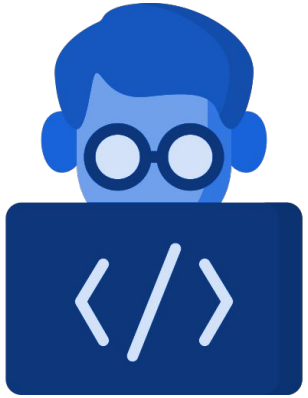
Initiation Python 3

Auteur : Cyril Vimard

Les programmes informatiques
L'algorithmie
Python : présentation
Installer Python
Interpréteur Python
Le Terminal
IDE
Calculs
Les variables
Formatage
Contrôle de flux
Opérateurs de comparaison
Les conditions
Les opérateurs logiques
Les listes
Les tuples
Les dictionnaires
Les boucles
Les fonctions
Programmation Orientée Objet



Les programmes informatiques

Un programme en Informatique : C’est quoi ?

Un programme informatique est **un ensemble d'instructions écrites dans un langage compréhensible par une machine** (ordinateur) pour effectuer une tâche spécifique.

Ces instructions sont généralement créées dans un langage de programmation tel que Python, Java, C++, etc.

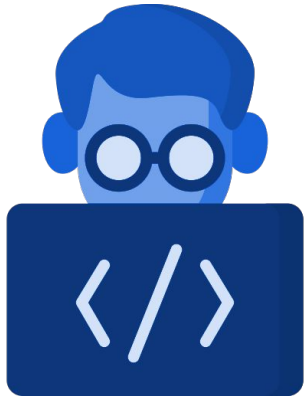
Les programmes sont conçus pour résoudre des problèmes, effectuer des calculs, manipuler des données, ou interagir avec des utilisateurs.

Un Langage de Programmation : C’est quoi ?

Un langage de programmation est un **moyen formel d'instruire un ordinateur** pour effectuer des tâches spécifiques.

Il permet aux programmeurs d'écrire des instructions compréhensibles par la machine, facilitant ainsi la création de logiciels et d'applications.

Les langages de programmation varient en termes de syntaxe, de sémantique et d'abstraction, mais ils partagent tous l'objectif commun de **traduire la logique humaine en une séquence d'instructions** exécutables par un ordinateur.



Différents Paradigmes de Programmation

Les paradigmes de programmation sont des approches philosophiques ou méthodologiques pour concevoir et structurer le code.

Chaque paradigme offre une manière particulière d'organiser les idées et les opérations dans un programme.

Voici quelques-uns des principaux paradigmes de programmation :

Impératif : Il se concentre sur la spécification de l'état initial et la définition des étapes à suivre pour obtenir un état final. Les langages impératifs incluent souvent des concepts tels que les boucles et les structures conditionnelles.
> ex : recettes

Déclaratif : Il se concentre sur la description de ce que le programme doit accomplir sans spécifier comment le faire. Le langage exprime une relation entre l'entrée et la sortie.
> ex : html

Orienté Objet : Il organise le code autour d'objets qui regroupent des données et des méthodes qui agissent sur ces données. Encapsulation, héritage et polymorphisme sont des concepts clés.
> ex : Python

L’Algorithmie

Un algorithme : c’est quoi ?

Un algorithme est **une séquence d'instructions détaillées** et non ambiguës qui décrit comment accomplir une tâche spécifique ou résoudre un problème.

Les algorithmes peuvent être exprimés de manière abstraite, indépendamment du langage de programmation.

Ils **constituent la base logique** sur laquelle les programmes informatiques sont construits. Un bon algorithme doit être **clair**, **précis**, **fini** et donner la **solution correcte** pour toutes les entrées valides.

Caractéristiques d'un Algorithme :

Clarté : Les étapes de l'algorithme doivent être compréhensibles sans ambiguïté.
Précision : Chaque étape de l'algorithme doit être définie de manière précise, sans interprétation multiple.
Finitude : L'algorithme doit se terminer après un nombre fini d'étapes.
Correction : L'algorithme doit fournir la solution correcte pour toutes les entrées valides.

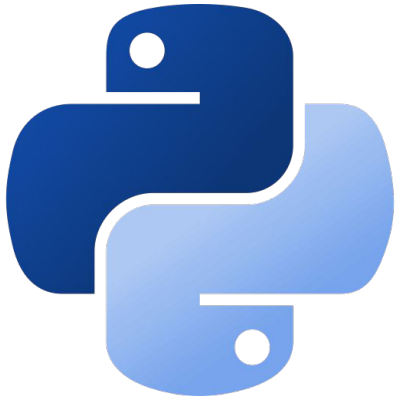
L'algorithme : Le pseudo-code.

Le **pseudo-code** est un moyen de décrire un algorithme de manière informelle, utilisant des conventions de langage naturel combinées avec des éléments de syntaxe empruntés à divers langages de programmation.

Voici une structure générale d'un algorithme en pseudo-code. Dans cet exemple on détermine si un nombre est pair ou impair:

```
Algorithme DetectionPairImpair
    // Demander à l'utilisateur d'entrer un nombre
    Afficher "Entrez un nombre : "
    Lire nombre

    // Vérifier si le nombre est pair ou impair
    Si nombre % 2 == 0 Alors
        Afficher "Le nombre est pair."
    Sinon
        Afficher "Le nombre est impair."
    FinSi
FinAlgorithme
```



Python : présentation

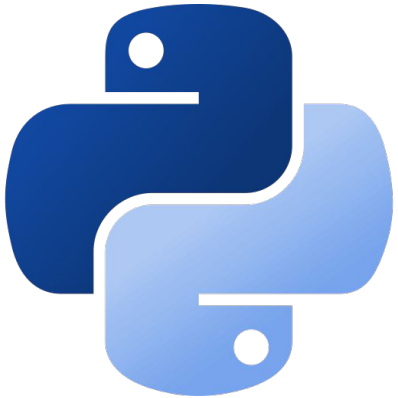
Python : c’est quoi ?

Python est un langage de programmation. Il est l'un des langages de programmation les plus intéressants et les plus populaires du moment. **Facile à apprendre**, python est souvent utilisé en exemple lors de l'apprentissage de la programmation.

Python est un langage de programmation inventé par Guido van Rossum en 1989. **La première version de python est sortie en 1991** et se nomme ainsi en hommage aux Monty Python’s Flying Circus.

Python est un langage de programmation interprété, c'est-à-dire qu'il n'est pas nécessaire de le compiler avant de l'exécuter.

Si vous avez déjà touché un peu à la programmation, vous verrez que ce langage possède une certaine poésie. Les programmeurs s'amusent souvent à trouver la manière la plus élégante et efficace d'écrire une suite d'instructions.

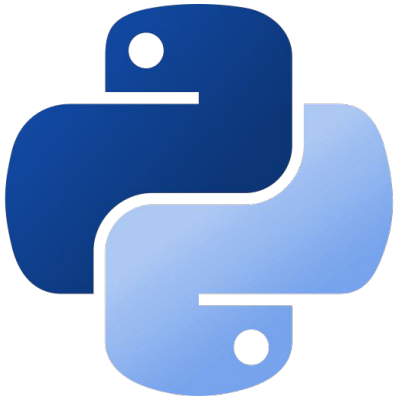


Python : que permet-il de faire ?

Python est à la fois **simple et puissant**, il vous permet d'écrire des scripts très simples et grâce à ses nombreuses bibliothèques, vous pouvez travailler sur des projets plus ambitieux.

- **Web**: Aujourd'hui python combiné avec le **framework Django** est un **très bon choix technologique pour des gros projets de sites internet**.

- **Système**: Python est également **souvent utilisé par les admin système pour créer des tâches dites répétitives** ou simplement de maintenance.
  D'ailleurs si vous voulez créer des applications java en codant en python, c'est possible grâce au projet Jython.



Python : pourquoi le préférer aux autres langages ?

Python est **un langage facile à apprendre** et **son code est plus lisible**, il est donc plus facile à maintenir. **Il est parfois jusqu'à 5 fois plus concis que le langage Java par exemple**, ce qui **augmente la productivité** du développeur et **réduit mécaniquement le nombre de bugs**.

L'environnement python est riche en librairies. Vous trouverez toujours des projets open source qui vous faciliteront la vie.

Python a été **pensé pour créer du code complexe en peu de lignes**. N'oublions pas qu'un bon codeur n'est pas celui qui arrive à faire comprendre à sa machine ce qu'il veut faire mais qui fait comprendre aux autres développeurs ce qu'il a voulu faire !

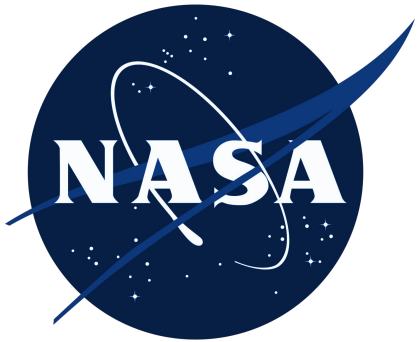
Python est également **utilisé dans les milieux scientifiques**, par exemple la bioinformatique. Des librairies sont disponibles pour ce domaine comme le module **biopython** .

Il existe également des bibliothèques facilitant la **création de jeux vidéo en 2D (et 3D)** exemple: **pyGame** .

La **documentation python est également extrêmement bien faite**, aussi bien pour les débutants que pour les experts. Et elle est très riche car il y a une très grande communauté derrière.

Python : Qui l’utilise ?

Google (Guido van Rossum a travaillé pour Google de 2005 à 2012), Yahoo, Microsoft, la Nasa revendiquent l'utilisation de Python, pour ne citer qu'eux.

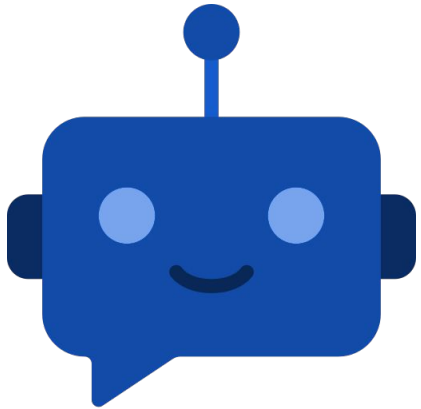


Python : et l'intelligence artificielle dans tout ça ?

Python est également très **populaire dans le domaine de l'IA** en raison de ses bibliothèques de haute qualité et de sa puissance de calcul.

Beaucoup de chercheurs et de développeurs utilisent Python pour **implémenter et expérimenter avec de nouveaux algorithmes d'apprentissage automatique et de deep learning**, en particulier grâce aux frameworks tels que **TensorFlow** et **PyTorch**.

En utilisant Python, **il est possible de développer rapidement des modèles complexes et puissants**, ce qui en fait un choix idéal pour les projets d'IA de toutes tailles.



Installer Python

Python : Installation

Installer python sur Linux ou MacOS Linux Python

Si vous travaillez dans un environnement Linux ou MacOS : bonne nouvelle Python est déjà installé !

Installer python sur Windows

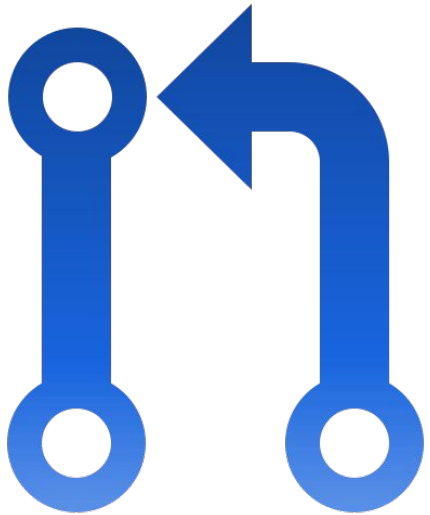
Téléchargez un fichier d'installation python à cette adresse: Télécharger Python (python.org)

Au cours de l'installation, **pensez à cocher "Ajouter Python 3.x au PATH"**.

Python : Quelle Version Choisir ?

Prenez la version la plus récente / stable.

À noter que la version la plus utilisée aujourd'hui est la **version 3.+**



Python : les bibliothèques (ou librairies)

En Python, une bibliothèque (ou un module) est un **ensemble de fonctions et de méthodes prédéfinies, réutilisables** dans différents programmes. Ces bibliothèques sont généralement conçues pour fournir des fonctionnalités spécifiques qui peuvent être intégrées dans des programmes Python.

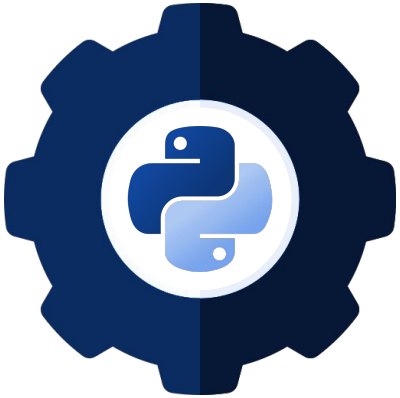
Elles permettent :

- La réutilisabilité du code
- l'extension des fonctionnalités

Certaines bibliothèques sont installées directement avec Python. D'autres ont besoin d'être installées en lignes de commande.

Pour importer une bibliothèque dans un script on place généralement la commande tout au début comme suit :

```
>>> import Requests
```



Interpréteur Python

Python : l'interpréteur

Python est un langage interprété. Cela signifie qu'il a besoin d'un programme, un interpréteur, afin de fonctionner et de transformer votre code en une suite d'instructions compréhensibles par la machine.

Quand vous avez installé Python sur votre ordinateur, vous avez installé entre autres choses un interpréteur.

Maintenant, quand vous aurez écrit des scripts, vous allez ensuite les tester ou les faire fonctionner en utilisant l'interpréteur en passant par le terminal.

Le Terminal

Python : le Terminal

Pour utiliser Python notamment vous devez lancer un terminal.

Un terminal fait référence à l'**interface en ligne de commande** elle-même. C'est un environnement texte dans lequel un utilisateur peut entrer des commandes pour interagir avec un système d'exploitation.

Par exemple, dans les systèmes basés sur Unix (comme Linux), le terminal est souvent appelé **"shell"**. Les utilisateurs peuvent y saisir des commandes pour effectuer des tâches telles que la navigation dans les fichiers, l'exécution de programmes, etc.

Pour lancer l'interpréteur python tapez python, puis entrée :

```
python
```

```
[~] Python 3.9.7 (default, Sep 16 2021, 13:09:58)
[GCC 7.5.0] :: Anaconda, Inc. on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Python : le Terminal

Vous remarquez les 3 chevrons **>>>** , cela signifie que l'interpréteur python est prêt à recevoir des instructions.

De manière générale, si vous voyez le symbole des 3 chevrons sur une page de tutoriel, cela signifie que vous devez exécuter le code affiché dans l'interpréteur python. Vous pouvez par exemple lui demander d'exécuter une simple addition:

```
python3

[~] Python 3.9.7 (default, Sep 16 2021, 13:09:58)
[GCC 7.5.0] :: Anaconda, Inc. on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> 3+1
4
>>>
```

IDE

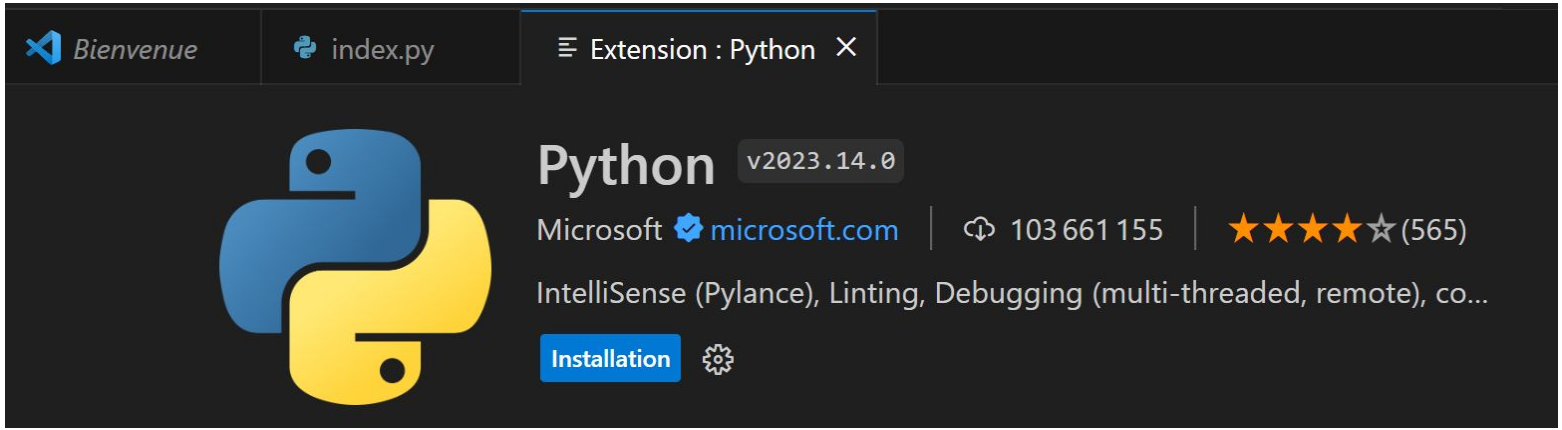
IDE: Installer et configurer VSCode

Une IDE (Integrated Development Environment) c'est un environnement de développement intégré, c'est à dire une application qui aide les programmeurs à développer dans un environnement, de façon optimisée.

Télécharger et installer VSCode depuis le site officiel : Visual Studio Code

IDE: Installer et configurer VSCode

Installer l'extension Python pour VSCode.
Ouvrir Visual Studio Code.
Aller dans l'onglet "**Extensions**" (icône carrée à gauche ou Ctrl+Shift+X).
Rechercher "**Python**" dans la barre de recherche.
Installer l'extension fournie par Microsoft appelée "**Python**" (elle a généralement l'icône Python).



IDE: créez votre premier fichier et l'exécuter

Étape 1 : **Créer un nouveau fichier Python** : Ouvrir VSCode. Cliquer sur "File" (Fichier) -> "New File" (Nouveau fichier). Enregistrer le fichier avec l'extension ".py", par exemple, "mon_script.py".

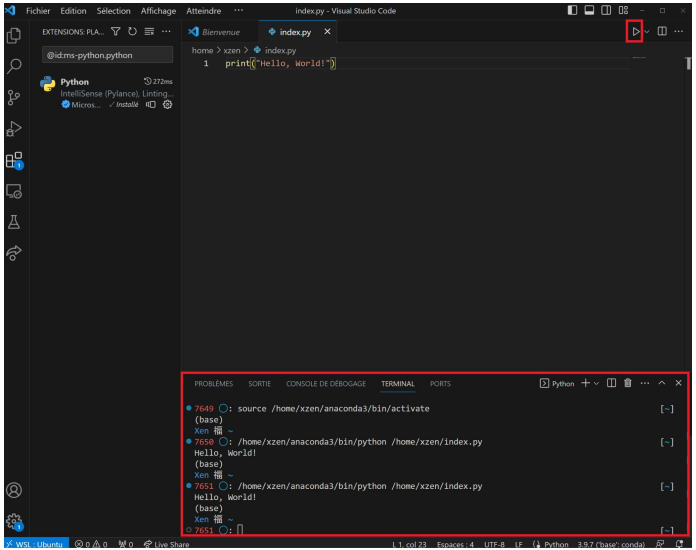
Étape 2 : **Écrire du code Python**

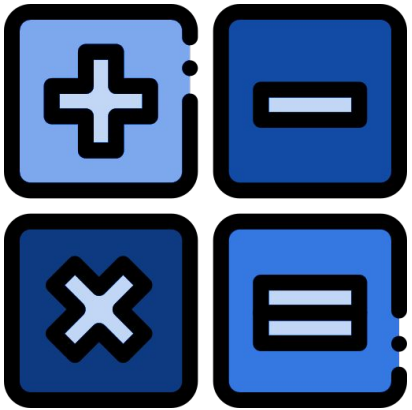
```
print("Hello, World!")
```

Étape 3 : **Exécuter le code Python**

Cliquer sur le bouton play situé en haut à droite de l'éditeur ou appuyer sur Ctrl+Shift+P pour ouvrir la palette de commandes, puis taper "**Python: Run Python File in Terminal**".

Le résultat s'affiche dans le terminal en bas de l'éditeur.





Calculs

Calculs

Une des fonctionnalités de base d'un interpréteur est de faire des calculs:

```
>>> 1+2
3
```

Vous pouvez ajouter des espaces, cela n'aura pas d'incidences:

```
>>> 1 + 2
3
```

Tous les opérateurs sont utilisables:

```
>>> 1-10      -9
>>> 2*10      20
>>> 100/4     25
>>> 10%4      2
>>> 10//4     2
>>> 2**3      8
```

La double étoile représente l'exposant.

Calculs

Il existe cependant une erreur à éviter pour les versions Python inférieures à Python 3 :

```
>>> 10/3
3
```

Surprise 10/3 = 3 .

Alors quelle est donc cette folie?
Python raisonne en nombres entiers puisque nous lui avons fourni deux nombres entiers. Pour avoir un résultat en décimales, il vous faudra utiliser cette syntaxe:

```
>>> 10.0/3    3.3333333333333335
>>> 10/3.0    3.3333333333333335
```

Calculs : Opérateurs mathématiques

De la priorité la plus élevée à la plus basse :

Les opérateurs	Opération	Exemple
`**`	Exposant	`2 ** 3 = 8`
`%`	Module/Reste	`22 % 8 = 6`
`//`	Division entière	`22 // 8 = 2`
`/`	Division	`22 / 8 = 2.75`
`*`	Multiplication	`3 * 3 = 9`
`-`	Soustraction	`5 - 2 = 3`
`+`	Ajout	`2 + 2 = 4`

Calculs : Opérateurs mathématiques

Exemples d'expressions :

```
>>> 2 + 3 * 6
20

>>> (2 + 3) * 6
30

>>> 2 ** 8
256

>>> 23 // 7
3

>>> 23 % 7
2

>>> (5 - 1) * ((7 + 1) / (3 - 1))
16.0
```

Les variables

Les variables

Dans les mathématiques élémentaires, **une variable désigne une grandeur dont la valeur est (provisoirement) indéterminée**, mais sur laquelle est effectuée une combinaison d'opérations avec soit des constantes, soit d'autres variables.

Une variable n'est pas totalement précisée, sa valeur peut même être inconnue, mais elle doit appartenir à un ensemble.

Faire des calculs avec les variables comme s'il s'agissait de nombres explicites permet de résoudre, en une seule fois, des problèmes analogues.

La variable est habituellement représentée par un symbole, le plus souvent une lettre de l'alphabet latin telle que x ou y.

Le terme variable provient du fait que lorsque l'antécédent x de la fonction varie, alors son image y varie aussi.

Les variables

Voici un exemple simple de déclaration de variable en pseudo-code :

```
VARIABLES
 nbr1 EST_DU_TYPE NOMBRE
 nbr2 EST_DU_TYPE NOMBRE
 hello EST_DU_TYPE CHAINE
DEBUT_ALGORITHME
FIN_ALGORITHME
```

Variable: Les variables

Une variable est une sorte de boîte virtuelle dans laquelle **on peut mettre une (ou plusieurs) donnée(s)**. L'idée est de **stocker temporairement une donnée** pour travailler avec.

Pour votre machine une variable est une adresse qui indique l'emplacement de la mémoire vive où sont stockées les informations que nous avons liées avec.

Affectons une valeur à la variable **age** que nous allons ensuite afficher:

```
>>> age = 30
>>> age
30
```

On va ensuite **ajouter 10** à la valeur de cette variable:

```
>>> age = 30
>>> age = age + 10
>>> age
40
```

Il est possible de **mettre une variable dans une autre variable**.

```
>>> age = 30
>>> age2 = age
>>> age2
30
```

Variable: Les variables

Vous pouvez mettre à peu près tout ce que vous voulez dans votre variable, y compris du texte:

```
>>> age = "J'ai 30 ans"
>>> age
"J'ai 30 ans"
```

Il est possible de concaténer, c'est à dire d'ajouter du texte à du texte:

```
>>> age = age + " et je suis encore jeune!"
>>> age
"J'ai 30 ans et je suis encore jeune!"
```

Vous pouvez même multiplier une chaîne de caractères:

```
>>> age = "jeune"
>>> age * 3
'jeunejeunejeune'
```

Evidemment, si vous essayez de faire des additions avec des variables qui sont des chiffres et d'autres qui sont du texte, l'interpréteur renverra une erreur:

```
>>> age = "J'ai 30 ans"
>>> age
"J'ai 30 ans"
>>> age + 1
Traceback (most recent call last):   File "<stdin>", line 1, in <module>
TypeError: cannot concatenate 'str' and 'int' objects
```

Variable: Échapper les quotes

Comment inclure le caractère quote puisqu'il indique un début de données et une fin de données?

```
>>> texte = "Bonjour je m'appelle \"Olivier\""
>>> texte
Bonjour je m'appelle "Olivier"
```

ou

```
>>> texte = 'Bonjour je m\'appelle "Olivier"'
>>> texte
'Bonjour je m\'appelle "Olivier"'
```

Il existe une autre manière de stocker du texte dans une variable: l'utilisation d'un triple quote. Cela permet de ne pas échapper les caractères quotes des données

```
>>> texte = """
... Bonjour je m'appelle "Olivier"
... """
>>> texte
'\nBonjour je m\'appelle "Olivier"\n'
```

Variable: Nommer une variable

Vous ne pouvez pas nommer les variables comme bon vous semble, puisqu'il existe déjà des mots utilisés par Python. Voici la liste des mots réservé par python:

```
print, in, and, or, if, del, for, is, raise, assert, elif, from, lambda,
return, break, else, global, not, try, class, except, while, continue,
exec, import, pass, yield, def, finally
```

Pourquoi ces mots sont-ils réservés ? Parce qu'**ils servent à faire autre chose**.
Nous verrons quoi plus en détail dans les prochains chapitres.

Pour nommer une variable vous ne pouvez utiliser que les lettres de l'alphabet, les chiffres et le caractère "_". Les accents et les signes de ponctuation sont interdits. De plus les chiffres ne doivent jamais se trouver en première position dans votre variable:

```
>>> 1var = 1
  File "<stdin>", line 1
    1var = 1
         ^
SyntaxError: invalid syntax
```

Comme vous le remarquez, python refuse ce genre de syntaxe, mais il acceptera **var1 = 1** .

Et dernier point, attention à la casse !!

Variable: opérateurs d'affectation augmentée

Opérateur	Équivalent
`var += 1`	`var = var + 1`
`var -= 1`	`var = var - 1`
`var *= 1`	`var = var * 1`
`var /= 1`	`var = var / 1`
`var %= 1`	`var = var % 1`

Exemples :

```
>>> greeting = 'Hello'
>>> greeting += ' world!'
>>> greeting
'Hello world!'

>>> number = 1
>>> number += 1
>>> number
2
```

```
>>> my_list = ['item']
>>> my_list *= 3
>>> my_list
['item', 'item', 'item']
```

Variable: opérateur de Walrus

L'opérateur **Walrus** permet l'**affectation de variables dans une expression tout en renvoyant la valeur** de la variable.
Exemple:

```
>>> print(my_var:="Hello World!")
'Hello world!'

>>> my_var="Yes"
>>> print(my_var)
'Yes'

>>> print(my_var:="Hello")
'Hello'
```

L' opérateur **Walrus**, ou **Assignment Expression Operator**, a été introduit pour la première fois en 2018 via PEP 572, puis officiellement publié avec Python 3.8 en octobre 2019.

Les types de variables

Types de variables

Les entiers (int)

Comme en mathématiques, les nombres entiers représentent des valeurs entières sans décimales.

```
x = 5
y = -10
```

Les décimaux (float)

Les nombres à virgule flottante représentent des valeurs avec des décimales.

```
pi = 3.14
price = 19.99
```

Les chaînes de caractères (str)

Les chaînes de caractères représentent du texte.

```
message = "Bonjour, monde!"
name = 'Alice'
```

Les booléens (bool)

Les valeurs booléennes peuvent prendre uniquement 2 valeurs : vrai (True) ou faux (False).

```
is_python_fun = True
is_learning = False
```

Les listes (list)

Les listes sont des collections ordonnées pouvant contenir divers types de données.

```
numbers = [1, 2, 3, 4, 5]
names = ['Alice', 'Bob', 'Charlie']
```

Les tuples (tuple)

Les tuples sont des collections ordonnées immuables.

```
coordinates = (10, 20)
colors = ('red', 'green', 'blue')
```

Les ensembles (set)

Les ensembles sont des collections d'éléments non ordonnées, non indexés et non modifiables qui n'acceptent pas de contenir plusieurs fois le même élément.

```
unique_numbers = {1, 2, 3, 4, 5}
```

Les dictionnaires (dict)

Les dictionnaires sont des collections associatives de paires clé-valeur.

```
person = {'name': 'Alice', 'age': 30, 'city': 'Wonderland'}
```

Les noneType (None)

None est utilisé pour représenter l'absence de valeur ou une valeur nulle.

```
result = None
```

Utilisation des types de variables

Les types de variables

Python est un **langage à typage dynamique**, ce qui signifie qu'il n'y a pas besoin de déclarer explicitement le type d'une variable.

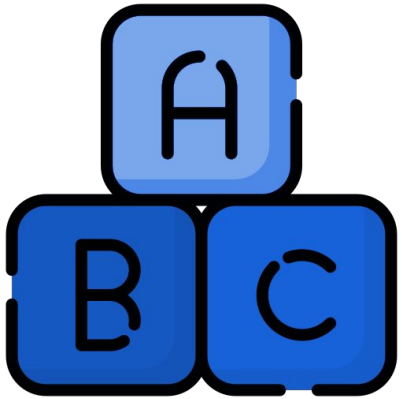
Tu peux utiliser **la fonction type()** pour connaître le type d'une variable :

```
x = 5
print(type(x))  # Affiche <class 'int'>
```

Python est également fortement typé, ce qui signifie que les opérations entre différents types peuvent générer des erreurs.

```
# Exemple d'erreur
result = "Hello" + 42  # Génère une TypeError
```

Python offre une flexibilité significative en termes de types de variables, ce qui facilite le développement, mais il est toujours important de comprendre les types pour écrire un code robuste.



Formatage

Formatage : Chaîne de caractères

Une f-string ou chaîne f est une chaîne littérale formatée, préfixée par **f ou F**.

Ces chaînes peuvent contenir des champs de remplacement, qui sont des expressions délimitées par **des accolades {}**.

Alors que les autres chaînes littérales ont toujours une valeur constante, **les chaînes formatées sont en fait des expressions évaluées au moment de l'exécution**.

```
>>> name = 'Elizabeth'
>>> print(f'Hello {name}!')
'Hello Elizabeth!'
```

Il est même possible de faire des opérations arithmétiques à l'intérieures de ces f-strings :

```
>>> a = 5
>>> b = 10
>>> f'a + b est égal à {a + b}, et le double est égal à {2 * (a + b)}.'
'a + b est égal à 15, et le double est égal à 30.'
```

Formatage : f-string multiligne

```
>>> name = 'Robert'
>>> messages = 12
>>> print(
...       f'Hi, {name}. '
...       f'You have {messages} unread messages'
...       )
'Hi, Robert. You have 12 unread messages'
```

Formatage : Le spécificateur =

Cela imprimera l'expression **et** sa valeur :

```
>>> from datetime import datetime
>>> now = datetime.now().strftime("%b/%d/%Y - %H:%M:%S")
>>> f'date and time: {now=}'

"date and time: now='Nov/22/2023 - 13:33:12'"
```

[Les Variables]
Workshop

Réaliser les exercices

Réalisez les exercices suivants (vous n'avez pas le droit d'utiliser les boucles, les conditions ou toutes autres possibilités algorithmique de python hormis les variables et tout ce qui peut s'y associer).

L'intégralité de vos exercices seront à faire avec **Visual Studio Code** dans un dossier variable avec à l'intérieur un fichier par exercice par exemple :

/variables
—--- exo1.py
—--- exo2.py
....

Exercices disponible ici.

Récupérer le Cours

Scannez moi