**Variables**

**Pourquoi devrions-nous utiliser des variables ?**

Les variables sont là pour rendre votre code dynamique ; lors de la programmation, vous souhaitez exécuter le même code mais avec des valeurs différentes ; c'est pourquoi les variables ont été créées.

**Syntaxe**

Pour définir une variable, nous écrivons son nom puis utilisons le signe égal pour lui attribuer la valeur que nous souhaitons.

hello = "Hello World."

Les variables doivent être écrites sous la forme d'un seul mot, mais vous pouvez utiliser des traits de soulignement au lieu d'espaces.

my\_age = 23

Les noms de variables doivent toujours être explicites.

**1. Déclarer une variable**

Déclarer une variable est vraiment simple : variable\_name = "variable value", par exemple :

my\_name = "Rick"

my\_age = 43

**2. Noms des variables**

Les noms de variables doivent respecter certaines conventions, décrites dans la documentation [PEP 8.](https://www.python.org/dev/peps/pep-0008/#function-and-variable-names) Cette documentation définit les normes que votre code doit respecter.  
Les noms de variables doivent :

* Être composé uniquement de lettres minuscules ou de chiffres, avec des mots séparés par des traits de soulignement
* Ne jamais commencer par un chiffre
* Soyez facile à lire et à comprendre
* Ne pas être l'un des [mots-clés Python](https://www.programiz.com/python-programming/keywords-identifier)

**3. Utilisation des variables**

Pour utiliser une variable, tapez son nom et Python le remplacera par sa valeur.

my\_age = "43"

my\_name = "Rick"

print("My name is " + my\_name + ", and I am " + my\_age + " years old.")

"My name is Rick, and I am 43 years old."

**Important!!**

Vous vous demandez peut-être pourquoi « » my\_age = "43"est entouré de guillemets.  
Il est important de se rappeler que les valeurs numériques non entourées de guillemets sont considérées comme des entiers.

my\_age = 43

my\_age1 = "43"

These two are not the same!

type(my\_age) # The 43 value makes this an integer

type(my\_age1) # The " " turns the number 43 into a string

Comme nous l'avons vu auparavant dans la leçon précédente, les entiers ne peuvent pas être concaténés ou formatés en phrases avec des chaînes, par exemple, si je devais essayer d'imprimer ceci :

my\_age = 43

print("My name is Rick, and I am " + my\_age + " years old.")

Une erreur apparaîtra

TypeError: can only concatenate str (not "int") to str

Cette erreur de type explique que les entiers et les chaînes ne peuvent pas être concaténés. Si cela se produit, pensez donc à placer vos nombres entre guillemets.  
Il existe cependant une solution pour contourner cette erreur grâce au formatage de chaîne ; vous trouverez des exemples ci-dessous.

**4. Afficher la valeur d'une variable**

Pour afficher la valeur d'une variable, vous pouvez utiliser la printfonction. Par exemple :

>>> my\_hair\_color = "black"

>>> print(my\_hair\_color)

"black"

Vous pouvez également passer plusieurs arguments à print, en les séparant par une virgule ,. Python affichera tous les éléments séparés par un espace. Par exemple :

my\_hair\_color = "black"

print("My hair is", my\_hair\_color)

"My hair is black"

**5. Incrémenter une variable**

Incrémenter signifie « Ajouter un à », pour incrémenter une variable, remplacez sa valeur par « elle-même plus un » :

my\_number = 5

# Increment it

my\_number = my\_number + 1

# Now my number is 6

On peut aussi écrire +=; cet opérateur a la même signification que +.

my\_number = 5

my\_number += 5

# += means "= my\_number +"

#In this example the value of "my\_number" will be 10

Cette syntaxe peut être utilisée avec **tous les opérateurs** .

**6. Formatage des chaînes**

Lorsque vous travaillez avec des chaînes de caractères contenant plusieurs variables, leur formatage peut s'avérer complexe. Python propose des solutions pour simplifier cette tâche, et à partir de Python 3.6, **les chaînes f** (littérales de chaîne formatées) sont devenues la méthode privilégiée pour le formatage des chaînes en raison de leur simplicité et de leur rapidité. Cependant, il est important de comprendre l'ancienne str.format()méthode, car vous pourriez la rencontrer dans du code obsolète.

**f-strings (préféré dans Python 3.6+)**

**Les chaînes de caractères (f-strings)** permettent d'insérer des expressions dans des chaînes littérales, à l'aide d'accolades {}. Voici leur fonctionnement :

1. Commencez la chaîne avec un favant les guillemets d'ouverture.
2. Placez toutes les variables ou expressions à l'intérieur des accolades {}.

Exemple:

name = "Frank"

age = 65

print(f"Hello, {name}. You are {age}.")

**Sortir:** 'Hello, Frank. You are 65.'

Les chaînes de caractères f sont évaluées à l'exécution, ce qui signifie que vous pouvez y intégrer n'importe quelle expression Python valide, et pas seulement des variables. Par exemple :

print(f"The result of 2 \* 35 is {2 \* 35}")

**Sortir:** 'The result of 2 \* 35 is 70'

C'est pourquoi les chaînes de caractères f sont plus rapides et plus lisibles que les anciennes méthodes de formatage. C'est l'approche la plus couramment utilisée dans le code Python moderne.

**La str.format()Méthode (Héritage)**

Avant les chaînes f, cette str.format()méthode était largement utilisée. Bien qu'elle soit moins courante dans Python 3, vous pourriez la rencontrer dans des bases de code plus anciennes ; il est donc important de la comprendre.

Avec str.format(), les champs de remplacement sont indiqués par des accolades {}. Voici un exemple :

name = "Frank"

age = 65

print("Hello, {}. You are {}.".format(name, age))

**Sortir:** 'Hello, Frank. You are 65.'

Dans cette méthode, le nombre d'accolades {}doit correspondre au nombre d'arguments passés à format(). Vous pouvez également réorganiser ou réutiliser les arguments en utilisant leur index entre les accolades :

print("Hello, {1}. You are {0}.".format(age, name))

**Sortir:** 'Hello, Frank. You are 65.'

**Exemple de comparaison :**

Voici un exemple de formatage d'une date utilisant à la fois des chaînes f etstr.format() :

* **En utilisantstr.format()** :

birthday\_day = 8

birthday\_month = 2

birthday\_year = 1979

print("I was born the {}/{}/{}".format(birthday\_day, birthday\_month, birthday\_year))

**Sortir:** "I was born the 8/2/1979"

* **Utilisation des f-strings** :

birthday\_day = 8

birthday\_month = 2

birthday\_year = 1979

print(f"I was born the {birthday\_day}/{birthday\_month}/{birthday\_year}")

**Sortir:** "I was born the 8/2/1979"

**Remarques importantes :**

* **Les chaînes f** sont évaluées au moment de l'exécution, elles peuvent donc inclure n'importe quelle expression Python à l'intérieur des accolades {}, ce qui les rend très flexibles.
* **str.format()** est toujours valide, mais est désormais considéré comme obsolète par rapport aux chaînes f. Il est utile de le savoir si vous devez maintenir ou comprendre du code existant.

**Exercice 1**

Vous avez une amie nommée Alice et vous souhaitez lui envoyer un message avec les détails suivants :

Nom : Alice

Âge : 30 ans

Ville : New York

Tâches:

Utilisez des chaînes f pour imprimer un message indiquant :

« Bonjour Alice ! Tu as 30 ans et tu vis à New York. »

Utiliser str.format()pour imprimer le même message.

**Exercice 2**

Demandez à l'utilisateur son âge à l'aide de la input()fonction et enregistrez-le dans une variable d'âge.

Convertissez l'âge saisi en un entier et calculez le nombre d'années jusqu'à ce qu'ils atteignent 100 ans.

Afficher un message : « Vous aurez 100 ans dans X années », où X est le nombre d'années calculé.

**7. Prendre les données de l'utilisateur**

La fonction de saisie permet de demander à l'utilisateur de saisir une information. Elle renvoie toujours une stringréponse.

input('Please state your name: ')

**Stockage des entrées utilisateur**

name = input('Please state your name: ')

**Opérations numériques Exemple**

Nous pouvons utiliser des variables pour effectuer toutes sortes d'opérations, y compris des opérations mathématiques. Si vous les utilisez avec une input()fonction, soyez prudent : elles input()renvoient toujours une chaîne de caractères ; vous devez donc convertir cette chaîne en entier si vous souhaitez effectuer une opération mathématique avec elle.

number = input('Multiply me by three: ')

number \* 3

>> '444' # incorrect

int(number) \* 3

>> 12 # correct

**Exercice**

Analysez le code ci-dessous et anticipez le résultat. Vérifiez les résultats dans votre shell Python.

age = input("How old are you? ")

print(f"You are {age} years old")