

# SNT – 01 – Bases de python

Alexis Lamiable

Septembre 2025

# Plan

Types de données

Variables

Structures de contrôle

# Plan

Types de données

Variables

Structures de contrôle

# Les types de données

Pour nous, écrire **12** désigne à la fois le nombre 12 ou un mot composé des deux caractères “1” et “2”. Par contre, le mot “bonjour” est forcément du texte, pas un nombre.

En informatique, les données ont un **type** qui est important.

Quelques types courants :

- ▶ Les nombres entiers — type **int** (*integer*)
- ▶ Les nombres à virgule — type **float** (*floating point*)
- ▶ Le texte — type **str** (*string*)
- ▶ Les valeurs de vérité – type **bool** (*boolean*)

## Quelques exemples

Avec la *fonction* python `type`, on peut savoir le type de quelque chose :

```
>>> type(3)
<class 'int'>
```

```
>>> type(3.14)
<class 'float'>
```

```
>>> type("bonjour")
<class 'str'>
```

```
>>> type(True)
<class 'bool'>
```

## Quelques opérations de base

On peut additionner, soustraire, coller ("concaténer") d'une manière assez logique :

```
>>> 3 + 16
```

```
19
```

```
>>> 3 + 6.5
```

```
9.5
```

```
>>> 3 + 6.0
```

```
9.0
```

```
>>> "bonjour" + "à tous"
```

```
'bonjourà tous'
```

```
>>> 3 + "bonjour"
```

```
Traceback (most recent call last):
```

```
File "<stdin>", line 1, in <module>
```

```
TypeError: unsupported operand type(s) for +: 'int'  
and 'str'
```

## Quelques opérations de base

Il y a des choses plus inattendues :

```
>>> "ab" * 6  
'abababababab'
```

```
>>> 3 // 4  
0
```

```
>>> 3 % 4  
3
```

## Conversions de type

On peut souvent changer le type de quelque chose :

```
>>> float(3)
```

```
3.0
```

```
>>> int(3.14)
```

```
3
```

```
>>> int("12")
```

```
12
```

```
>>> int("bonjour")
```

```
Traceback (most recent call last):
```

```
File "<stdin>", line 1, in <module>
```

```
ValueError: invalid literal for int() with  
base 10: 'bonjour'
```

## Un dernier type courant : la liste

```
>>> [1, 4, 3, 2, 16]
```

```
>>> type([1, 4, 3, 2, 16])  
<class 'list'>
```

```
>>> [3, 8] + 6
```

```
Traceback (most recent call last):
```

```
  File "<stdin>", line 1, in <module>
```

```
TypeError: can only concatenate list (not "int") to list
```

```
>>> [3, 8] + [6]  
[3, 8, 6]
```

# Plan

Types de données

Variables

Structures de contrôle

# Variables

Une variable, c'est **un nom** qu'on **associe** à une zone dans la mémoire.

Vous pouvez aussi voir ça (pour l'instant) comme une boîte dans laquelle on peut stocker une valeur.

Mettre une valeur dans la boîte s'appelle l'**affectation**.

```
>>> a = 12
>>> type(a)
<class 'int'>
```

```
>>> a = "toto"
>>> type(a)
<class 'str'>
```

## Plus sur l'affectation

Attention, l'affectation se produit au moment où c'est écrit, mais ne crée pas de lien permanent entre deux variables.

```
>>> x = 3
```

```
>>> b = x
```

```
>>> x = 4
```

Que vaut b?

```
>>> a = 3
```

```
>>> b = 6
```

Que dois-je écrire pour inverser les valeurs de a et b?

# Plan

Types de données

Variables

Structures de contrôle

## Les boucles – le for

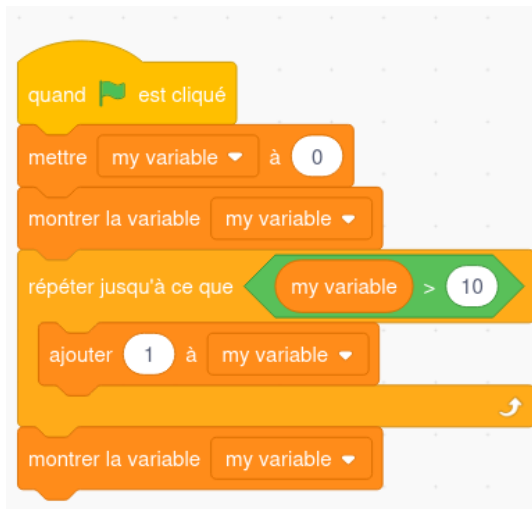


## Les boucles – le for

```
var = 0
print(var)
for i in range(0, 10):
    var = var + 1
print(var)
```



## Les boucles – le while




The image shows a Scratch script on a grid background. It starts with a yellow 'when clicked' block. This is followed by an orange 'set my variable to 0' block. Then another orange 'show variable my variable' block. A green 'repeat until' loop block follows, with the condition 'my variable > 10'. Inside the loop is an orange 'add 1 to my variable' block. After the loop, there is a final orange 'show variable my variable' block.

```
when green flag clicked
  set my variable to 0
  show variable my variable
  repeat until (my variable > 10)
    add 1 to my variable
  show variable my variable
```

## Les boucles – le while

```
var = 0
print(var)
while var <= 10
    var = var + 1
print(var)
```



The image shows a Scratch script on a grid background. It starts with a yellow 'when green flag clicked' block. This is followed by an orange 'set my variable to 0' block. Then, an orange 'show variable my variable' block. A green 'repeat until' loop block follows, with the condition 'my variable > 10'. Inside the loop is an orange 'add 1 to my variable' block. The loop ends with an orange 'show variable my variable' block.

## Les choix conditionnels – le if



A Scratch script illustrating conditional logic. It starts with a yellow 'when green flag clicked' block. This is followed by an orange 'set my variable to 0' block. Then a blue 'ask Enter a number and wait' block. Next is another orange 'set my variable to réponse' block. The core logic is an orange 'if' block with a green condition 'my variable > 50'. Inside the 'if' block, there is a blue 'go to random position' block. Below the 'if' block is an orange 'else' block containing a blue 'set heading to 90' block.

```
when green flag clicked
  set my variable to 0
  ask Enter a number and wait
  set my variable to réponse
  if my variable > 50 then
    go to random position
  else
    set heading to 90
```

## Les choix conditionnels – le if

```
var = 0
x = int( input("Entrez un chiffre: ") )
if x > 50:
    ...
else:
    ...
```