Les variables en Python

1) Qu'est-ce qu'une variable ?

C'est une information (une donnée) temporaire que l'on stocke dans une case de la RAM. On dit qu'elle est variable car la valeur peut changer pendant le déroulement du programme.

Une **variable** est caractérisée par 3 choses :
sa **valeur** : c'est la donnée qu'elle stocke
son **nom** : il permet de la reconnaître et d'accéder à son contenu.
Son **type** : quel genre de valeur peut-on ranger dedans. Un nombre, une lettre…

2) Instructions de base sur les variables.

Les instructions de base que l'on peut pratiquer avec une variable sont les suivantes :

La saisie (input) :

on demande à l'utilisateur du programme de donner la valeur de la variable : **Lire A .**
En python :
a = input()
Ou si on souhaite afficher un message d'invite :
a=input(''Quelle est la valeur de a ?'')

a sera égal à la saisie de l'utilisateur consécutive à l'affichage du message ''Quelle est la valeur de a ? ''.

L'affectation :

l'algorithme donne une valeur à la variable : c'est ici le programme qui affecte une valeur à la variable : **a prend la valeur 2**.

En python : **a=2**

Le symbole égal utilisé ici n'a rien à voir avec celui des mathématiques. Ce n'est pas une équation, mais une affectation : on donne à a la valeur 2.

L'affichage des résultats (print) :

on affiche la valeur de la variable : **Afficher A**.
En python : **print(a)**

Exemple :

Écrire un programme « film.py » qui vous demandera quel est votre film préféré, stockera cette valeur dans une variable qu'on appellera *film*, et affichera ensuite la valeur de *film* à l'écran.

```
film = input("Quel est  votre film préféré ? :")
print ("J'ai bien compris que votre film préféré est :")
print (film)
```

3) Opérations arithmétiques sur les variables.

a) Les opérations.

Un programme est bien sur capable d'effectuer des opérations mathématiques. Voila les symboles en Python des opérations les plus courantes :

addition : +
soustraction : -
multiplication : *
division : /
puissance **

Ouvrir l'interpréteur python (avec les >>>), et tester les instructions :

```
>>> a = 12
>>> b = 3
>>> somme = a + b
>>> print (somme)
```

Exemple : super pactole.

Au grand jeu du "super pactole qui rend trop riche", Thomas a misé 6 euros. Après tirage au sort, il apprend qu'il remporte 3.2 fois sa mise.

Écrire un programme qui commencera par affecter à la variable "mise" la valeur 6. Le programme renverra ensuite la somme gagnée par Thomas, diminuée de la mise (on pourra utiliser une seconde variable appelée gain).

```
mise = 6
gain = (mise * 3.2) - mise
print (gain)
```

b) Modifier une variable à partir de sa propre valeur.

Exemple : **a = a + 3**
On incrémente de 3 la valeur présente dans *a*

petit programme exemple : Le prix d'un article est fixé à 15€. Ce prix augmente de 3€. Comment calculer dans le programme le nouveau prix ?

```
>>> prix = 15
>>> prix = prix + 3
>>> print (prix)
```

--> on peut expliquer l'instruction "prix = prix + 3" par : prix devient prix + 3.

c) Modifier une variable à l'aide d'une autre variable.

Lors de la création d'une variable en Python, l'ordinateur va :

- créer et mémoriser un nom de variable
- créer et mémoriser une valeur
- établir un lien entre le nom de la variable et l'emplacement de la mémoire qui stocke la valeur.

Dans le 3)b), nous avons vu que nous pouvions modifier la valeur en mémoire. Il est aussi possible, sans changer le nom de la variable, de modifier l'emplacement de la mémoire vers lequel elle pointe. Commençons par taper :

```
>>> a = 1
>>> b = 5
>>> print (b)
5
```

Nous pouvons maintenant faire pointer "b" vers l'emplacement mémoire de "a" :

```
>>> b = a
>>> print (b)
1
```

Exemple
Le professeur de mathématiques a permuté les notes de Tom et Max.
Écrire un programme « permut-notes.py » qui commencera par demander la note erronée de Tom, celle de Max, remettra un peu d'ordre dans tout cela, et affichera au final :

Tom :
"valeur réelle de la note de Tom"
Max :
"valeur réelle de la note de Max"

Algorithme proposé par un pas malin :
LIRE noteTom
LIRE noteMax
noteTom = noteMax
noteMax = noteTom

Cet algo ne fonctionne pas ! La note de Max ne sera pas modifiée ! Il est nécessaire d'utiliser une « variable tampon ».

LIRE noteTom
LIRE noteMax
tampon = noteMax
noteMax = noteTom
noteTom = tampon

```
tom = input("note fausse de tom :")
max = input("note fausse de max :")

temp = tom
tom = max
max = temp

print ("nouvelle note de tom :"+tom)
print ("nouvelle note de max :"+max)
```