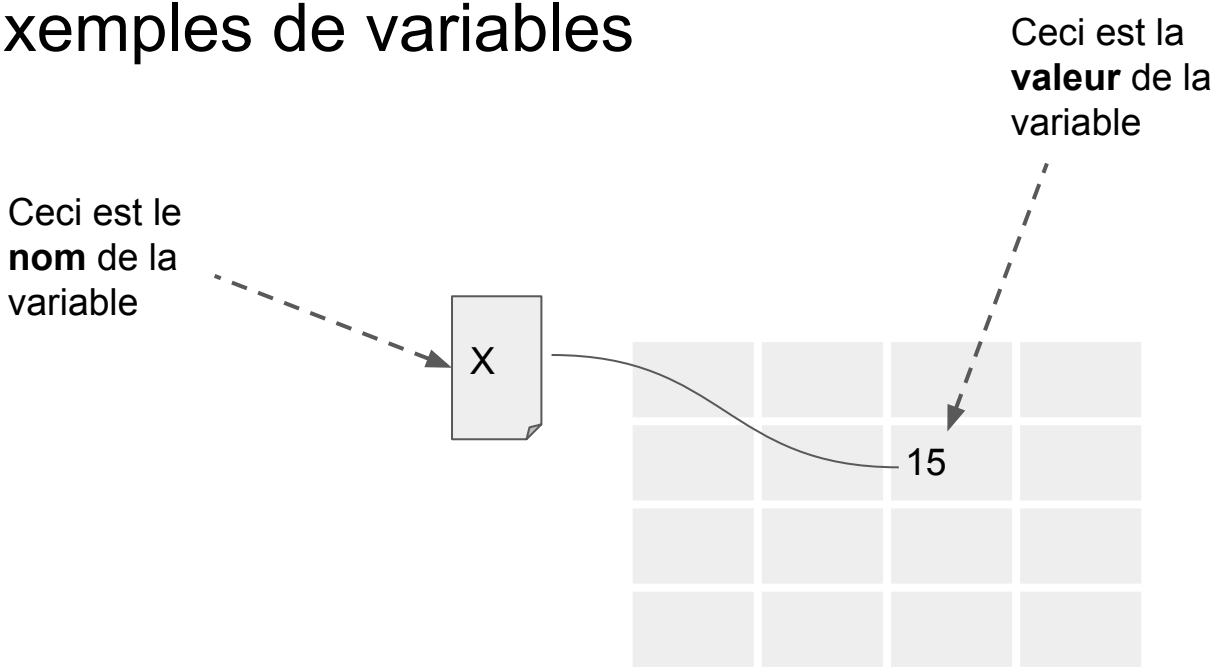
Durant l’exécution d’une application, les données utiles sont stockées dans la mémoire centrale (mémoire vive).

La mémoire centrale peut être imaginée comme un ensemble de cases dans lesquelles on peut stocker des valeurs.

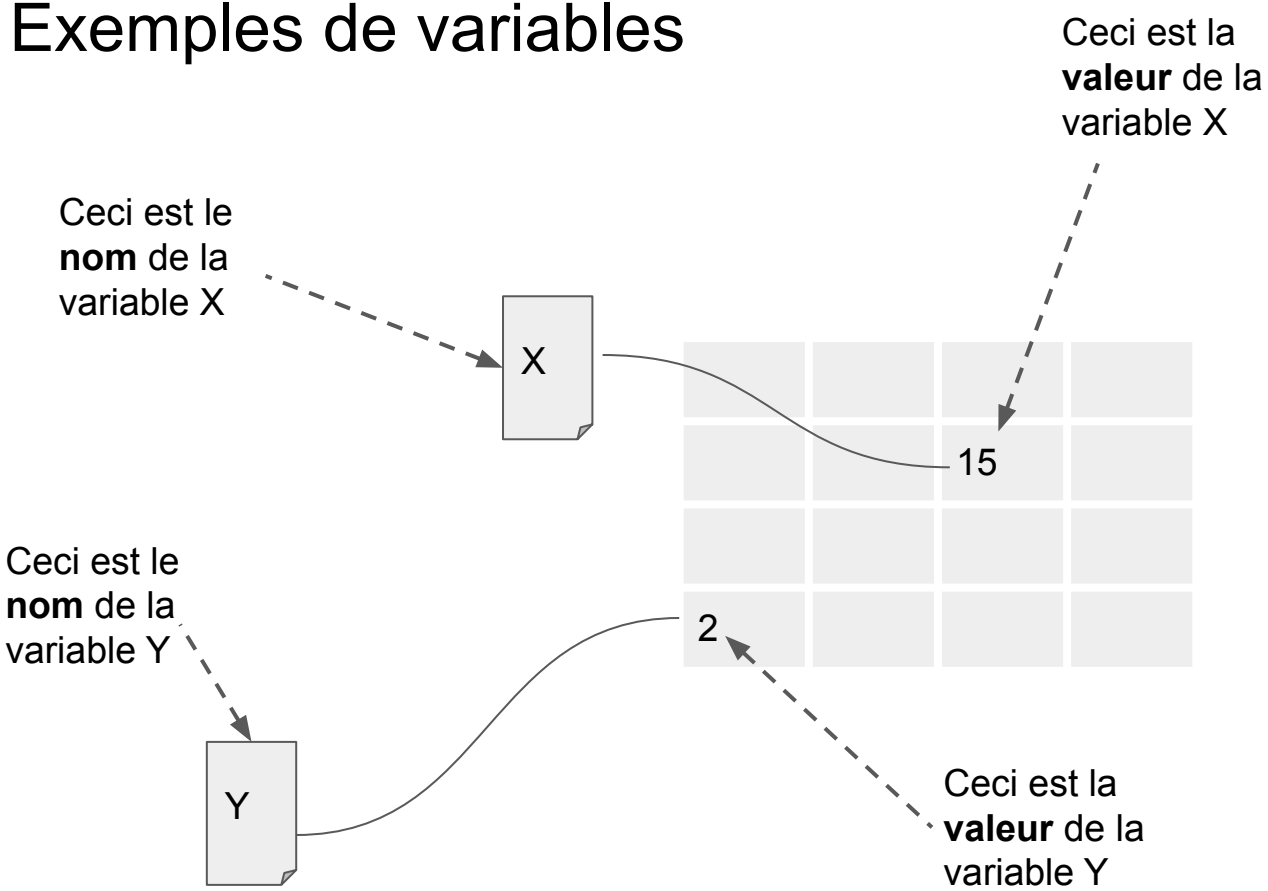
Lorsqu’on a besoin d’une valeur, on demande à l’ordinateur d’aller la chercher (la lire) dans sa mémoire.



Exemples de variables



Exemples de variables



Comment peut-on nommer les variables ?

Dans la plupart des langages de programmation, le nom d'une variable ne peut pas **commencer** par un chiffre.

De même, on ne met pas d'**espace** vide dans le nom d'une variable.

Dans ce cours, on prend l'habitude de toujours commencer le nom d'une variable par une minuscule, et d'utiliser soit l'écriture "camel case", soit des underscores pour les espaces. Exemples d'utilisation du camel case et des underscores :

ceciEstUnNomDeVariable ceci_est_un_nom_de_variable

nomDuPersonnage

ageDuPersonnage

Comment peut-on nommer les variables ?

Règles de nommage strictes (contrôlées par l'interpréteur Python) :

- Ne pas commencer par un chiffre.
- Ne pas inclure d'autres caractères que des lettres majuscules et minuscules, les chiffres et le caractère "_".

Règles de nommage conventionnelles :

- Éviter les accents.
- Éviter les majuscules au début du nom des variables.
- Éviter les noms trop peu précis ou vagues.

Si la variable est destinée à garder la même valeur sans jamais changer :

- On peut l'écrire tout en majuscule.

Les variables prédéfinies et les mots-clés

- Tout langage de programmation possède des mots-clés, qui ont une **signification déjà décidée** au sein du langage et sont en général **réservés**. Exemples en Python: `if, for, ...`

Déclaration de variable

- À **gauche** de l'égalité se situe le **nom** de la variable.
- À **droite** de l'égalité se situe la **valeur** de la variable.
- Exemple A:

```
x  = 3
```

- Exemple B:

```
x = 8.4
y = 0.2
z = x + y
```

Valeur issue de l'évaluation de l'**expression** `x + y`

Que peut-on mettre "dans" une variable ?

La valeur d'une variable peut être un nombre, un texte, ou encore d'autres quantités dont on parlera plus loin dans le cours.

Le **type** de valeur que contient la variable est une donnée importante.

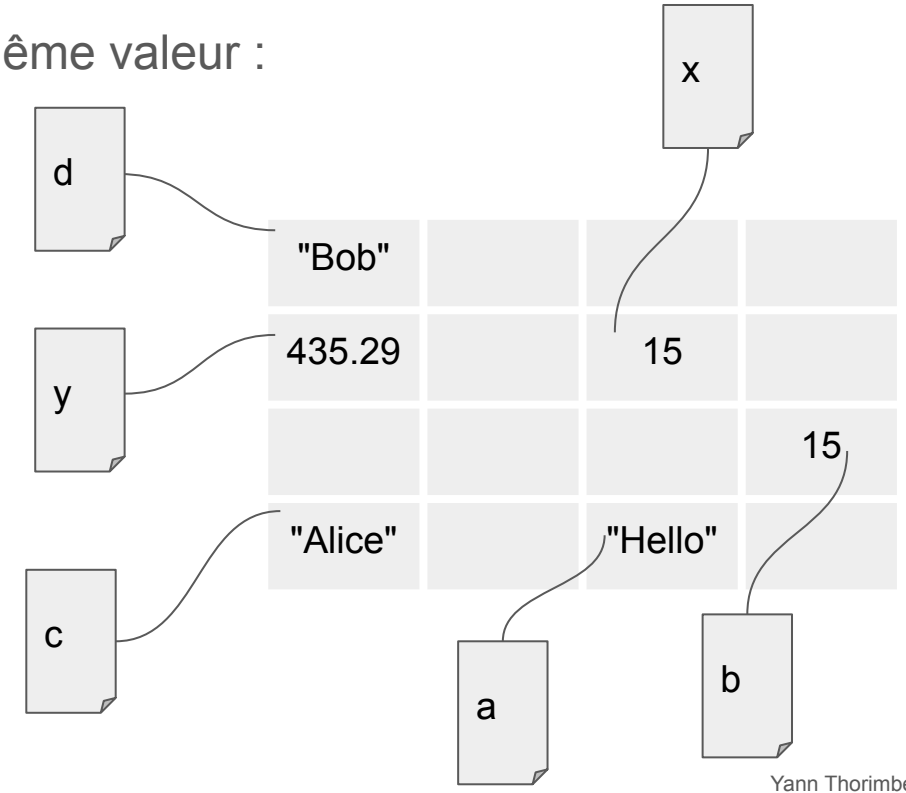
```
a = "Bonjour à tous"
b = 15
```

Dans le code ci-dessus, a et b ne sont visiblement pas du même type.

Que peut-on mettre "dans" une variable ? (suite)

Plusieurs variables peuvent avoir la même valeur :

```
a = "Hello"
b = 15
c = "Alice"
d = "Bob"
x = 15
y = 435.29
```

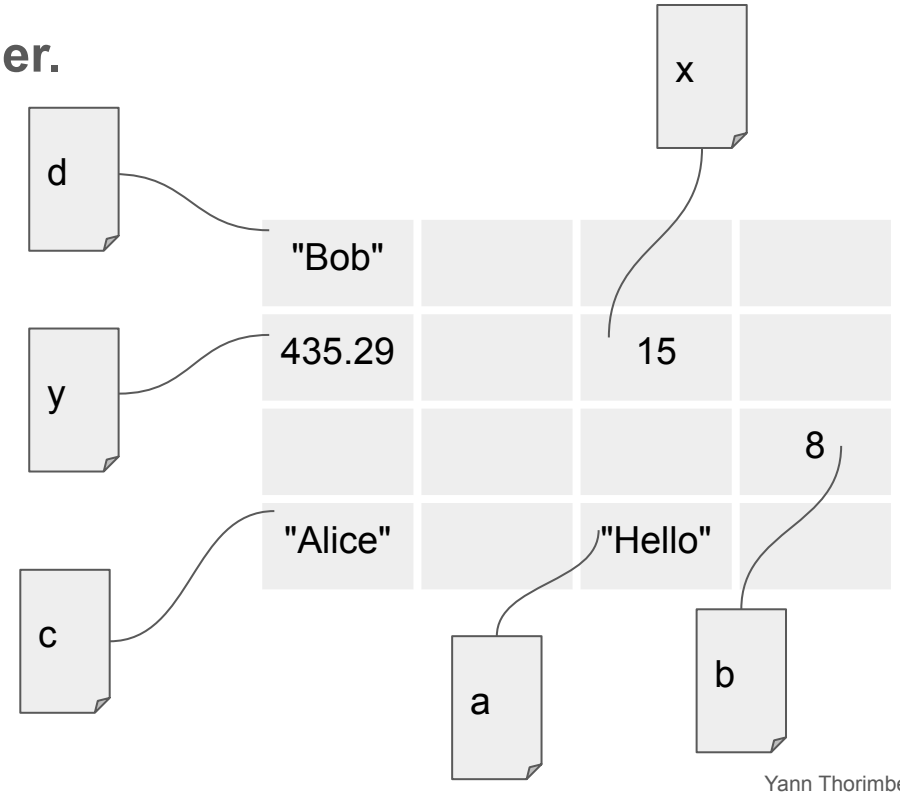


Que peut-on mettre "dans" une variable ? (suite)

La valeur d'une variable peut **changer.**

Ici, la variable b est modifiée dans la dernière ligne du code.

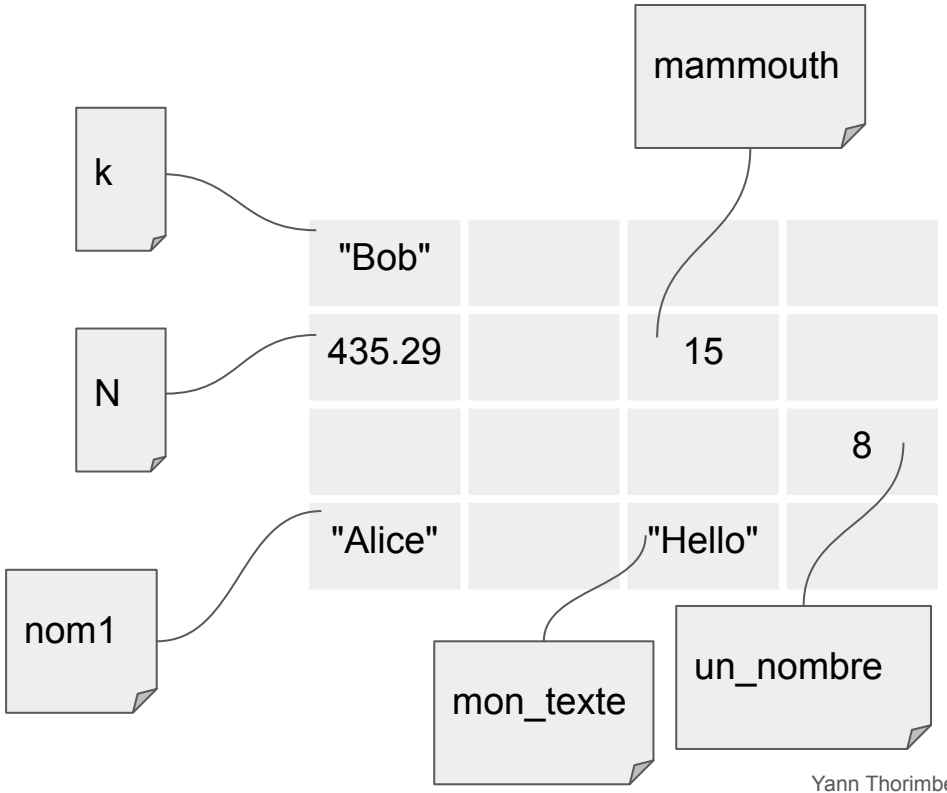
```
a = "Hello"
b = 15
c = "Alice"
d = "Bob"
x = 15
y = 435.29
b = 8
```



Que peut-on mettre "dans" une variable ? (suite)

C’est le programmeur qui **décide** du nom des variables.

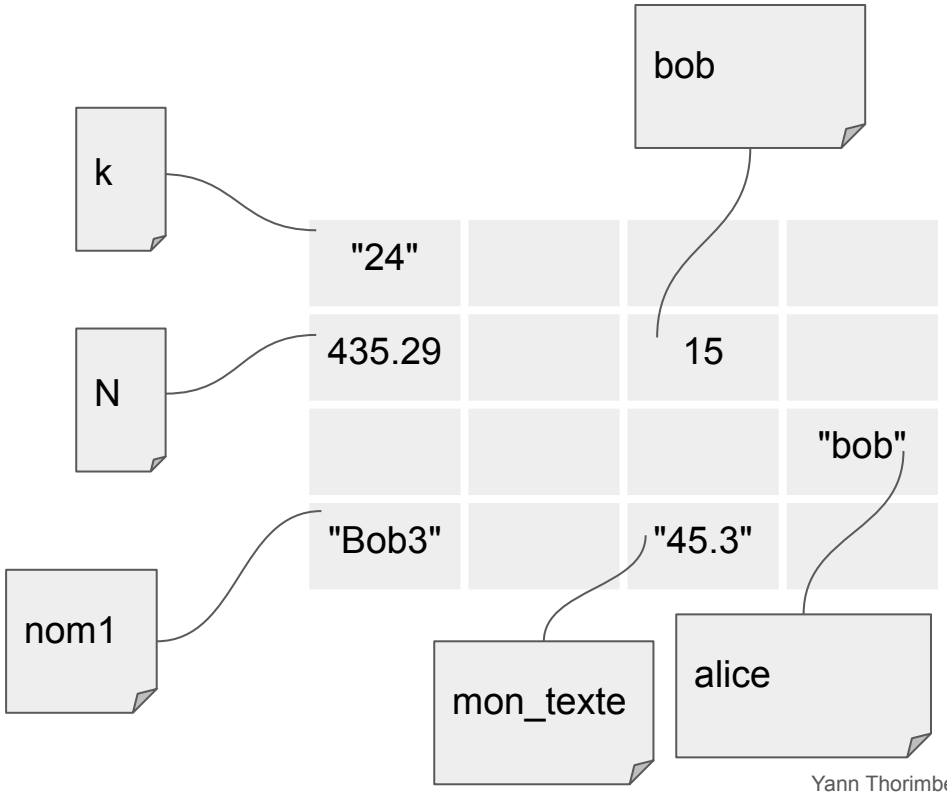
Dans l’exemple ci-contre, toutes les variables n’ont pas des noms très explicites, ce qui est à éviter !



Que peut-on mettre "dans" une variable ? (suite)

Attention, un texte peut contenir des chiffres !

Texte : **chaîne** de caractères (*string* en anglais)



Exemples de types de variables

Type d'information	Alias en Python	Exemples d'assignation
Nombre entier relatif	`int`	`x=-5, y=12000, z=500_000`
Nombre à virgule	`float`	`x=-2.76, y=float(3)`
Texte	`str`	`x="Ciao!", y=":)", z="32", t=" "`
Booléen	`bool`	`x=True, y=False`

Attention aux déductions automatiques de Python !
Comme on le verra plus loin, la division de deux entiers ne donne pas un entier.
Cf. cours théorique pour les aspects techniques derrière ces types.

Types de variables

- Comprendre le type de variable que l'on utilise est capital pour programmer correctement.
- En étant permissif (typage "dynamique"), Python peut vous inciter à prendre de mauvaises habitudes.
- Au fil du cours, nous découvrirons d'autres types.
- **Avertissement** : il est attendu de vous que vous puissiez en tout temps expliquer le type de toutes les variables que vous déclarez dans vos codes. Utilisez l'IA de façon responsable pour vous-même, ne devenez pas dépendants.

Obtenir le type d’une variable

- La fonction `type(x)` permet d’obtenir le type de `x`.
  Exemples :

```
a = -8
print(type(a)) # int
b = "Salut"
print(class(b)) # string
```

- Quand on programme, il est impératif de connaître les types afin de comprendre les erreurs qui s’affichent.

Variables (rappel)

- Le nom de la variable est indiqué à gauche du signe d'égalité.
- La valeur de la variable est indiquée à droite du signe d'égalité.
- Exemples :

```
mon_nombre = 3.14
un_autre = -12
x = 2^4
y = mon_nombre + un_autre
z = sqrt(36)
```

Expressions

Désigne ce qui peut être **évalué** pour en obtenir une valeur.

- Une variable est une expression.
- Un littéral est une expression qui est la représentation textuelle d'une valeur.
  Exemples : `3`, `"salut"`
- Une opération qui combine des variables et des littéraux pour retourner une valeur est elle-même une expression.
  Exemple : `y = x + 8`

  Valeur issue de l'évaluation de l'expression `x + 8`,
  où l'opérateur d'addition est utilisé en combinant la valeur de la variable `x` et d'un littéral.

Assignations

- Le signe d'égalité ne s'entend pas comme en mathématiques !
- Le signe d'égalité s'entend comme "devient…", "se transforme en…", "prend la valeur de…". C'est une **assignation**.
- *Exemple A :*

```
x = 3
x = x + 2
print(x) # ceci affiche 5
```

- *Exemple B :*

```
y = 4
y = 1
print(y) # ceci affiche 1
```

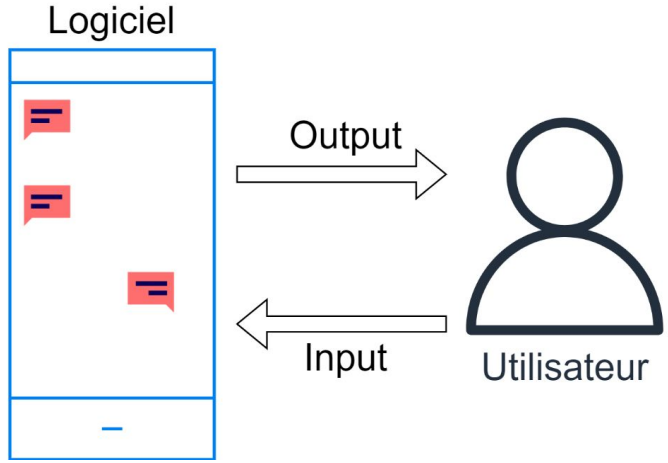
Input, output et rôle de l'utilisateur

- Lorsqu'on **utilise** une application ou un logiciel, le code du programmes nous est caché.

- L'utilisateur :
  - Reçoit des informations (output).

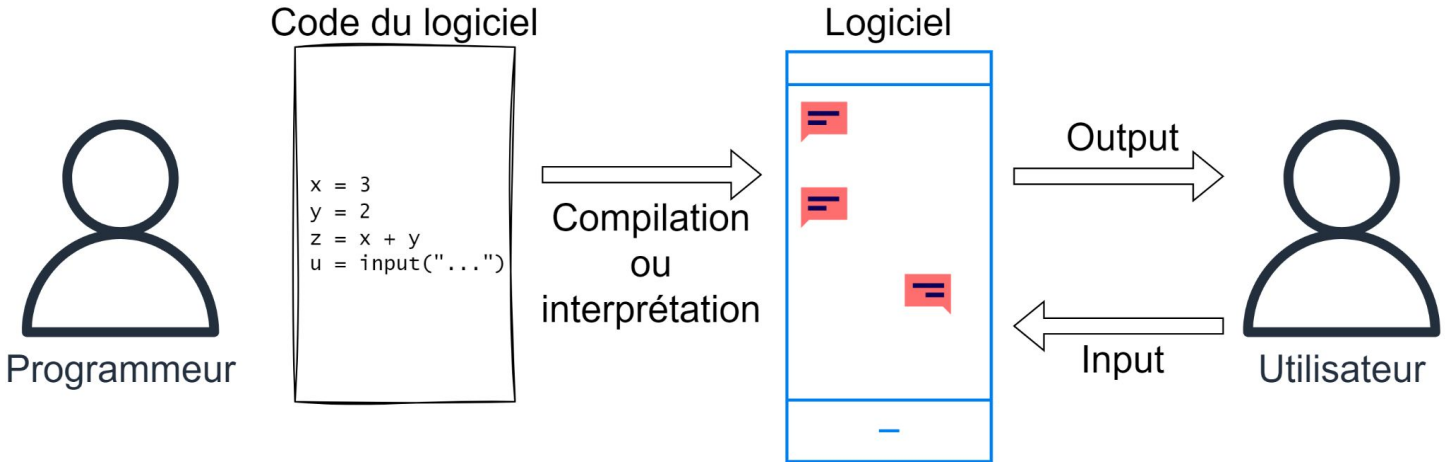
    Par exemple : lire un message affiché.
  - Envoie des informations (input).

    Par exemple : écrire un message.



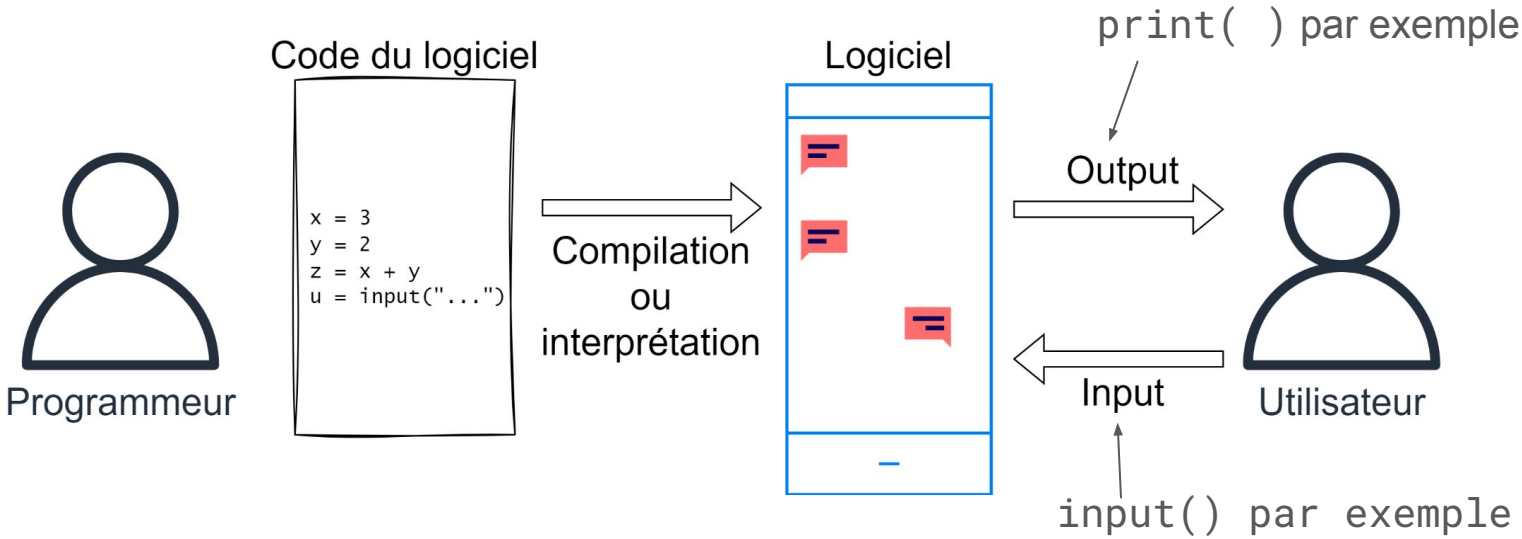
Input, output et rôle de l'utilisateur

Lorsqu'on **crée** un logiciel, on doit faire **semblant** d'être l'utilisateur lorsqu'on **teste** notre programme.



Input, output et rôle de l'utilisateur

Lorsqu'on **crée** un logiciel, on doit faire **semblant** d'être l'utilisateur lorsqu'on **teste** notre programme.



Input, output et rôle de l'utilisateur

- Certaines variables se voient assigner une valeur qui **dépend de l'input** de l'utilisateur et qui n'est pas connue au moment de l'écriture du programme.
  Exemple : `nom = input("Entrez votre nom:")`

- D'autres variables ont une valeur décidée uniquement par le programmeur.
  Exemple : `age_minimum = 18`

- Dans tous les cas, le type et le nom de la variable sont fixés par le programmeur.

Exemples d'input en Python

```
name = input("Quel est votre nom ?")

age = int(input("Quel est votre âge ?"))

taille = float(int("Quel est votre taille ?"))
```

Et l'output ?

- En Python l'output textuel vers la console est en général effectué via la fonction `print()`.

- En règle générale, les fonctions d'input et d'output dépendent des **périphériques** utilisés. Dans ce cours, on ne considère que les inputs et outputs affichés par l'écran au sein de la **console** ("Command Window").

Exemple d'affichage de valeurs avec `print()`

```
a = "Bonjour"
b = 12
c = 2.718
```

(coloration fantaisiste pour des raisons pédagogiques)

Exemple d'affichage de valeurs avec `print()`

```
a = "Bonjour"
b = 12
c = 2.718

print("Mon texte:", a)
print("Mon entier:", b)
print("Mon nb à virgule:", c)
print("Les trois à la suite:", b, c, a)
```

Note sur le formattage

Depuis la version 3.6 de Python, il est possible d'utiliser les f-strings pour afficher des variables dans des chaînes de caractères :

```
print(f"La variable b vaut {b}.")
```

Le formattage des valeurs affichées (par exemple chiffres après la virgule) sera vu plus loin dans les TPs.

Fonctions simples

- "Bout de code" qui peut être réutilisé depuis d'autres scripts.
- Effectue une suite d'instructions sur les paramètres d'entrée (**arguments**).
- Peut **retourner** une valeur.
- Exemple de **déclaration** de fonction :

```
def cube(x):
    return x * x * x
```

- Exemple **d'utilisation** de fonction :
  cube(3) # retourne 27

Fonctions sans valeur de retour

- Exemple : fonction uniquement destinée à afficher un graphique, une image.
- Dans ne nombreux langages, on parle de **procédure** dans ce cas.
- En Python :

```
def dire_bonjour(x):
    print("Bonjour", x)
```

Un piège classique

```
x = input("Entrez un nombre:")

print("Voici son triple:", x*3)
```

Ce code génère-t-il une erreur ?
Fait-il ce que l'on peut raisonnablement attendre ?

Un piège classique

```
x = input("Entrez un nombre:")

print("Voici son triple:", x*3)
```

Ce code génère-t-il une erreur ? **Non !**
Fait-il ce que l'on peut raisonnablement attendre ? **Non plus !**

Ce qu'il faut retenir : être attentif aux **types**.

Pas de piège ici

```
x = 4.5
y = str(x)
print("Salut " + y) #Ceci affiche "Salut 4.5"
```

Ce code ne génère pas d'erreur, on peut convertir un nombre en texte grâce à la fonction prédéfinie `str()`.

Un autre piège classique

```
str = "bonjour"
x = 4.5
y = str(x)
```

Ce code génère-t-il une erreur ?

Un autre piège classique

```
str = "bonjour"
x = 4.5
y = str(x)
```

Ce code génère-t-il une erreur ? **Oui !**

Ce qu'il faut retenir : le nom d'une fonction de conversion de type n'est pas un mot-clé réservé.

Opérations sur les variables

On peut manipuler arithmétiquement les nombres et assigner le résultat d'opérations au sein de variables:

```
nb_minutes = 3
nb_sec = nb_minutes * 60
print(nb_minutes, " min = ", nb_sec , " sec.")
```

Opérateurs utilisés dans ce cours

Nom de l'opération	Opérateur Python	Exemple Python
Addition	`+`	`3 + 4 #vaut 7`
Soustraction	`-`	`3 - 4 #vaut -1`
Multiplication	`*`	`3 * 4 #vaut 12`
Division	`/`	`3 / 4 #vaut 0.75`
Modulo	`%`	`3 % 4 #vaut 3`
Puissance	`**`	`3 ** 4 #vaut 81`