# Introduction

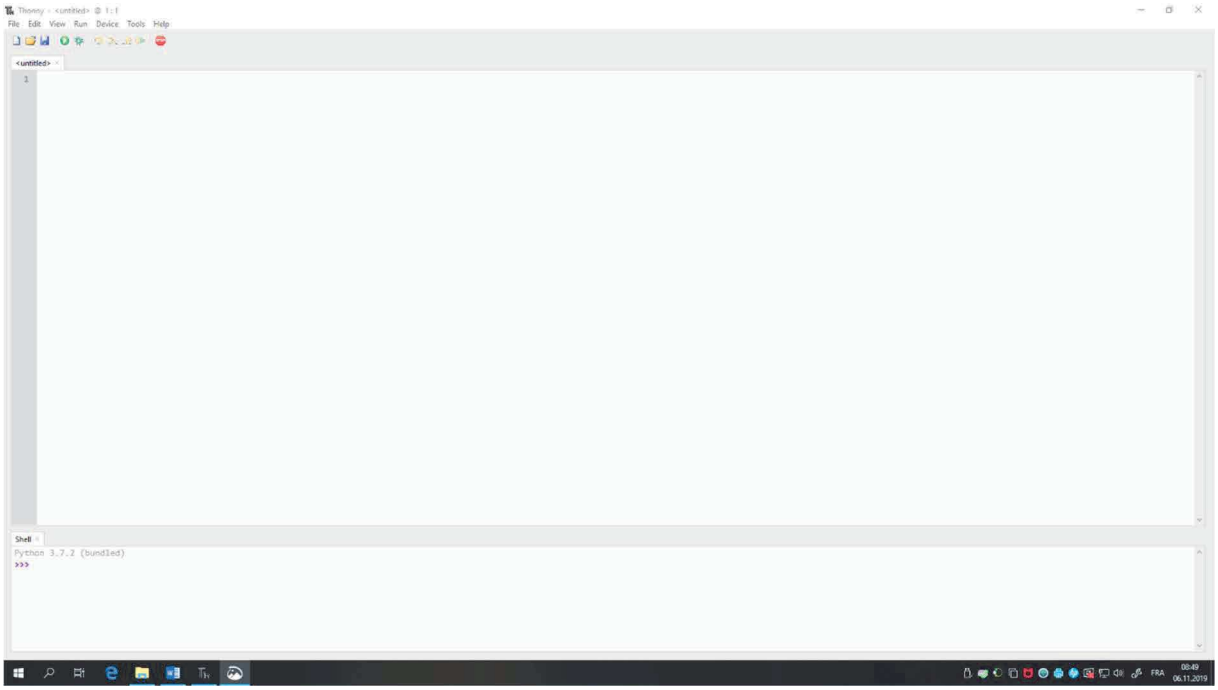
Python est un langage de programmation qui a vu le jour dans les années 90. C'est un langage *interprété* qui présente l'avantage d'être facile d'accès ce qui fait de lui un langage particulièrement adapté à l'apprentissage de la programmation.

Comme tout langage de programmation, Python repose sur des mots réservés c'est-à-dire sur un vocabulaire constitué de mots dédiés dérivés de l'anglais (for, if, while, etc.) ainsi que sur une syntaxe qui doit être évidemment respectée sous peine de voir vos programmes échouer lamentablement.

Comme tout langage interprété, Python a besoin d'un environnement spécifique pour traduire le code et l'exécuter. L'environnement que nous utiliserons est l'environnement Thonny qui implémente Python dans ses versions 3.7 et suivantes. Cet environnement est téléchargeable sur internet et installable très facilement, le tout gratuitement.

# L'environnement Thonny

Lorsque vous exécutez Thonny pour la première fois, vous verrez ceci :




La fenêtre supérieure –celle dont l'onglet est intitulé <untitled>- est *la fenêtre de code*, c'est-à-dire celle dans laquelle vous taperez votre code sous forme de scripts et de fonctions. Nous y reviendrons par la suite, les concepts de scripts et de fonctions étant fondamentaux en Python.

La fenêtre inférieure –celle dont l'onglet est intitulé Shell- est *l'interpréteur de commandes*. Comme son nom l'indique, vous pourrez y taper des commandes Python qui seront immédiatement exécutées en pressant la touche <Entrée> de votre clavier.

# Pour commencer…

Pour le moment, vous allez utiliser l'interpréteur de commandes Python.

A cet endroit, tapez la commande

```
print("Hello world!")
```

puis pressez la touche <Entrée>.

Vous obtenez immédiatement le résultat de l'exécution de votre commande :

```
Hello world!
```

Dans la réalité des faits cette commande est *une instruction* Python. Cette instruction est constituée du mot réservé ou commande `print` et de son paramètre "Hello world!". Comme vous l'avez constaté la commande `print` affiche son paramètre à l'écran, le mot « `print` » signifiant « afficher » ou « imprimer » en anglais. Ce paramètre peut être de différents types : numériques, alphabétiques, etc. Pour la précision, toute commande, pour être exécutée, doit être systématiquement suivie par la pression de la touche <Entrée>.

Par exemple :

```
print(2)
```

affiche `2`.

La commande `print` peut admettre plusieurs paramètres y compris de types différents, séparés les uns des autres par des virgules.

Par exemple :

```
print("Il y a ", 3, " étudiants dans cette salle de cours.")
```

affiche `Il y a 3 étudiants dans cette salle de cours.`

Enfin, elle permet aussi d'afficher le résultat d'un calcul :

Par exemple :

```
print(2 + 3)
```

affiche `5`.

En combinant :

```
print("La somme de 2 + 3 est ", 2 + 3, " .")
```

affiche `La somme de 2 + 3 est  5 .`

En cas d'erreur dans la syntaxe d'une commande, vous avez la possibilité de remonter la liste des commandes que vous avez déjà tapées à l'aide de la touche fléchée ↑ autant de fois que vous le voulez.

Par exemple :

```
print(Salut mon ami.")
```

affiche

```
File "<pyshell>", line 1
    print(Salut mon ami.")
                ^
SyntaxError: invalid syntax
```

Vous pouvez ensuite reprendre votre commande en pressant une fois flèche vers le haut, corriger votre erreur (il manque le premier guillemet) puis presser la touche <Entrée>.

Ainsi

```
print("Salut mon ami.")
```

affiche

```
Salut mon ami.
```

# La question des variables

Une variable est un objet qui peut être représenté par une lettre ou un mot et dont le contenu est une valeur au sens large du terme dans la mesure où il peut s'agir d'un nombre, d'un mot, d'une phrase, etc.

Quelques contraintes :

- Le nom d'une variable doit obligatoirement débuter par une lettre.
- Un nom de variable ne peut pas comporter d'espace.
- On évitera d'utiliser les caractères accentués dans les noms de variables.

Par exemple :

`a`, `total`, `monNom`, `Mon_Prenom`, etc. sont des noms de variables valides.

Par contre :

`mon Nom`, `12a`, etc. ne sont pas des noms de variables valides.

Pour vous faire une idée se rapprochant d'une notion en principe connue ou au moins effleurée, la notion de variable en informatique s'apparente à celle de variable en mathématiques : x, y, z, etc.

Dans l'interpréteur de commandes Python, tapez ce qui suit :

```
a = 2
```
puis pressez la touche <Entrée>.

Ensuite, tapez encore :

```
b = 3
```
puis pressez la touche <Entrée>.

Rien ne s'est affiché à l'écran pour le moment…

Tapez maintenant :

```
print(a)
```

Cette fois, la valeur `2` s'affiche. Cela signifie que Python a conservé dans la mémoire de votre machine la valeur que vous aviez affectée à la variable `a`. Il en va de même, évidemment, avec la valeur de la variable b.

Si vous affectez une nouvelle valeur à la variable `a`, par exemple 12, la valeur de `a` passera à 12, comme vous pouvez vous en douter.

A partir de là, on peut affecter le résultat de la somme des valeurs des variables `a` et `b` à une nouvelle variable, par exemple `c`.

Tapez ce qui suit :

```
c = a + b
```
puis pressez la touche <Entrée>.

Puis :

```
print(c)
```

Le résultat `5` s'affiche à l'écran.

Un exemple avec une variable-phrase :

```
salut = "Salut mon ami."
```

Puis :

```
print(salut)
```

affiche `Salut mon ami.`

Une remarque importante : il n'y a pas de lien entre un nom de variable et son contenu. En clair, les affectations `salut = "Salut"` puis `salut = 2` produiront l'affichage `Salut` dans le premier cas, et l'affichage `2` dans le second. Il n'y a donc pas de relation entre le nom choisi pour cette variable et la valeur qui lui a été affectée.