

Variables et types de données

Table des matières

I. Variables	3
II. Exercice : Quiz	4
III. Types	5
IV. Exercice : Quiz	6
V. Notre premier programme	7
VI. Exercice : Quiz	9
VII. Essentiel	10
VIII. Auto-évaluation	10
A. Exercice	10
B. Test	10
Solutions des exercices	11

I. Variables

Durée : 1 h

Environnement de travail : interpréteur de commandes dans la console

Contexte

Une variable Python est un emplacement mémoire réservé pour stocker des valeurs. En Python, nous n'avons pas besoin de spécifier le type de variable car Python est un langage d'inférence et suffisamment intelligent pour obtenir le type de variable. Néanmoins il est important d'être capable d'identifier les différents types de données.

Dans cette unité vous allez découvrir la notion de typage. Vous verrez ensuite que toutes les valeurs possèdent un type de données, il en existe plusieurs :

- Les entiers (*integer*)
- Les nombres à virgules (*float*)
- Les chaînes de caractères (*string*)
- Les listes (*list*)
- Les nombres complexes (*complex*)
- Les booléens (*bool*)

Vous pourrez stocker ces valeurs dans des variables pour les réutiliser plus tard. Une fois que vous maîtriserez ces notions, vous pourrez écrire votre premier programme. Nous ne traiterons pas tous les types ici, seulement les trois principaux, integer, float et string. Ce cours est une initiation aux types de données.

Définition

Les variables sont l'un des concepts qui se retrouvent dans la majorité (et même, en l'occurrence, la totalité) des langages de programmation. Autant dire que sans variable, on ne peut pas programmer, et ce n'est pas une exagération.

Une variable est comme une boîte dans la mémoire de l'ordinateur où vous pouvez stocker une valeur unique. Si vous souhaitez utiliser le résultat d'une expression évaluée ultérieurement dans votre programme, vous pouvez l'enregistrer dans une variable. Considérez une variable comme une boîte étiquetée où une valeur est placée.

Une variable est un mot qui commence par une lettre minuscule ou majuscule et qui ne contient que des lettres, des chiffres et le caractère *underscore* « `_` ». Par convention, les variables, attributs et fonctions ne contiennent que des lettres minuscules et éventuellement des chiffres. Un bon nom de variable décrit les données qu'il contient. Imaginez que vous déménagez dans une nouvelle maison et étiquetez toutes vos boîtes « *trucs*. » Vous ne trouverez jamais rien !

S'ils sont composés de plusieurs mots, ces derniers sont séparés par des caractères soulignés.

```
1 >>> ma_variable
```

Les noms de variables sont sensibles à la casse, ce qui signifie que « *spam* », « *SPAM* », « *Spam* » et « *sPaM* » sont quatre variables différentes. Comme nous l'avons dit, il est d'usage en Python de démarrer vos variables avec une lettre minuscule.

Vous pouvez aussi bien utiliser les « *snake_case* » pour les noms de variables avec le tiret du 8 (« `_` »/underscore). C'est-à-dire que les variables peuvent ressembler à : `look_like_this`.

L'affectation

Vous stockerez les valeurs dans des variables avec une instruction d'affectation. Une déclaration d'affectation se compose d'un nom de variable (conteneur), d'un signe égal (appelé opérateur d'affectation) et la valeur (contenu) à stocker.

```
1 >>> ma_variable = 42
```

Une variable est initialisée (ou créée) la première fois qu'une valeur y est stockée. Après cela, vous pouvez l'utiliser dans des expressions avec d'autres variables et valeurs.

Aucun mot-clé n'est nécessaire, ni aucune déclaration préalable : nous sommes dans un langage à typage fort dynamique. Le même nom de variable peut être utilisé plus loin pour décrire une variable d'un type différent. Le type de la variable n'est pas déterminé par le conteneur mais par le contenu.

Une variable peut contenir une opération plus complexe.

```
1 >>> ma_variable = 42 + 3
```

Elle peut également contenir une autre variable.

```
1 >>> ma_variable2 = ma_variable1 * 2.0
```

Rappel

Il ne faut retenir que deux choses. En Python tout est objet, ci-dessous l'exemple 1 est un entier et l'exemple 2 est un flottant. Vous pouvez vérifier le type de données contenues dans une variable avec la commande `type()`.

```
1 #Exemple 1
2 >>> type(ma_variable1)
3 <class 'int'>
4 #Exemple 2
5 >>> type(ma_variable2)
6 <class 'float'>
```

Lorsqu'une variable reçoit une nouvelle valeur, l'ancienne valeur est oubliée.

```
1 >>> legumes = 3
2 >>> print(legumes)
3 3
4 >>> legumes = 5
5 print(legumes)
6 5
```

Exercice : Quiz

[solution n°1 p.13]

Question 1

Lequel de ces types est valide en Python ?

- ☐ Integer
- ☐ Collection
- ☐ Itérable

Question 2

Les variables Python sont sensibles à la casse.

- ☐ Vrai
- ☐ Faux

Question 3

On peut concaténer une variable de type str et une variable de type integer.

- ☐ Vrai
- ☐ Faux

Question 4

Par quoi est déterminé le type de la variable ?

- ☐ Le conteneur
- ☐ Le nom du type
- ☐ Le contenu

Question 5

La norme snake_case est acceptée pour donner un nom à une variable Python.

- ☐ Vrai
- ☐ Faux

III. Types

Typage

Vous rencontrerez différents types de données dans les programmes Python. Vous aurez parfois besoin de les analyser et de les convertir pour effectuer certaines opérations.

Le typage faible versus le typage fort est une notion sans grand intérêt puisque quasiment tous les langages sont à typage fort. Un typage faible n'accorde de l'importance qu'au contenu tandis qu'un typage fort accorde également de l'importance au type.

Par exemple, il n'est pas possible de concaténer deux types de données différentes puisque Python est un langage à typage fort.

```
1 >>> a="toto"
2 >>> b=5
3 >>> print a+b
4 Traceback (most recent call last):
5   File "<stdin>", line 1, in <module>
6 TypeError: cannot concatenate 'str' and 'int' objects
```

Le typage est dit dynamique, la vérification du type se fait au moment de l'exécution du programme. Aussi, les variables dynamiques peuvent changer de type au cours du programme. Python est un langage à typage dynamique, lui permettant une plus grande souplesse (modifier le type d'une variable au cours de l'exécution, par exemple).

Type de données

Toutes les valeurs ont un type de données, il s'agit d'une catégorie de valeur et chaque valeur appartient à une seule catégorie de données.

Les valeurs - 2 et 30 sont des valeurs entières. Le type de données entier (ou integer) indique des valeurs qui sont des nombres entiers.

```
1 -2, -1, 0, 1, 2
```

Les nombres avec un point décimal, tels que 3.14, sont appelés nombres à virgule flottante (ou float).

```
1 -1.25, -1.0, 0.0.5, 1.0, 1.25
```

Notez que même si la valeur 42 est un entier, la valeur 42.0 sera un nombre à virgule flottante.

Les programmes Python peuvent également avoir des valeurs de texte appelées chaînes ou chaînes de caractères (string). Entourez toujours votre chaîne de caractères entre guillemets simples (') ou doubles (") afin que Python sache où la chaîne commence et se termine.

```
1 'a', 'Hello', '42'
```

Vous pouvez vérifier le type de données grâce à la commande : type()

```
1 >>> type('pomme')
2 <class 'str'>
3 >>> type(6)
4 <class 'int'>
```

Remarque

Python ne peut pas concaténer une chaîne de caractères et un entier puisqu'il s'agit d'un langage à typage fort.

Vous pouvez même avoir une chaîne sans caractères : '', appelée « *chaîne vide* ». Si vous voyez le message d'erreur « *SyntaxError : EOL* » lors de l'analyse de la chaîne, vous avez probablement oublié le dernier caractère de guillemet à la fin de la chaîne.

```
1 >>> legume = 'pomme
2 File "<stdin>", line 1
3     legume = 'pomme
4                 ^
5 SyntaxError: EOL while scanning string literal
```

Exercice : Quiz

[solution n°2 p.14]

Question 1

Un langage à typage dynamique peut :

- ☐ Changer le type d'une variable lors de l'exécution
- ☐ Concaténer des variables de types différents
- ☐ Attribuer des types à la volée

Question 2

Les nombres avec un point décimal sont considérés comme étant des éléments de type float.

- ☐ Vrai
- ☐ Faux

Question 3

Si j'oublie les guillemets pour une variable de type str, Python détecte quand même le bon type.

- ☐ Vrai
- ☐ Faux

Question 4

Comment déclarer une chaîne de caractères vide ?

- ☐ En utilisant les guillemets sans rien dedans
- ☐ En ne notant rien après l'opérateur =
- ☐ En notant null après l'opérateur =

Question 5

La fonction `type()` permet de connaître le type d'une variable.

- ☐ Vrai
- ☐ Faux

V. Notre premier programme

Écriture d'un programme

Dans un contexte professionnel ces fonctionnalités vous permettront de tester votre code et de le rendre plus compréhensible aux autres développeurs.

La fonction `print()`

La fonction `print()` affiche la valeur de chaîne entre parenthèses à l'écran.

La ligne `print("Bonjour")` signifie « *Imprimez le texte dans la chaîne 'Bonjour'* ».

Lorsque Python exécute cette ligne, vous dites que Python appelle la fonction `print()` et la valeur de la chaîne est passée à la fonction. Une valeur transmise à un appel de fonction est un **argument**.

Vous remarquerez que les apostrophes ne sont pas imprimées à l'écran puisqu'elles marquent juste où la chaîne commence et se termine ; elles ne font pas partie de la valeur de la chaîne.

```
1 # Ce programme dit bonjour et demande mon prénom
2 print("Bonjour")
3 print("Comment t'appelles-tu ?") #Demander le prénom
```

La fonction `input()`

La fonction `input()` attend que l'utilisateur tape du texte sur le clavier et appuie sur « *Entrée* ». Cet appel de fonction évalue une chaîne de caractères égale au texte que l'utilisateur a renseigné et l'affecte à la variable `myName`. Vous pouvez considérer l'appel de fonction `input()` comme une expression correspondant à la chaîne de caractères que l'utilisateur a tapée.

```
1 my_Name = input()
```

Complément

L'appel `print()` suivant contient en fait l'expression « *Je suis ravi de te rencontrer,'+ myName'* » entre les parenthèses. N'oubliez pas que les expressions peuvent toujours correspondre à une seule valeur. Si « *votre prénom* » est la valeur stockée dans `myName` sur la ligne précédente, cette expression évalue « *Il est bon de vous rencontrer, prénom* ». Cette valeur de chaîne unique est ensuite transmise à `print()`, qui l'imprime à l'écran.

```
1 print("Je suis ravi de te rencontrer, " + myName)
```

La fonction len()

Vous pouvez transmettre à la fonction `len()` une chaîne de caractères (ou une variable contenant une chaîne de caractères), et la fonction évalue le nombre de caractères dans cette chaîne.

Tout comme dans cet exemple, `len(myName)` est évalué en entier. C'est alors passé à `print()` pour être affiché sur l'écran. Notez que `print()` permet de passer des valeurs entières ou des chaînes de caractères.

Vous ne pouvez pas utiliser l'opérateur « + » pour concaténer un entier à une chaîne de caractères car cela n'est pas grammaticalement correct en Python. Vous pouvez résoudre ce problème en convertissant la donnée.

```
1 print("Ton prénom contient " + str(len(myName)) + " caractères")
2 print("Quel âge as-tu ?") # Demander l'âge
3 my_Age = input()
```

Les fonctions de type casting

Si vous souhaitez concaténer un entier tel que 26 avec une chaîne de caractères avec `print()`, vous devrez obtenir la valeur '26', qui est la chaîne de caractères de 26. La fonction `str()` peut recevoir un entier et l'évaluera comme une chaîne de caractères.

Parce que `str(26)` est évalué à '26', l'expression « `J'ai` » + `str(26)` + 'ans' donne : « `J'ai` » + « 26 » + 'ans', lequel, à son tour évalué à « `J'ai 26 ans` ». Il s'agit de la valeur finale transmise à la fonction `print()`.

Les fonctions `str()`, `int()` et `float()` évalueront la chaîne de caractères, l'entier et le chiffre à virgule de la valeur que vous passez.

Les exemples précédents appellent les fonctions `str()`, `int()` et `float()` et leur transmettent des valeurs d'autres types de données pour obtenir une chaîne de caractères, un entier et un chiffre à virgule.

La fonction `str()` est pratique lorsque vous avez un entier ou un chiffre à virgule que vous voulez concaténer à une chaîne de caractères.

La fonction `int()` est également utile si vous avez un nombre comme chaîne de caractères que vous souhaitez utiliser dans une opération mathématique.

La fonction `input()` renvoie toujours une chaîne de caractères même si l'utilisateur entre un nombre. La valeur stockée dans le `myAge` n'est pas l'entier 26 mais la chaîne '26'. Si vous voulez faire des calculs en utilisant la valeur de `myAge`, utilisez la fonction `int()` pour obtenir un entier de `myAge` puis stockez-la en tant que nouvelle variable `myAge`.

Vous devriez maintenant pouvoir traiter la variable `my_Age` comme un entier à la place d'une chaîne de caractères.

Notez que si vous passez une valeur à `int()` qu'elle ne peut pas évaluer comme un entier, Python affichera un message d'erreur: `ValueError: invalid literal for int()`.

La fonction `int()` est également utile si vous devez arrondir un chiffre à virgule vers le bas.

Dans votre programme, vous avez utilisé les fonctions `int()` et `str()` pour obtenir un type de données approprié pour l'opération mathématique et la concaténation.

La variable `myAge` contient la valeur renvoyée par `input()`, comme la fonction `input()` renvoie toujours une chaîne de caractères (même si l'utilisateur a tapé un nombre), vous pouvez utiliser la fonction `int(myAge)` pour renvoyer un entier dans `myAge`.

Cette valeur entière est ensuite incrémentée de 1 dans l'expression `int(myAge) + 1`. Le résultat de cette opération est passé à la fonction `str()`.

La valeur de chaîne de caractères est renvoyée pour être concaténée avec les chaînes "Tu seras âgé de" et "ans dans un an" pour évaluer une nouvelle chaîne de caractères qui sera finalement passée à `print()` pour être affichée à l'écran.

```
1 print("Tu seras âgé de " + str(int(myAge)+1) + " ans dans un an")
```


Commenter son code

Python ignore les commentaires et vous pouvez les utiliser pour écrire des notes ou vous rappelez ce que le code essaie de faire. Le texte qui suit un dièse « # » fait partie d'un commentaire.

Parfois, les programmeurs mettent un « # » devant une ligne de code pour la supprimer temporairement lors du test d'un programme. Cela s'appelle commenter, cette pratique est utile lorsque vous essayez de comprendre pourquoi un programme ne fonctionne pas. Vous pouvez supprimer le « # » plus tard lorsque vous êtes prêt à remettre la ligne dans votre code.

Python ignore également la ligne vierge après le commentaire. Vous pouvez en ajouter autant que vous le souhaitez. Cela peut rendre votre code plus facile à lire, comme les paragraphes d'un livre.

Rappel

Si vous essayez d'utiliser l'opérateur « + » sur une chaîne et une valeur entière, Python ne saura pas comment le gérer et il affichera un message d'erreur.

```
1 MyAge = "Aujourd'hui j'ai"  
2 MyAge + 30  
3 TypeError: can only concatenate str (not "int") to str
```

Exercice : Quiz

[solution n°3 p.14]

Question 1

Quel est le nom d'une chaîne passée à une fonction ?

- ☐ Un argument
- ☐ Un paramètre
- ☐ Un exposant

Question 2

Lors d'un appel à la fonction print(), la valeur est imprimée avec ses guillemets.

- ☐ Vrai
- ☐ Faux

Question 3

À quoi sert la fonction input() ?

- ☐ À recueillir la saisie de l'utilisateur
- ☐ À enregistrer une chaîne de texte dans la mémoire
- ☐ À recueillir la liste des différents périphériques entrant

Question 4

La fonction len() permet d'obtenir la longueur d'une chaîne de caractères.

- ☐ Vrai
- ☐ Faux

Question 5

Il est possible de changer le type d'une variable en la définissant de cette manière : `maVariable = (int) '26'`.

- ☐ Vrai
- ☐ Faux

VII. Essentiel

Pendant ce cours nous avons vu comment définir des variables et nous avons effectué une brève introduction aux typages sur Python. Vous avez pu voir les trois principaux types sur Python. Les strings qui représentent les textes ou les chaînes de caractères, les integer qui représentent les chiffres ou nombre entier et enfin les float qui représentent les nombres décimaux.

Vous avez également pu voir que l'on peut imprimer la valeur des variables à l'écran ou encore demander des saisies à l'utilisateur grâce aux fonctions `print()` et `input()` respectivement. Enfin vous avez vaguement abordé la notion de type casting grâce aux fonctions `str()`, `int()` et `float()` qui permettent de convertir la variable passée en argument dans le type indiqué par le nom de la fonction.

Vous êtes maintenant capable de rédiger des programmes basiques en Python.

VIII. Auto-évaluation

A. Exercice

Vous venez de voir les notions de base nécessaire à l'écriture d'un programme basique en Python. Vous allez donc maintenant rédiger un programme simple en Python. Le but de ce programme est de calculer la température en degrés Celsius depuis une saisie en Fahrenheit.

Question 1

[solution n°4 p.16]

La première étape du programme est de demander une saisie à l'utilisateur pour récupérer la température en Fahrenheit. Comment allez-vous organiser cette étape ?

Question 2

[solution n°5 p.16]

La seconde étape du programme est d'effectuer le calcul et d'afficher la température en degrés Celsius, comment allez-vous organiser cela ?

B. Test

Exercice 1 : Quiz

[solution n°6 p.16]

Question 1

Quels sont les trois types de données principaux ?

- ☐ Entier
- ☐ Chaînes de caractères
- ☐ Chiffre à virgule
- ☐ Variable
- ☐ Expression

Question 2

De quoi est composée une expression ?

- ☐ Une valeur
- ☐ Deux valeurs
- ☐ Un opérateur
- ☐ Une variable

Question 3

Quels sont les noms de variables valides ?

- ☐ 0
- ☐ 'spam'
- ☐ spam
- ☐ spam_spam

Question 4

Qu'est-ce qu'une déclaration de variable ?

- ☐ Une déclaration permet de stocker une donnée dans une variable
- ☐ Une déclaration permet d'imprimer le résultat
- ☐ Une déclaration permet de modifier le type de données

Question 5


La valeur de retour de la fonction len() est systématiquement un integer.

- ☐ Vrai
- ☐ Faux

Solutions des exercices


Exercice p. 4 Solution n°1**Question 1**

Lequel de ces types est valide en Python ?

- ☒ Integer
- ☐ Collection
- ☐ Itérable
-  C'est le type integer qui est valide en Python, les autres types sont issues de Java.


Question 2

Les variables Python sont sensibles à la casse.

- ☒ Vrai
- ☐ Faux
-  Les noms de variables sont sensibles à la casse, ce qui signifie que « *spam* », « *SPAM* », « *Spam* » et « *sPaM* » sont quatre variables différentes.


Question 3

On peut concaténer une variable de type str et une variable de type integer.

- ☐ Vrai
- ☒ Faux
-  On ne peut pas les concaténer car Python est un langage à fort typage dynamique.


Question 4

Par quoi est déterminé le type de la variable ?

- ☐ Le conteneur
- ☐ Le nom du type
- ☒ Le contenu
-  Le type de la variable est déterminé par le contenu comme dans tous les langages typés dynamiquement.

Question 5


La norme snake_case est acceptée pour donner un nom à une variable Python.

- ☒ Vrai
- ☐ Faux
-  La norme snake_case est valide comme nom de variable.

Exercice p. 6 Solution n°2


Question 1

Un langage à typage dynamique peut :

- ☒ Changer le type d'une variable lors de l'exécution
- ☐ Concaténer des variables de types différents
- ☐ Attribuer des types à la volée
-  Il peut changer le type d'une variable lors de l'exécution du programme qui la contient.


Question 2

Les nombres avec un point décimal sont considérés comme étant des éléments de type float.

- ☒ Vrai
- ☐ Faux
-  Les éléments avec une décimale sont considérés comme étant de type float.


Question 3

Si j'oublie les guillemets pour une variable de type str, Python détecte quand même le bon type.

- ☐ Vrai
- ☒ Faux
-  Python détectera une erreur car il sera incapable de déterminer le type de la variable.


Question 4

Comment déclarer une chaîne de caractères vide ?

- ☒ En utilisant les guillemets sans rien dedans
- ☐ En ne notant rien après l'opérateur =
- ☐ En notant null après l'opérateur =
-  Il faut utiliser les guillemets sans rien à l'intérieur pour déclarer une chaîne de caractères vide.

Question 5

La fonction type() permet de connaître le type d'une variable.


- ☒ Vrai
- ☐ Faux
-  La fonction type() permet de connaître le type d'une variable.

Exercice p. 9 Solution n°3

Question 1

Quel est le nom d'une chaîne passée à une fonction ?


- ☒ Un argument
- ☐ Un paramètre
- ☐ Un exposant

 La variable contient une chaîne ou tout autre type de contenu, qui est passée à une fonction et est appelée « un argument ».

Question 2

Lors d'un appel à la fonction `print()`, la valeur est imprimée avec ses guillemets.


- ☐ Vrai
- ☒ Faux

 Lors d'un appel à la fonction `print()`, le résultat est toujours imprimé sans guillemets.

Question 3

À quoi sert la fonction `input()` ?


- ☒ À recueillir la saisie de l'utilisateur
- ☐ À enregistrer une chaîne de texte dans la mémoire
- ☐ À recueillir la liste des différents périphériques entrant

 La fonction `input()` sert à recueillir la saisie de l'utilisateur.

Question 4

La fonction `len()` permet d'obtenir la longueur d'une chaîne de caractères.


- ☒ Vrai
- ☐ Faux

 Cette fonction retourne le nombre de caractères composant une chaîne.

Question 5

Il est possible de changer le type d'une variable en la définissant de cette manière : `maVariable = (int) '26'`.

- ☐ Vrai
- ☒ Faux

 Pour changer le type d'une variable, par exemple passer de string à integer, il faudra utiliser la fonction `int()` pour convertir une chaîne de caractères en integer.

p. 10 Solution n°4

Pour être très clair avec l'utilisateur, il faudra commencer par lui demander de saisir la température en Fahrenheit, nous allons donc commencer par utiliser la fonction `print()` pour demander explicitement l'information que cherche notre programme.

```
1 print('Bonjour, indiquez ta température en Fahrenheit')
```

La deuxième étape consiste à permettre à l'utilisateur la saisie d'une information que nous allons stocker dans une variable pour la réutiliser plus tard.

```
1 Temperature_F = input()
```

p. 10 Solution n°5

Nous allons créer une variable `Temperature C` pour stocker la valeur en Celsius après le calcul de conversion.

```
1 Temperature_C = (Temperature_F-32) * 5/9
```

Nous avons maintenant la valeur en Celsius il suffit de l'afficher. Vous savez le faire grâce à `print()`. Ici, petite difficulté, la valeur à afficher est un integer et le texte que nous allons mettre avant, une string. Comme vous le savez il est impossible de concaténer une string et un integer, nous allons donc utiliser le type casting.


```
1 print("La température est de " + str(Temperature_C) + " degrés Celsius")
```

Exercice p. 10 Solution n°6

Question 1

Quels sont les trois types de données principaux ?


- ☒ Entier
- ☒ Chaînes de caractères
- ☒ Chiffre à virgule
- ☐ Variable
- ☐ Expression

 Les trois types de données principaux sont les entier (integer), les chaînes de caractères (string), les chiffres à virgules (float).

Question 2

De quoi est composée une expression ?


- ☐ Une valeur
- ☒ Deux valeurs
- ☒ Un opérateur
- ☐ Une variable

 Une expression est composée d'un opérateur et de deux valeurs.

Question 3


Quels sont les noms de variables valides ?

- ☐ 0
- ☐ 'spam'
- ☒ spam
- ☒ spam_spam

 Le chiffre 0 n'est pas valide et le nom spam entre guillemets non plus, on ne met pas de caractères spéciaux dans un nom de variable.


Question 4

Qu'est-ce qu'une déclaration de variable ?

- ☒ Une déclaration permet de stocker une donnée dans une variable
- ☐ Une déclaration permet d'imprimer le résultat
- ☐ Une déclaration permet de modifier le type de données
-  ☐ Une déclaration permet d'attribuer une valeur à une variable.

Question 5

La valeur de retour de la fonction len() est systématiquement un integer.

- ☒ Vrai
- ☐ Faux
-  Cette fonction retourne toujours la longueur d'une chaîne de caractères, donc un chiffre ou un nombre entier, elle retourne donc systématiquement un integer.