Variables

I/ En analyse

1) Encyclopédie

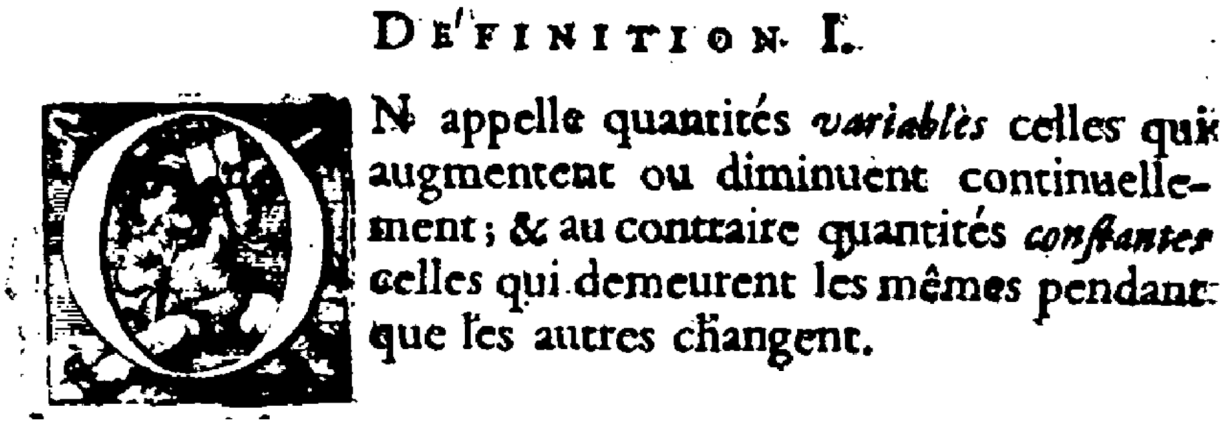
Dans l'encyclopédie de d'Alembert et Diderot, l'article *variable* dit :

> VARIABLE, adj. (*Alg.* & *Géom.*) on appelle *quantités variables* en Géométrie, les quantités qui varient suivant une loi quelconque. Telles sont les abscisses & les ordonnées des courbes, *&c*.
>
> On les appelle ainsi par opposition aux quantités constantes, qui sont celles qui ne changent point, comme le diamètre d'un cercle, *&c*.
>
> On exprime communément les variables par les dernières lettres de l'alphabet *x*, *y*, *z*.
>
> Quelques auteurs au-lieu de se servir de l'expression de *quantités variables*, disent des *fluentes*. *Voyez* FLUENTE& FLUXION.

Une relation entre les variables x et y telle que toute valeur de x est en relation avec une seule valeur de y, est une fonction. Par exemple la relation x×y=1 est la fonction inverse y=1/x.

2) Guillaume de l'Hospital

Dans son traité sur les dérivées[1], Guillaume de l'Hospital écrit :



DEFINITION I.

ON appelle quantités *variables* celles qui augmentent ou diminuent continuellement ; & au contraire quantités *constantes* celles qui demeurent les mêmes pendant que les autres changent.

En Python par contre, une variable n'évolue pas continuellement mais par sauts. Pour faire varier une variable, on lui injecte une nouvelle valeur.

1 *Analyse des infiniment petits, pour l'intelligence des lignes courbes* paru en 1696.

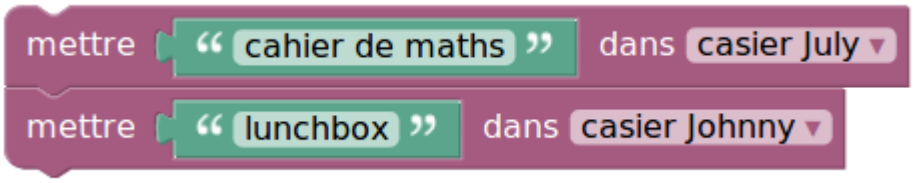
II/ Casiers

Dans un lycée américain, chaque élève a un casier à son nom ; casier dont le contenu varie lorsque l'élève y range des affaires ou en retire des affaires dont il a besoin. Les variables de Python fonctionnent de la même manière, avec un nom écrit sur le casier (ou la variable) et un contenu qui peut varier. Le nom de la variable est une constante : il ne varie pas, c'est le contenu qui varie.

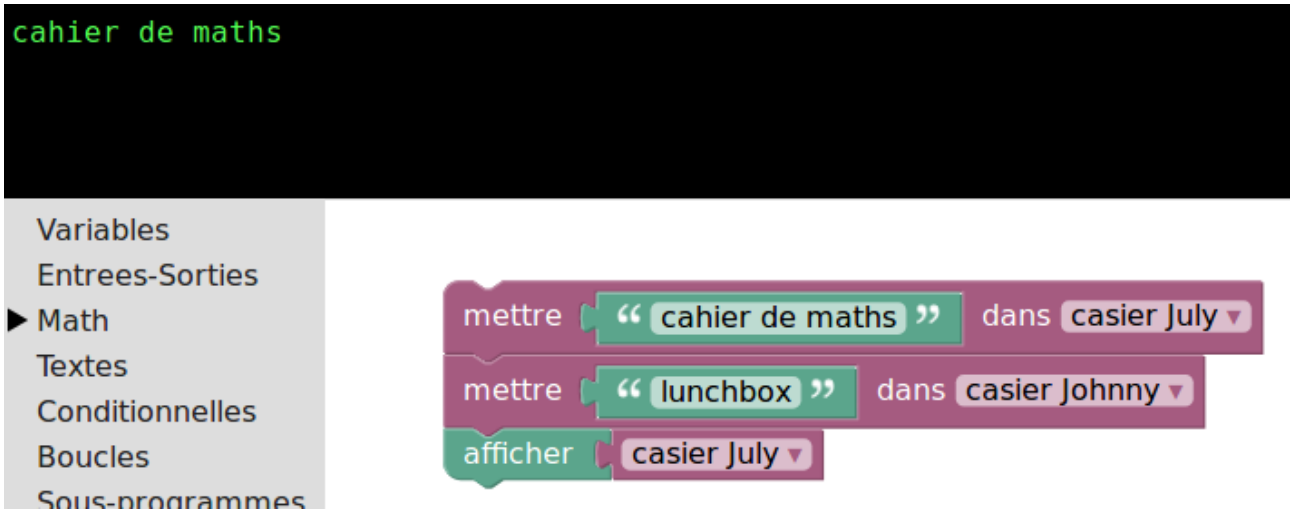
1) SofusPy974

L'outil SofusPy974 permet de produire aisément du code Python et du pseudo-code. Pour l'utiliser, aller sur https://alainbusser.github.io/SofusPy974/ à l'aide d'un navigateur Internet.

Johnny n'a pas de casier tant qu'on n'a pas mis quelque chose dedans. Pour ce faire, on va dans le menu des variables et là, on choisit « mettre … dans ... », puis on renomme la variable et on y met le lunchbox de Johnny :



À ce stade on a deux variables, représentant les casiers respectifs de Johnny et de July. Si on veut savoir ce qu'il y a dans le casier de July, on affiche celui-ci :



2) Python

Le bouton représentant deux serpents permet d'accéder à un éditeur Python :

En cliquant dessus, on a la traduction automatique du programme Sofus, en un programme écrit en

Python :

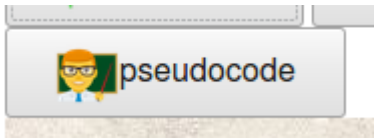
```
cahier de maths
```

```
1
2
3  casier_July = 'cahier de maths'
4  casier_Johnny = 'lunchbox'
5  print(casier_July)
6
```

On voit que « mettre dans » se traduit en Python, par un signe d'égalité, et que « afficher » se traduit par « print ».

3) Pseudo-code

SofusPy974 permet d'écrire mieux l'algorithme, en cliquant sur le bouton représentant un professeur :



L'écriture est plus claire pour un humain :

```
casier_July ← 'cahier de maths'
casier_Johnny ← 'lunchbox'
afficher (casier_July)
```

Pour donner une valeur à une variable, on injecte la valeur dans la variable, ce qui se fait en citant le nom de la variable.

III/ Expressions

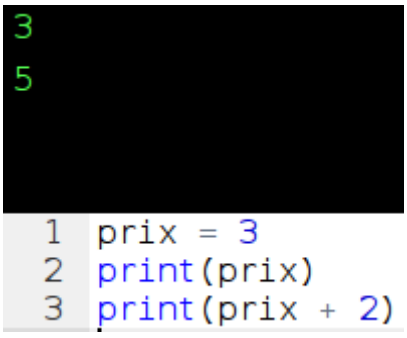
1) Affichage

Pour afficher le ***contenu*** d'une variable, on cite uniquement le ***nom*** de la variable :

Le script n'affiche pas le mot « prix » mais le nombre 3 (la constante qui est stockée dans la variable de nom `prix`). Cela est général : dans une expression algébrique, les mots représentent des variables et c'est le contenu de ces variables qui sera utilisé lorsqu'on veut évaluer l'expression. Par exemple si on veut savoir ce que vaut `prix+2`, on n'écrit pas « 2 de plus que le contenu de la variable `prix` » mais juste `prix+2` :

Noter que l'évaluation d'une expression comme prix+2 ne modifie pas la variable, elle a seulement été consultée pour l'évaluation, pas modifiée :

En Python il en est de même :



2) Compteurs

Ainsi, si on évalue l'expression `compteur+1`, cela n'incrémente pas le `compteur`, mais ne fait que ***calculer*** son successeur. Il reste encore à injecter `compteur+1` dans `compteur`, avec

```
compteur ← compteur + 1
```

Ainsi, les deux opérations suivantes ont le même effet :

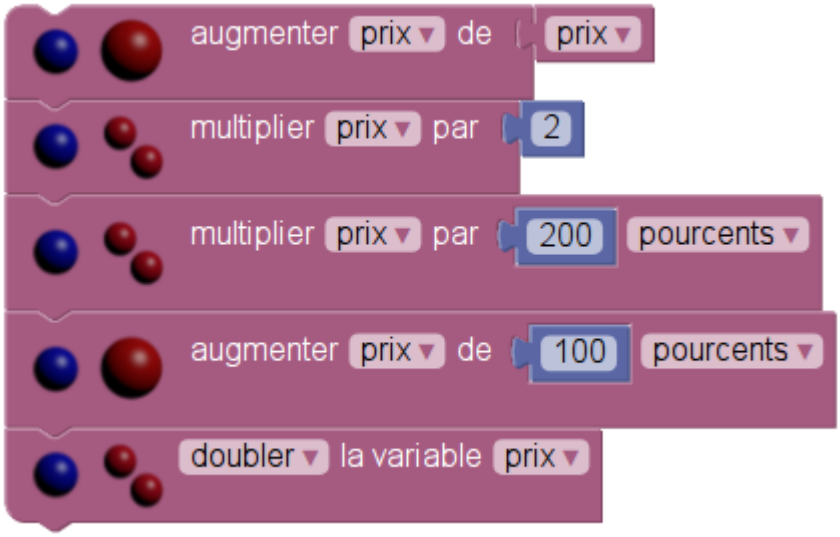
ainsi que le montre leur traduction en Python :

```
compteur = compteur + 1

compteur = compteur + 1
```

3) Doublement d'une variable

Il y a plusieurs façons différentes de doubler une variable :



On peut l'augmenter d'elle-même, la multiplier par 2, en prendre les 200 % (c'est le double) ou l'augmenter de 100 %…

En Python cela donne

```
prix = prix + prix

prix = prix * 2

prix = prix * 200 / 100

prix = prix + prix * 100 / float(100 )

prix = prix * 2
```

et en pseudo-code :

prix ← prix + prix

prix ← prix × 2

prix ← prix × 200 / 100

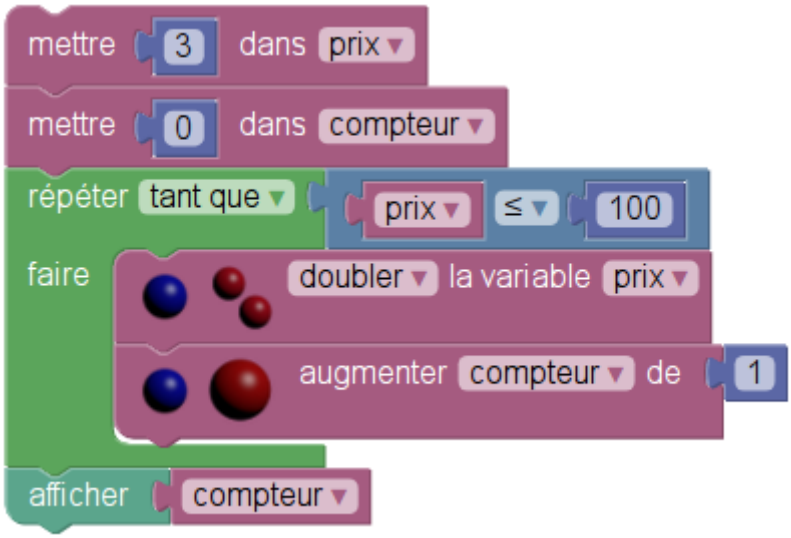
prix ← prix + prix × 100 / float(100)

prix ← prix × 2

4) Recherche de seuil

C'est l'inflation : un biscuit qui au départ ne coûtait que 3 roubles, voit son prix doubler toutes les semaines. Au bout de combien de semaines son prix dépassera-t-il 100 roubles ?

Pour résoudre ce problème, on a besoin de deux variables, le prix du biscuit, et un compteur :



En Python cela donne

```
prix = 3
compteur = 0
while prix <= 100:
    prix = prix * 2
    compteur = compteur + 1
```