



SNT Python 1 - Types de données

1 - Thonny

HTML et **CSS** sont des langages

Scratch est un langage

Python est un langage



SNT Python 1 - Types de données

1 - Thonny

HTML et **CSS** sont des langages de description.

Scratch est un langage de programmation par bloc.

Python est un “ ” par lignes de code.

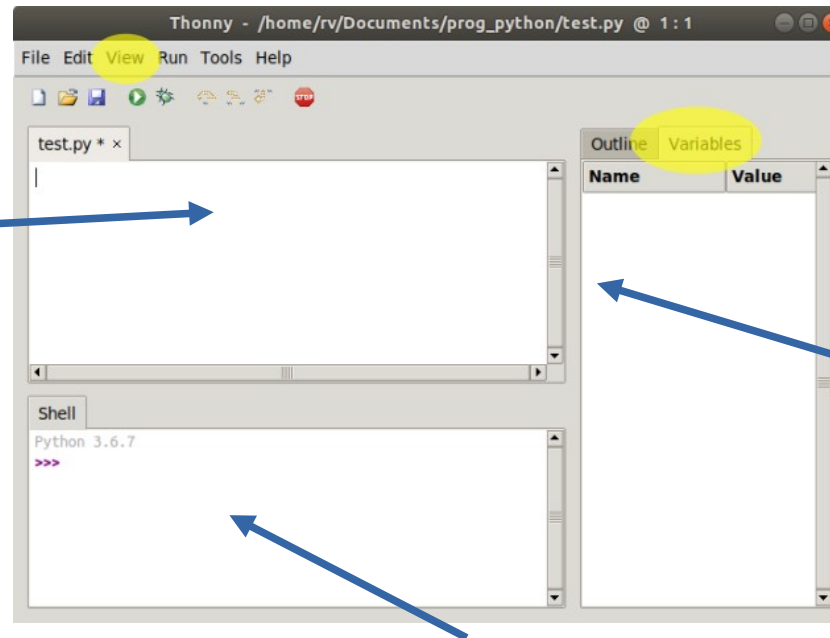
Thonny permet de programmer en Python et d'interagir en direct avec l'utilisateur (dans la zone console avec les >>>).

Pour créer des programmes Python, nous utiliserons **Thonny** : gratuit et libre (on peut lire son code-source).

Zone de programmation

Permet de mettre du code enregistré en mémoire.

On active le programme en appuyant sur la flèche VERTE



Onglet des Variables

Permet d'observer directement le contenu en mémoire.

Zone interactive >>> (console ou shell)

On tape des instructions qui s'exécutent tout de suite mais une seule fois

2 - Le type int : integer / entier

Type de données correspondant aux entiers.

Voici les opérateurs classiques : + - * /

>>> 7 + 2 **>>> 7 * 2** **>>> 7 / 2** **>>> 10 ** 3**

Les opérateurs de la division euclidienne :

→ Quotient **>>> 13 // 5** **>>> 13 // 6**

→ Reste **>>> 13 % 5** **>>> 13 % 6**

13 | 5

|

13 | 6

|

2 - Le type int : integer / entier

Type de données correspondant aux entiers.

Voici les opérateurs classiques : + - * / et exposant avec **

```
>>> 7 + 2      >>> 7 * 2      >>> 7 / 2      >>> 10 ** 3
9              14              3.5             1000 = 103 = 10*10*10
```

Les opérateurs de la division euclidienne :

```
→ Quotient    >>> 13 // 5      >>> 13 // 6
2              2
→ Reste       >>> 13 % 5      >>> 13 % 6
3              1
```

$$\begin{array}{r|l} 13 & 5 \\ -10 & \\ \hline 3 & 2 \end{array}$$

$$\begin{array}{r|l} 13 & 6 \\ -12 & \\ \hline 1 & 2 \end{array}$$

Priorités

D'abord :	()
Ensuite	**
Ensuite	* / % //
Puis	+ -

3 - Le type str : string, chaîne de caractères

Le type texte se nomme string (str). Il est délimité par des guillemets.

COLLER et COMPLETER LE DOCUMENT n°1

Concaténation de str avec +

```
>>> "Bon" + "jour"
```

```
>>> "5" + "2" + "3"
```

Suppléments

```
>>> "Bo" + "um " * 3
```

Répétition d'un str avec *

```
>>> "Bon" * 2
```

```
>>> "Bon" + "jour " * 2
```

```
>>> ("Bo" + "um") * 3
```

3 - Le type str : string, chaîne de caractères

Le type texte se nomme string (str). Il est délimité par des guillemets.

COLLER et COMPLETER LE DOCUMENT n°1

Concaténation de str avec +

```
>>> "Bon" + "jour"
```

```
'Bonjour'
```

```
>>> "5" + "2" + "3"
```

```
'523'
```

Répétition d'un str avec *

```
>>> "Bon" * 2
```

```
'BonBon'
```

```
>>> "Bon" + "jour " * 2
```

```
'Bonjour jour '
```

Suppléments

```
>>> "Bo" + "um " * 3
```

```
'Boum um um '
```

```
>>> ("Bo" + "um") * 3
```

```
'BoumBoumBoum'
```


Attention aux types des données :

```
>>> 5 * 2
```

```
>>> "5" * 2
```

Attention aux types des données :

```
>>> 5 * 2
```

```
10
```

```
>>> "5" * 2
```

```
'55'
```

COLLER et COMPLETER LE DOCUMENT n°2

Fonction **chr()** : renvoie le _____ au code fourni entre parenthèses.

Fonction **ord()** : renvoie le _____ du caractère fourni entre parenthèses.

Fonction **len()** : renvoie le _____ d'éléments dans la donnée fournie.

Fonction **print()** : ne renvoie rien mais affiche et interprète l'entrée.

```
>>> chr(65)
```

```
'A'
```

```
>>> ord('A')
```

```
_____
```

```
>>> len("Bob")
```

```
_____
```

```
>>> chr(66)
```

```
'B'
```

```
>>> ord('B')
```

```
_____
```

```
>>> len("B o b")
```

```
_____
```

```
>>> chr(10)
```

```
'\n'
```

```
>>> ord('\n')
```

```
_____
```

```
>>> len("Bo\nb")
```

```
_____
```

COLLER et COMPLETER LE DOCUMENT n°2

Fonction **chr()** : renvoie le **caractère correspondant** au code fourni .
Fonction **ord()** : renvoie le **code** du caractère fourni entre parenthèses.
Fonction **len()** : renvoie le **nombre** d'éléments dans la donnée fournie.
Fonction **print()** : ne renvoie rien mais affiche et interprète l'entrée.

```
>>> chr(65)
```

```
'A'
```

```
>>> ord('A')
```

```
65
```

```
>>> len("Bob")
```

```
>>> chr(66)
```

```
'B'
```

```
>>> ord('B')
```

```
66
```

```
>>> len("B o b")
```

```
>>> chr(10)
```

```
'\n'
```

```
>>> ord('\n')
```

```
10
```

```
>>> len("Bo\nb")
```

COLLER et COMPLETER LE DOCUMENT n°2

Fonction **chr()** : renvoie le **caractère correspondant** au code fourni .
Fonction **ord()** : renvoie le **code** du caractère fourni entre parenthèses.
Fonction **len()** : renvoie le **nombre** d'éléments dans la donnée fournie.
Fonction **print()** : ne renvoie rien mais affiche et interprète l'entrée.

```
>>> chr(65)
```

```
'A'
```

```
>>> ord('A')
```

```
65
```

```
>>> len("Bob")
```

```
3
```

```
>>> chr(66)
```

```
'B'
```

```
>>> ord('B')
```

```
66
```

```
>>> len("B o b")
```

```
5
```

```
>>> chr(10)
```

```
'\n'
```

```
>>> ord('\n')
```

```
10
```

```
>>> len("Bo\nb")
```

```
4
```

Notez bien que `\n` représente 1 caractère : le passage à la ligne.

```
>>> print( ' Bob ' )
```

```
...
```

```
>>> print( ' Bo\nb ' )
```

```
...
```

```
>>> print( ' abc\nABC\nZ ' )
```

```
...
```

Notez bien que `\n` représente 1 caractère : le passage à la ligne.

```
>>> print( ' Bob ' )
```

```
Bob
```

```
>>> print( ' Bo\nb ' )
```

```
Bo
```

```
b
```

```
>>> print( ' abc\nABC\nZ ' )
```

```
abc
```

```
ABC
```

```
Z
```

4 - Le type bool : True ou False

Le type booléen ne possède que deux valeurs possibles :

→ True pour dire vrai

→ False pour dire faux.

Les opérateurs booléens de cette partie ont une priorité plus petite que les opérateurs arithmétiques (* + ...)

COLLER et COMPLETER LE DOCUMENT n°3

```
>>> 8 > 10
```

```
_____
```

```
>>> 8 >= 5+3
```

```
_____
```

```
>>> 8 == 5 + 3
```

```
_____
```

```
>>> 8 != 5 + 3
```

```
_____
```

```
>>> 'on' in 'Bonbon'
```

```
_____
```

```
>>> 'ON' in 'Bonbon'
```

```
_____
```

```
>>> len('on') > len('Bonbon')
```

```
_____
```

COLLER et COMPLETER LE DOCUMENT n°3

```
>>> 8 > 10
```

```
False
```

```
>>> 8 >= 5+3
```

```
True
```

```
>>> 8 == 5 + 3
```

```
True
```

```
>>> 8 != 5 + 3
```

```
False
```

```
>>> 'on' in 'Bonbon'
```

```
True
```

```
>>> 'ON' in 'Bonbon'
```

```
False
```

```
>>> len('on') > len('Bonbon')
```

```
False
```

Explications (inutiles de les noter une fois compris)

>>> **8 > 10** : 8 est-il strictement supérieur à 10 ? Non donc False.

>>> **8 >= 5+3** : 8 est-il supérieur ou égal à (5+3) ? Oui car 8 est égal à (5+3)

>>> **8 == 5 + 3** : 8 est-il égal à (5+3) ? Oui, 8 est égal à (5+3)

>>> **8 != 5 + 3** : 8 est-il différent de (5+3) ? Non, 8 n'est pas différent de (5+3)

>>> **'on' in 'Bonbon'** : 'on' est-il présent dans 'Bonbon' ? Oui : True

>>> **len('on') > len('Bonbon')**

La longueur de 'on' est-elle strictement supérieure à la longueur de 'Bonbon' ? Non donc 2 n'est pas supérieur à 6. Donc False.

COLLER et COMPLETER LE DOCUMENT n°4

Bilan des opérateurs à connaître

+ est l'opérateur _____ ou _____

***** est l'opérateur _____ ou _____

/ est l'opérateur _____

// est l'opérateur _____ de la division euclidienne

% est l'opérateur _____ de la division euclidienne

in est l'opérateur _____

== est l'opérateur _____

!= est l'opérateur _____

>, >=, < et **<=** sont les opérateurs de _____

COLLER et COMPLETER LE DOCUMENT n°4

Bilan des opérateurs à connaître

+ est l'opérateur **addition** ou **concaténation**

***** est l'opérateur **multiplication** ou **répétition**

/ est l'opérateur **division**

// est l'opérateur **quotient** de la division euclidienne

% est l'opérateur **reste** de la division euclidienne

in est l'opérateur **appartenance**

== est l'opérateur **d'égalité**

!= est l'opérateur **de différence**

>, **>=**, **<** et **<=** sont les opérateurs **de comparaison**

5 - Exemples de gestion de données [PARTIE OPTIONNELLE]

Fonction `str()` : renvoie une copie des données sous forme d'un string.

```
>>> str(5)
```

```
'5'
```

```
>>> str(5)*4
```

```
'5555'
```

Fonction `int()` : renvoie une copie des données sous forme d'un entier.

```
>>> int("5")
```

```
5
```

```
>>> int("5")*4
```

```
20
```

La fonction `split()` renvoie un tableau contenant les différents morceaux **du string placé devant le point** en utilisant le caractère de séparation fourni.

```
>>> "Bonjour à tous".split(" ")
['Bonjour', 'à', 'tous']
      0         1         2
```

COLLER et COMPLETER LE DOCUMENT n°5

```
>>> "Bonjour à tous".split(" ") [0]
```

```
>>> "Bonjour à tous".split(" ") [2]
```

```
>>> "80.100.15.3".split(".")[1]
```

```
>>> int( "80.100.15.3".split(".")[3] )
```

```
>>> "Bonjour à tous".split(" ")
['Bonjour', 'à', 'tous']
      0         1         2
```

COLLER et COMPLETER LE DOCUMENT n°5

```
>>> "Bonjour à tous".split(" ") [0]
'Bonjour'
```

```
>>> "Bonjour à tous".split(" ") [2]
'tous'
```

```
>>> "80.100.15.3".split(".")[1]
'100'
```

```
>>> int( "80.100.15.3".split(".")[3] )
3
```


6 - Séquentialité dans la console [PARTIE OPTIONNELLE]

L'interpréteur exécute les lignes une par une.

Tant qu'il n'a pas fini une ligne, il ne passe pas à la suivante.

Console interactive

L'invite de commande est notée >>>

Impossible de modifier une ligne déjà exécutée.

En plaçant votre souris sur la console, on peut récupérer les dernières instructions avec la flèche UP.