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**.

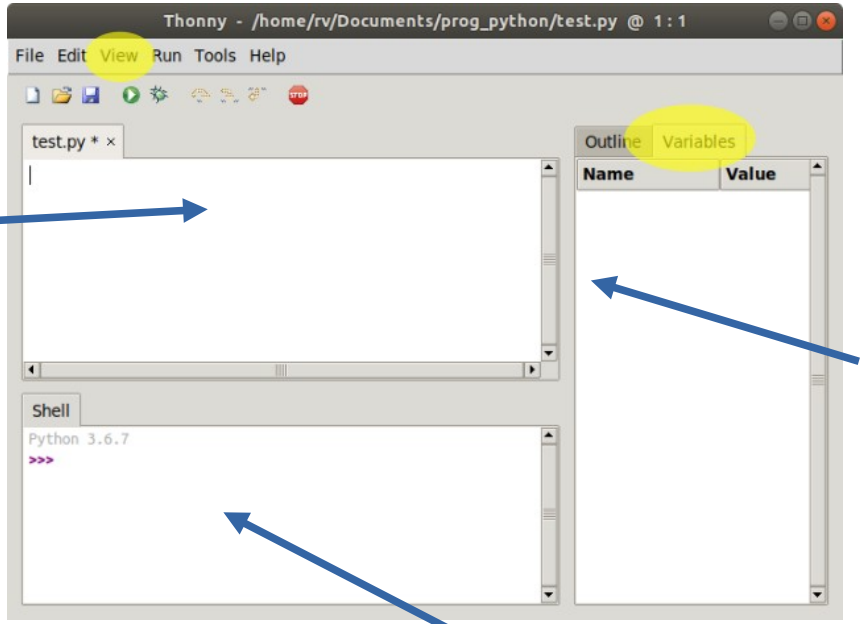
Scratch est un **langage de programmation par bloc**.

Python est **un langage de programmation par lignes de code**.

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'éxécutent tout de suite mais une seule fois

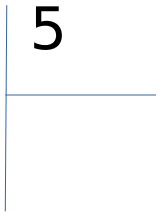
2 – Le type int : integer / entier

Il s'agit du type de données correspondant aux entiers.
On trouve les opérateurs classiques (+, -, *, /)

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

Mais également les opérateurs de la division euclidienne.

```
>>> 13 // 5     → Quotient de la division euclidienne
____
>>> 13 % 5      → Reste de la division euclidienne
```



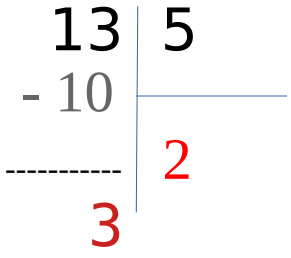
2 – Le type int : integer / entier

Il s'agit du type de données correspondant aux entiers.
On trouve les opérateurs classiques (+, -, *, /)

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

Mais également les opérateurs de la division euclidienne.

```
>>> 13 // 5     → Quotient de la division euclidienne
2
>>> 13 % 5      → Reste de la division euclidienne
3
```



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.

Concaténation de str avec +

```
>>> "Bon" + "jour"
'Bonjour'
```

```
>>> "5" + "2" + "3"
'523'
```

Répétition d'un str avec *

```
>>> "Bon" * 2
'BonBon'
```

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

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

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

3 – Le type str : string, chaîne de caractères

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

Concaténation de str avec +

```
>>> "Bon" + "jour"
'Bonjour'
```

```
>>> "5" + "2" + "3"
'523'
```

Répétition d'un str avec *

```
>>> "Bon" * 2
'BonBon'
```

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

```
>>> "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'
```

Fonction ord()

Renvoie un code

```
>>> ord('A')
65
>>> ord('\n')
10
```

Fonction chr()

Renvoie un caractère

```
>>> chr(65)
'A'
>>> chr(10)
'\n'
```

Fonction len()

Renvoie le nombre de caractères du string.

```
>>> len('abc ABC')
```

___

```
>>> len('abc\nABC\n')
```

___

Fonction ord()

Renvoie un code

```
>>> ord('A')
65
>>> ord('\n')
10
```

Fonction chr()

Renvoie un caractère

```
>>> chr(65)
'A'
>>> chr(10)
'\n'
```

Fonction len()

Renvoie le nombre de caractères du string.

```
>>> len('abc ABC')
7
```

```
>>> len('abc\nABC\n')
8
```

Fonction print()

Affiche l’exécution des caractères, même ceux de contrôle.

Ne renvoie rien.

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

Fonction print()

Affiche l’exécution des caractères, même ceux de contrôle.

Ne renvoie rien.

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

4 – Le type bool : True ou False

Ces opérateurs ont une priorité plus faible que + et -.

- Strictement supérieur >
- Supérieur ou égal >=
- Strictement inférieur <
- Inférieur ou égal <=
- Opérateur d’égalité ==
- Opérateur de différence !=
- Opérateur d’appartenance in

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

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

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

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

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

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

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

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

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

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

```
>>> 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
```

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 _____

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 – Séquentialité dans la console

L'interpréteur exécute les lignes une par une.

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

Pour comprendre un programme Python, il est donc vital de ne pas "sauter" d'instructions mais de comprendre TOUTES les lignes DANS L’ORDRE d’exécution.

Console interactive

L'invite de commande est notée >>>

On valide une ligne avec ENTREE.

Impossible de modifier une ligne déjà exécutée.

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