Programmation en Python

S.N.T.

Lycée Carcouët

18 novembre 2019

Résumé

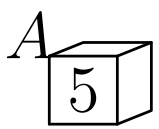
1. Variables
2. Listes
3. Si...alors...sinon
4. Boucles

Sur une calculatrice de lycée, on dispose de 26 " mémoires" : A, B, C, ... Z. On peut y stocker des nombres :

$$5 \longrightarrow A$$

(la flèche s'obtient avec $\boxed{\rightarrow}$ sur une Casio, avec $\boxed{\text{sto} \rightarrow}$ sur TI et Numworks)

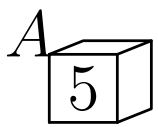
On peut imaginer une boîte nommée A qui contient 5 :

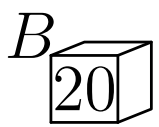


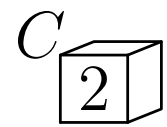
On dit que **la variable A prend la valeur 5** ou que **on affecte 5 à la variable A.**
En algorithmique, on écrira plutôt $A \longleftarrow 5$.

Sur la calculatrice, stocker 5 dans la mémoire A, 20 dans B et 2 dans C.

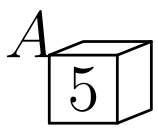
$$5 \longrightarrow A$$
$$20 \longrightarrow B$$
$$2 \longrightarrow C$$

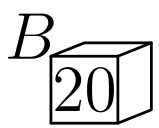


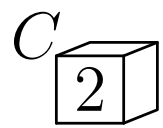




Éteindre la calculatrice, rallumer. Taper A, B, C : les mémoires ont-elles été effacées ?



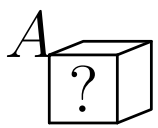


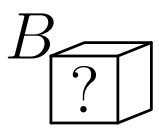


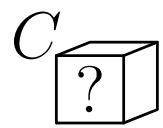
Taper

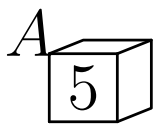
$$10 \longrightarrow A$$
$$A + B \longrightarrow C$$

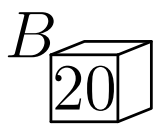
Que valent à présent les variables A, B et C ?

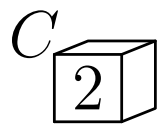








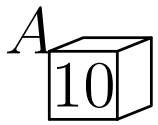


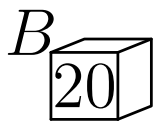


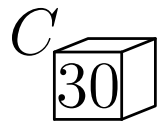
Si on tape

$$10 \longrightarrow A$$
$$A + B \longrightarrow C$$

les variables A, B et C valent alors :







(les anciennes valeurs de A et C ont été effacées, écrasées)

Dans un ordinateur, on peut stocker des nombres mais aussi des lettres, des mots, des phrases des listes de nombres, de mots ... dans la mémoire. Le nom des variables peut être plus long :

$a \longleftarrow 5$

$nombre \longleftarrow 1$

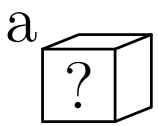
$prenom \longleftarrow$ "Gustave"

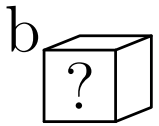
$listedePrenoms \longleftarrow$ ["Firmin","Barnabé","Lucien"]

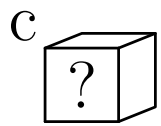
$maPhrase \longleftarrow$ "Il fait beau."

En Python, au lieu d'une flèche, on utilise le signe $=$.

$$
\begin{aligned}
a &= 5 \\
b &= 7 \\
a &= a + 4 \\
c &= (a + b)/2 \\
b &= 10 * b
\end{aligned}
$$

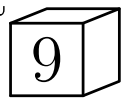


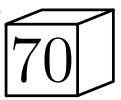


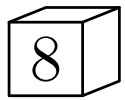


Réponse :

$$
\begin{aligned}
a &= 5 \\
b &= 7 \\
a &= a + 4 \\
c &= (a + b)/2 \\
b &= 10 * b
\end{aligned}
$$

a 

b 

c 

Résumé

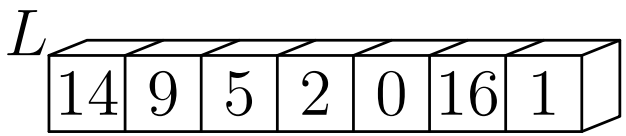
1 Variables

2 Listes

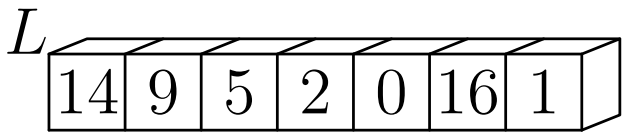
3 Si...alors...sinon

4 Boucles

Les variables vues précédemment contenaient des nombres entiers (ou des nombres décimaux (à virgule)).
Pour transformer nos images, nous aurons besoin de listes.
On peut voir une liste comme un regroupement de boîtes :

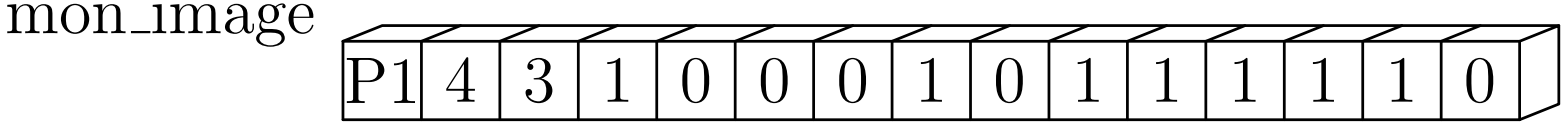


On peut stocker dans la liste L plein de valeurs d'un coup.



Pour obtenir une des valeurs, c'est simple :
$L[0]$ désigne la valeur 14.
$L[1]$ désigne la valeur 9.
$L[2]$ désigne la valeur 5.
...
$L[6]$ désigne la valeur 1.

On pourra stocker toute une image dans une seule liste !



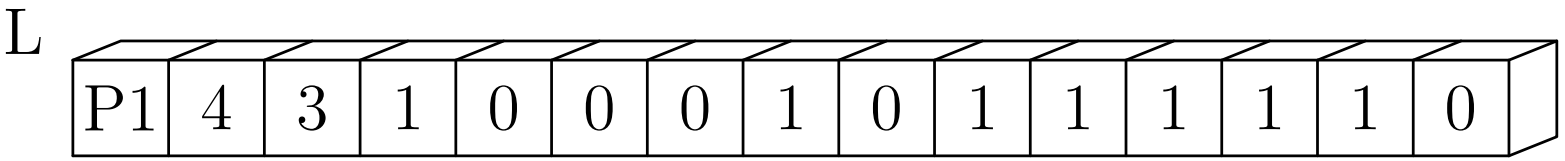
Résumé

1 Variables

2 Listes

3 Si...alors...sinon

4 Boucles



Notre but : inverser le noir et le blanc. Il nous faut changer tous les 1 en 0 et les 0 en 1.
Par exemple, pour $L[3]$ (le bit situé après le 3), on voudrait dire à l'ordinateur :

Si $L[3]$ vaut 1
alors $L[3] \longleftarrow 0$
sinon $L[3] \longleftarrow 1$

(en clair : si un bit vaut 1, tu le changes en 0. Sinon tu le changes en 1)

Si $L[3]$ vaut 1
alors $L[3] \longleftarrow 0$
sinon $L[3] \longleftarrow 1$

s'écrit en Python

```
if L[3] == 1 :
    L[3] = 0
else :
    L[3] = 1
```

On sait changer un bit de noir à blanc ou le contraire. Par contre, si notre image comporte des centaines de bits, ça va être pénible.

Résumé

1 Variables

2 Listes

3 Si...alors...sinon

4 Boucles

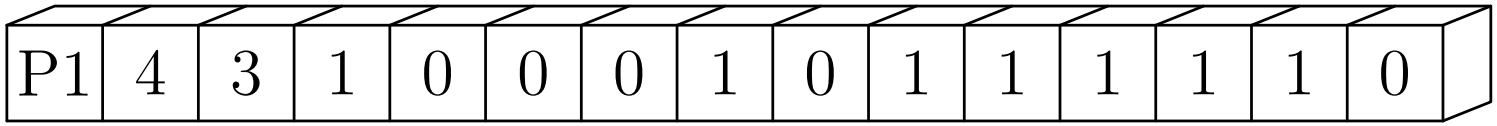
Il serait pratique d'ordonner à la machine :

"répète : "

Si $L[i]$ vaut 1 alors $L[i] \longleftarrow 0$ sinon $L[i] \longleftarrow 1$

"de $i = 0$ jusqu'à $i =$ ... (le dernier indice de notre image)".

L



Pour répéter plein de fois la même action, Python dispose des boucles **Tant que** (While) :

Si le dernier indice de notre image est 14 :

```
i = 3
while i < 15 :
    if L[i] == 1 :
        L[i] = 0
    else :
        L[i] = 1
    i = i + 1
```