L'affectation des variables - Corrigé

NUMWORKS

1 Exercice

Écrire une suite d'instructions qui échange le contenu de deux variables.

1.1 Analyse de l'énoncé

Il est important de bien comprendre la question posée. Pour cela, rien de mieux que de se baser sur un exemple.
On va imaginer deux enveloppes contenant une certaine somme d'argent : une enveloppe bleue contient un billet de 20 euros et une enveloppe rouge contient un billet de 50 euros.
On peut tout de suite définir deux variables indiquant la somme contenue dans l'enveloppe bleue et celle contenue dans l'enveloppe rouge.

```
bleue = 20
rouge = 50
```

Nous avons ainsi stocké la valeur 20 dans la variable `bleue` et la valeur 50 dans `rouge`.
Nous souhaitons maintenant échanger le contenu des deux enveloppes. Le billet de 50 euros se retrouve donc dans l'enveloppe bleue et le billet de 20 euros dans l'enveloppe rouge.
L'idée est d'écrire les instructions Python qui effectuent cette opération.

1.2 Résolution naïve qui ne fonctionne pas

Nous avons donc deux variables `bleue` et `rouge` dont nous voulons échanger le contenu.
Comme on a vu que l'affectation se fait simplement avec le signe `=`, il est très tentant d'écrire :

```
rouge = bleue
bleue = rouge
```

On teste dans la console.
Cela n'a pas donné le résultat attendu puisque le contenu de **bleue** est resté à 20…

Que s'est-il passé ?

Lorsque l'on écrit **rouge = bleue**, le contenu de **bleue**, 20, est stocké dans **rouge**. Le nouveau contenu de **rouge** est donc 20. Nous sommes donc dans l'état suivant :

```
>>> bleue
20
>>> rouge
20
```

Lorsque l'on écrit ensuite **bleue = rouge**, le contenu de **rouge**, qui est maintenant 20, est stocké dans **bleue**. Le nouveau contenu de **bleue** est donc 20. Nous sommes donc dans l'état suivant :

```
>>> bleue
20
>>> rouge
20
```

Cette méthode ne fonctionne donc pas puisque la première instruction a écrasé le contenu de la deuxième variable. L'idée est donc de stocker quelque part le contenu de la deuxième variable avant de l'écraser.

1.3 Apparition d'une troisième variable

Nous allons donc, avant toute chose, stocker le contenu de la deuxième variable dans une troisième variable, que nous allons nommer **stock**, pour éviter de perdre l'information. Un peu comme si nous avions une troisième enveloppe vide dans laquelle nous gardions au chaud le billet de 50 euros.

Instruction 1 :

```
stock = rouge
```

Maintenant que l'information est saine est sauve, nous pouvons mettre le contenu de la première variable dans la deuxième variable. Dans notre exemple nous mettons le billet de 20 euros dans l'enveloppe rouge (maintenant vide).

Instruction 2 :

```
rouge = bleue
```

Il ne reste plus qu'à mettre le billet de 50 euros dans l'enveloppe bleue, c'est-à-dire le contenu de `stock` dans la première variable.

Instruction 3 :

```
bleue = stock
```

Au terme de ces trois instructions, le contenu des deux variables a bien été échangé.

1.4 Pour terminer : une petite astuce propre à Python

En fait, le langage Python permet d'éviter de passer par une autre variable. Il est en effet possible de réaliser l'affectation de deux variables lors de la même instruction.

```
a,b = 2,3
```

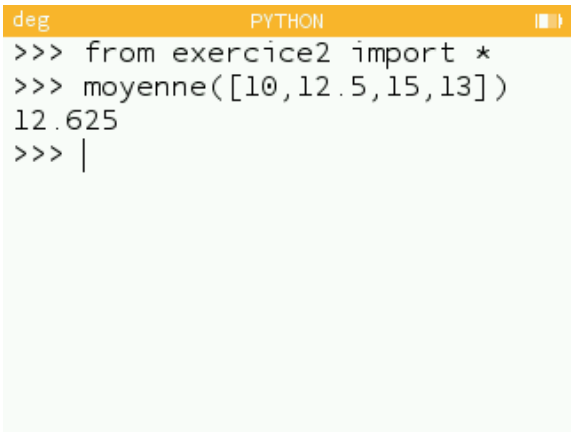
Cette instruction permet de stocker 2 dans `a` et 3 dans `b`.

Pour notre cas d'exemple il était donc possible d'écrire directement la chose suivante. Comme l'affectation est réalisée lors de la même instruction, le contenu de `rouge` n'est pas écrasé et le contenu des deux variables est échangé.

2 Un autre exercice à traiter

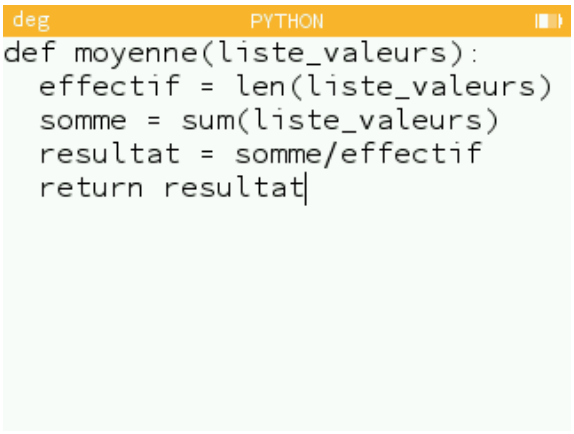
Écrire une fonction `moyenne` qui prend une liste de valeurs en argument et qui renvoie la moyenne arithmétique de ces valeurs.

2.1 Corrigé

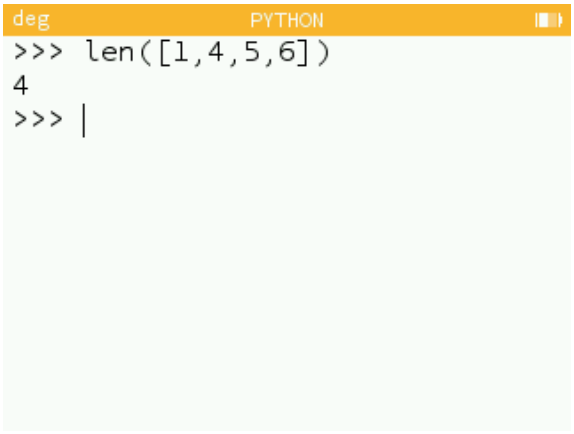


Cet exercice permet de montrer l'utilisation pratique de variables concrètes à l'intérieur d'une fonction.

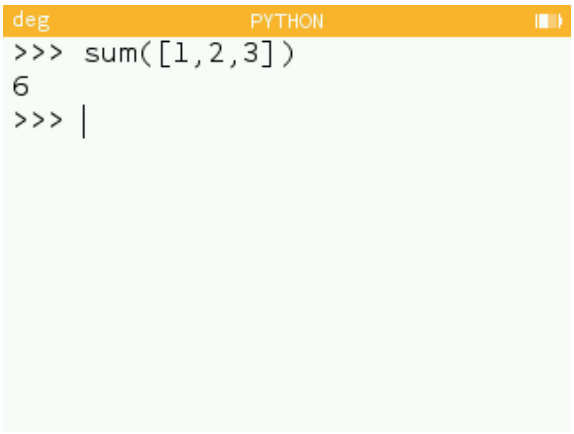
2.2 Quelques remarques sur le corrigé



Une liste de nombres est une suite de nombres entre crochets séparés par des virgules. Dans la console vous écrirez donc :



La fonction `len` permet de calculer la longueur d'une liste (nombre d'éléments).



La fonction `sum` permet de calculer la somme des éléments d'une liste, ici la somme des valeurs entrées.