

| | | |
|----------------------|---------------|----------------|
| Dernière mise à jour | Informatique | Denis DEFAUCHY |
| 30/03/2021 | 4 – Variables | Cours |

Informatique

4

Variables

Cours

| | | |
|----------------------|---------------|----------------|
| Dernière mise à jour | Informatique | Denis DEFAUCHY |
| 30/03/2021 | 4 – Variables | Cours |

| | |
|--|-----------|
| Les variables | 3 |
| 1.I. Noms des variables | 3 |
| 1.II. Variables inexistantes | 5 |
| 1.III. Création de variables classiques | 5 |
| 1.III.1 Variables de type entiers et décimaux | 5 |
| 1.III.2 Variable de type chaînes de caractères | 6 |
| 1.III.3 Variables de type Booléens | 8 |
| 1.III.4 La variable « Rien » | 9 |
| 1.III.5 Vérification du type d'une variable | 9 |
| 1.IV. Création d'une variable via demande à l'utilisateur | 10 |
| 1.V. Visualiser les variables créées | 11 |
| 1.VI. Echanger deux variables..... | 11 |
| 1.VII. Effacer des variables | 11 |

| | | |
|----------------------|---------------|----------------|
| Dernière mise à jour | Informatique | Denis DEFAUCHY |
| 30/03/2021 | 4 – Variables | Cours |

Les variables

Taper 10+10, c'est bien, mais on le fera très rarement dans un programme. Nous allons manipuler des variables dans lesquelles sont stockées des valeurs.

1.1. Noms des variables

Une variable est un « mot », c'est-à-dire une chaîne de caractères qui doit commencer par une lettre, sans espaces, en évitant les caractères spéciaux à part underscore « _ », on évitera les accents, à laquelle on associe une valeur, un mot entre guillemets etc. Elle est définie en inscrivant le nom de celle-ci, puis le signe « = » et enfin ce qu'elle doit contenir. Lorsqu'elle est définie, il suffit de taper son nom pour obtenir sa valeur :

```
>>> a1 = 2
>>> 1a = 2
File "<console>", line 1
1a = 2
^
SyntaxError: invalid syntax
```

| Programme | Console lors de l'appel des variables après exécution du programme |
|--|---|
| <pre>a = 1 Nom = "Denis" Variable_1 = 10</pre> | <pre>>>> Nom 'Denis' >>> a 1 >>> Variable_1 10</pre> |

Il est fortement recommandé de choisir des noms correspondant à ce que contient la variable pour plusieurs raisons :

- De base, vous serez évaluées en partie à l'écrit, vos programmes doivent être compréhensibles
- D'une manière générale, vos codes doivent être lisibles pour d'autres personnes, il faut donc qu'ils soient facilement compréhensibles
- Lorsque vous travaillerez sur de gros programmes, vous oublierez vite ce que vous avez fait quelques semaines avant et si vous ne respectez pas ce conseil, vous pourriez avoir besoin de repasser beaucoup de temps à comprendre ce que vous avez fait

Il faudra donc par exemple, préférer :

- « Nom_Eleve » plutôt que « n »
- « Coeff_Conductivité » plutôt que « mu »
- « Nb_Eleves » plutôt que « n »

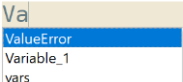
Cela permettra d'utiliser la « n » dans un autre contexte car vous n'auriez pas pu l'utiliser 2 fois...

Attention, **majuscules et minuscules sont différents** dans python. Si la variable *a* existe, *A* n'existe pas et peut avoir une valeur différente.

Lorsqu'une variable existe et qu'on la redéfinit, on écrase sa valeur précédente.

Vous pourrez remarquer que dès qu'une variable existe dans le Workspace, Pyzo permet en tapant ses quelques premières lettres de l'écrire entièrement, c'est assez pratique et ça évite souvent une faute de frappe pour les longs noms. Il suffit alors de cliquer sur le nom de la variable ou d'appuyer sur la touche « Tabulation ».

```
Variable_1 = 2
```



Remarque : double cliquer quelque part sur le nom d'une variable permet de la sélectionner

| | | |
|----------------------|---------------|----------------|
| Dernière mise à jour | Informatique | Denis DEFAUCHY |
| 30/03/2021 | 4 – Variables | Cours |

Attention, danger ! **Il ne faut surtout pas utiliser de nom de fonctions python** comme noms de variables, prenons les exemples de input (fonction abordée un peu plus loin) et print

Il peut vous arriver d'écrire :

`a = input = ("Valeur de n")` au lieu de `a = input("Valeur de n ? ")`

Ou encore :

`print = 2` au lieu de `print(2)`

Ce sont de grosses erreurs qui ont des conséquences importantes. Supposons avoir écrit ces deux erreurs et avoir exécuté le code en question, puis appelons les fonctions input et print :

| | |
|--|--|
| <code>print(2)</code> | Traceback (most recent call last): File "<console>", line 1, in <module> TypeError: 'int' object is not callable |
| <code>b = input("Valeur de i ? ")</code> | Traceback (most recent call last): File "<console>", line 1, in <module> TypeError: 'str' object is not callable |

Il faut savoir qu'écrire **mot()** consiste à appeler une fonction mot (nous aborderons les fonctions plus tard) en précisant qu'elle n'a pas d'argument (parenthèse vide). Ecrire **mot(2)** appelle la fonction mot avec comme argument la valeur 2. Or, en écrivant les lignes d'erreurs `print=2` et `input=2`, on a défini des variables de type int dont les noms sont print et input. Pyzon nous informe donc que ces variables de type int ne sont pas appelables (callable) car ce ne sont pas des fonctions, et c'est logique.

Il faut donc veiller à éviter d'appeler ses variables avec un nom de fonction native de Python, et souvent mettre une majuscule le permet. Par exemple, la fonction « max » existe, alors créer un maximum en écrivant « Max= » avec majuscule permet de ne pas écraser « max ».

Si vous écrasez une fonction, la solution consiste à supprimer ces variables qui ont le nom de fonctions dans python avec la commande del :

```
del input
del print
```

Ce qui supprime les variables créées par erreur avec des noms de fonctions existantes dans python pour pouvoir à nouveau les utiliser.

Attention donc à ne pas appeler des variables avec des noms de fonctions et à savoir utiliser del si vous l'avez fait par erreur

Heureusement, on ne peut pas modifier la fonction « del » ☺

```
>>> del = 3
      File "<console>", line 1
        del = 3
          ^
```

Remarque : une autre solution consiste à fermer Pyzo, ce qui vide le « Workspace »

| | | |
|----------------------|---------------|----------------|
| Dernière mise à jour | Informatique | Denis DEFAUCHY |
| 30/03/2021 | 4 – Variables | Cours |

1.II. Variables inexistantes

Il faut bien retenir que l'ordinateur ne connaît que ce qu'on définit.

Pour le moment, nous avons créé les variables *a*, *b*, *c* et quelques autres avec des noms plus grands.

Tapez *z* puis pressez Entrée dans la console. Un message d'erreur apparaît :

```
>>> z
Traceback (most recent call last):
  File "<console>", line 1, in <module>
NameError: name 'z' is not defined
```

C'est normal, la variable *z* n'étant pas définie, Python nous le dit : « name 'z' is not defined ».

En général, vous aurez cette erreur pour des fautes de frappe dans les noms de variable, ou pour une majuscule intervertie avec une minuscule. Ex : *Nb_Eleves* - *Nb_Eleve* - *Nb_eleve* – *nb_Eleve*...

Un autre cas qui peut arriver, c'est un code qui contenait la création d'une variable « *a=1* », qui a ensuite disparue. Votre code fonctionne alors tant que la variable *a* est dans le Workspace. Mais si vous fermez Pyzo, et que vous l'ouvrez à nouveau, la variable ayant disparue de votre code, il y aura une erreur lorsque celui-ci fera appel à cette variable inexistante « *b = a + 1* ».

1.III. Création de variables classiques

1.III.1 Variables de type entiers et décimaux

Les deux types de variables qui nous intéresseront ici sont les types « integer » pour « entier » et « floating » pour décimal. Les types de variables se créent automatiquement en fonction du nombre entré.

Dans la console, tapez « *a = 10* » puis validez, ensuite tapez « *type(a)* » et valider.

```
>>> a=10
>>> type(a)
<class 'int'>
```

« *a* » est une variable de type « int » pour « integer » ou entier.

Tapez maintenant « *b=10.0* » et validez, puis « *type(b)* » et regardez le résultat :

```
>>> b=10.0
>>> type(b)
<class 'float'>
```

Le nombre vaut la même chose, mais c'est maintenant un « float » ou décimal.

On obtient un arrondi de *x* à *n* chiffres après la virgule en écrivant : *round(x, n)*

Une opération entre un entier et un décimal donnera toujours un nombre décimal. Tapez dans la console *c = a + b* puis *type(c)*, vous verrez :

| | | |
|----------------------|---------------|----------------|
| Dernière mise à jour | Informatique | Denis DEFAUCHY |
| 30/03/2021 | 4 – Variables | Cours |

```
>>> c=a+b
>>> type(c)
<class 'float'>
```

Il est possible de transformer/convertir un entier en décimal et inversement, un décimal en entier, soit prendre sa partie entière.

Essayez `float(10)`, il donnera 10.0 - Essayez `int(10.2)`, il donnera 10

1.III.2 Variable de type chaînes de caractères

Une chaîne de caractères est un mot composé de lettres. Il faut bien faire la différence entre une variable `z` et la lettre `z`, python ne les traite pas de la même manière.

Dans la console, si `z` n'est pas une variable existante (ie n'est pas visible dans le Workspace), écrivez `z` et tapez « Entrée », un message d'erreur dit :

```
>>> z
Traceback (most recent call last):
  File "<console>", line 1, in <module>
NameError: name 'z' is not defined
```

Effectivement, la variable `z` n'a pas été créée, et ne contient aucune donnée « à l'intérieur ».

Tapez maintenant, au choix :

```
'z' ; "z" ; '''z''' ; """z"""
```

Python renvoie dans tous les cas la lettre `z` entre apostrophes.

Définissons la lettre `z` dans la variable `d` : tapez `d = 'z'` puis tapez `type(d)`, vous verrez que `d` est un « string » ou chaîne de caractères. La variable `d` contient la lettre `z`.

Attention, lettres majuscules et minuscules ne sont pas identiques pour Python.

Il est possible de regrouper deux chaînes de caractère avec le symbole « + », par exemple :

```
>>> "Classe 1 " + "Eleve 1"
'Classe_1 Eleve 1'
```

Attention, lorsque l'on utilise deux apostrophes pour créer une chaîne de caractères, il est évidemment impossible d'utiliser cette même apostrophe dans la chaîne de caractère puisque celle-ci mettra fin à ladite chaîne :

```
>>> 'Nous l'avons utilisée.'
File "<console>", line 1
'Nous l'avons utilisée'
      ^
SyntaxError: invalid syntax
```

| | | |
|----------------------|---------------|----------------|
| Dernière mise à jour | Informatique | Denis DEFAUCHY |
| 30/03/2021 | 4 – Variables | Cours |

Dans ce cas, plusieurs solutions :

- Utiliser des guillemets pour encadrer la chaîne de caractères :

```
>>> "Nous l'avons utilisée"
"Nous l'avons utilisée"
```
- Utiliser l'anti slash (touches Alt+8) devant l'apostrophe qui doit rester du texte et non une délimitation :

```
>>> 'Nous l\'avons utilisée'
"Nous l'avons utilisée"
```

Allez, juste pour s'amuser, on peut même écrire :

```
""" Je l'avais dit: "C'est possible" """
```

Il est possible de récupérer une lettre en fonction de sa position dans une chaîne de caractères. Pour cela, il faut dans un premier temps définir une variable contenant cette chaîne de caractères puis de demander la lettre en question à l'aide de crochets en précisant la place, attention, en partant de 0 pour la première lettre :

```
>>> Nom = 'Denis'

>>> Nom[2]
'n'
```

Si on demande une lettre au-delà de la taille du mot enregistré dans la variable, ici Nom, l'erreur suivante apparaît :

```
>>> Nom[6]
Traceback (most recent call last):
  File "<console>", line 1, in <module>
IndexError: string index out of range
```

Pyzo nous prévient que l'index 6 est hors de portée (out of range).

Il est possible de créer une chaîne de caractères contenant à la fois des mots et la valeur d'une variable. Dans ce cas, il est obligatoire de dire à Python qu'il faut considérer la valeur de la variable comme une chaîne de caractères (équivalent à un problème d'homogénéité) à l'aide de la commande `str` afin de créer une chaîne de caractères comme somme de chaînes de caractères, qui sera affichée à l'aide de la fonction `print`.

Soit `a` une variable qui vaut 10, pour créer la phrase « Il y a 10 pommes » telle que si `a` change, le 10 change, il faut écrire :

```
>>> "Il y a " + str(a) + " pommes"
'Il y a 10 pommes'
```

La commande `str(a)` permet de transformer la valeur de `a` en un texte qui sera ajouté au reste du texte.

Erreur à ne pas commettre :

```
>>> "Il y a " + "a" + " pommes"
'Il y a a pommes'
```

| | | |
|----------------------|---------------|----------------|
| Dernière mise à jour | Informatique | Denis DEFAUCHY |
| 30/03/2021 | 4 – Variables | Cours |

Il est possible de créer un texte « à trous » dans lequel seront ajoutées des valeurs précisées à la fin :

```
>>> print("Mon nom est %s et j'ai %s ans" % ("Denis",30))
Mon nom est Denis et j'ai 30 ans
```

La structure est la suivante :

- Texte entre guillemets et %s pour chaque trou
- Ajout d'un % après le texte
- Parenthèse avec chaque élément à intégrer dans les trous, dans l'ordre d'apparition
- Attention : pour afficher le symbole % dans le texte, il faut le doubler en écrivant %%

Voire même, pouvoir en réutiliser :

```
>>> print('{0} {1}, \n[...] \nAurevoir cher {0} '.format('Monsieur','Defauchy'))
Monsieur Defauchy,
[...]
Aurevoir cher Monsieur
```

```
>>> 2 * 'Cou'
```

Remarque : Multiplier une chaîne de caractères par un entier la reproduit : 'CouCou'

1.III.3 Variables de type Booléens

Nous aborderons plus précisément les variables booléennes dans la suite du cours, dans l'algorithmique et les conditions.

Nous pouvons toutefois définir dès maintenant deux variables booléennes :

- « True » - Equivalent à « Juste », « Oui », « 1 »
- « False » - Equivalent à « Faux », « Non », « 0 »

Une variable booléenne peut être créée directement, par exemple « a = True », ou être le résultat d'un test, par exemple :

```
>>> a = 2==2
```

```
>>> a
True
```


| | | |
|----------------------|---------------|----------------|
| Dernière mise à jour | Informatique | Denis DEFAUCHY |
| 30/03/2021 | 4 – Variables | Cours |

1.III.4 La variable « Rien »

Lorsque l'exécution d'un code ne renvoie rien, en réalité, elle renvoie la variable « None ».

Lorsque l'on tape par exemple :

```
>>> a = 2
```

```
>>> 2+2
4
```

L'exécution 2+2 renvoie 4 alors que la commande a = 2 ne renvoie rien. En fait, cela renvoie None.

```
>>> None
```

```
>>>
```

Il sera donc possible, plus tard, de renvoyer None si un code ne doit rien renvoyer, de tester le résultat d'une commande en vérifiant s'il vaut None ou pas, de préremplir une matrice vide plutôt que pleine de zéros...

1.III.5 Vérification du type d'une variable

On peut vérifier qu'une variable est d'un type donné, par exemple :

```
>>> a = 1
```

```
>>> b = 1.2
```

```
>>> c = ""
```

```
>>> type(a) == int
True
```

```
>>> type(b) == int
False
```

```
>>> type(b) == float
True
```

```
>>> type(c) == str
True
```

| | | |
|----------------------|---------------|----------------|
| Dernière mise à jour | Informatique | Denis DEFAUCHY |
| 30/03/2021 | 4 – Variables | Cours |

1.IV. Création d'une variable via demande à l'utilisateur

Il est possible de faire en sorte qu'à un moment soit demandé à l'utilisateur d'entrer une variable par lui-même au cours d'un programme, pour cela il suffit d'utiliser la fonction input :

| Programme | Console après exécution |
|---|---|
| <pre>Nb_Eleves = input("Entrer le nombre d'élèves: ")</pre> | <pre>>>> (executing lines 1 to 3 of "Essai_1.py") Entrer le nombre d'élèves:</pre> |

Attention, quelle que soit la donnée entrée, le résultat d'un input est une chaîne de caractères (string).

Il suffit alors, dans la console, d'entrer le nombre associé puis validez avec la touche Entrée.

```
Entrer le nombre d'élèves: 10
```

Et enfin, vérifiez ce que contient la variable :

```
>>> Nb_Eleves
'10'
```

ATTENTION : la variable Nb_Eleves est une chaîne de caractères, on le voit à la présence des guillemets. Pour pouvoir l'utiliser dans des calculs, il faut la convertir en flottant par exemple, il faut donc ajouter obligatoirement :

```
>>> Nb_Eleves
'10'

>>> Nb_Eleves = float(Nb_Eleves)

>>> Nb_Eleves
10.0
```

Les erreurs classiquement rencontrées avec le « input » sont associées à l'oubli de transformation de la chaîne de caractères en nombre :

```
>>> a = input("Quel âge as tu ? ")
b = 2 + a
Quel âge as tu ? 10
Traceback (most recent call last):
  File "<console>", line 1, in <module>
TypeError: unsupported operand type(s) for +: 'int' and 'str'
>>> a = input("Quel âge as tu ?")
b = 2 * a
print(b)
Quel âge as tu ?32
3232
```

Remarques :

- Lors de l'utilisation d'un input, ajoutez « : » ou « ? » à la fin du message à l'utilisateur afin de voir clairement que l'ordinateur attend une entrée. Une erreur classique est de ne pas se rendre compte que l'ordinateur attend, et de lancer plusieurs fois le programme... Il faudra alors remplir autant de fois la console avec une valeur ! Ou « casser » le code avec les touches « Ctrl + i » autant de fois.
- La commande Input sera utilisée avec parcimonie, on lui préférera la création directe de variables dans le fichier de travail.

| | | |
|----------------------|---------------|----------------|
| Dernière mise à jour | Informatique | Denis DEFAUCHY |
| 30/03/2021 | 4 – Variables | Cours |

1.V. Visualiser les variables créées

Allez dans « Tools » puis cochez « Workspace ». Regardez en bas à droite, une nouvelle fenêtre « Workspace » est apparue et on y voit les variables existantes (noms, types, valeur).

| Name | Type | Repr |
|------------|-------|---------|
| Variable_1 | int | 10 |
| Nom | str | 'Denis' |
| b | float | 10.0 |
| a | int | 1 |

1.VI. Echanger deux variables

Sous Python, on a deux possibilités pour échanger deux variables :

| | |
|--------------------------------|------------------------|
| <pre>aa = a a = b b = aa</pre> | <pre>a, b = b, a</pre> |
|--------------------------------|------------------------|

Attention, si la solution de gauche fonctionne quel que soit le programme, celle de droite fonctionne sous Python. Quid des autres ...

1.VII. Effacer des variables

Rien de plus simple. Effaçons la variable d créée précédemment et visible dans l'explorateur de variables. Il suffit d'écrire « del a » et de valider pour la voir disparaître.