Chapitre 4

Algorithmique, première partie

Dans ce chapitre nous allons découvrir les bases d'un nouveau langage de programmation qui nous permettra de construire des fonctions sur un ordinateur. Ce langage de programmation est née à la fin des années 1991 aux Pays-Bas. Il est l'oeuvre du programmeur Guido van Rossum. Ce langage est conçu pour optimiser la productivité des programmeurs en offrant des outils de haut niveau tout en proposant une syntaxe simple d'emploi.

4.1 Affectation de variable et opérations

Un programme informatique consiste en une liste (ordonnée) d'instructions utilisant des variables. Une variable peut-être vue comme une sorte de « boite »permettant à l'ordinateur de stocker momentanément des données (nombre, valeur fournie par un utilisateur, chaine de caractères, ...).

En ce qui nous concerne, nous utiliseront principalement des variables numériques. Autrement dit, les variables mises en jeu stockeront des valeurs numériques.

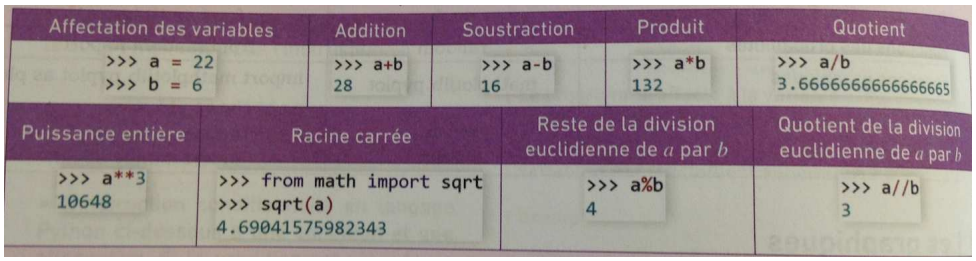
Pour stocker une valeur dans une variable, il faut utiliser un précédé *d'affectation.* En langage Python, cela revient à utiliser le symbole = avec la syntaxe *nom_de_la_variable=valeur.*

Exemple 4.1.1. L'instruction *longueur=5.7* affecte la valeur 5.7 à la variable *longeur.*

Remarque. Prenons garde aux faits suivants :

- le nom d'une variable ne doit pas contenir d'espace (ici les tirets « _ »les remplacent),
- sous Python les nombres décimaux s'écrivent avec un point et non un virgule.

Puisque nous avons affaire à des variables numériques, il est possible d'effectuer des opérations élémentaires entre elles. Voici un exemple dans lequel la variable a reçoit la valeur 22 et la valeur b reçoit la valeur 6.



Voici un exemple permettant d'appréhender ces nouvelles opérations ainsi que les résultats fourni par l'ordinateur après l'exécution de telles commandes.

Exemple 4.1.2. Le loyer mensuel d'un appartement est de 500 euros au cours de l'année 2018. Ce loyer augmente de 5% en 2019.

1. Expliquer ce que font les instructions suivantes.
   (a) $loyer = 500$

   (b) $taux = 0.05$.

2. Le locataire veux calculer le montant de l'augmentation ainsi que le nouveaux prix du loyer en 2019, et affecter ces deux valeurs à deux variables nommées *augmentation* et *nouveau-prix*. Proposer des instructions en langage Python puis donner le résultat affiché par l'ordinateur.

4.2 Les fonctions

En programmation, il est possible de créer des fonctions. Celles-ci s'apparentent à des petits programmes et sont souvent utilisées à l'intérieur d'un programme plus complexe et plus vaste. Ces fonctions sont désignées par un nom choisi par le programmeur et utilisent zéro, une ou plusieurs variables.

Présentons la syntaxe permettant de créer de tels objets.

```
>>> def nom_de_la_fonction(parmètre1,paramètre2,etc) :
        instruction(s)
        return resultat
```

Exemple 4.2.1. La fonction *somme_carres* ci dessous calcule et renvoie la somme $a^2 + b^2$ lorsque l'utilisateur donne les valeurs de variables a et b.

```
>>> def somme_carres(a,b) :
        somme=a**2+b**2
        return somme
```

Remarque. A nouveau, le nom de la fonction ne doit comporter aucun espace. Il est possible de les substituer par des tirets « _ ». Si la fonction n'utilise aucun paramètre, nous la noterons *def nom_de_la_fonction()*.

L'instruction *return* permet de renvoyer une valeur (entier, décimal ou chaine de caractère) qui peut-être utiliser de nouveau dans un autre programme ou une autre fonction. Elle interrompt le programme dès qu'elle s'est exécutée.

Notons qu'il est également possible d'utiliser une fonction écrite dans l'éditeur en utilisant directement son nom dans la console.

Exemple 4.2.2. Supposons que nous avons entré dans l'éditeur le programme déterminant la fonction *somme_carres* définies ci-dessus. Si nous tapons ensuite l'instruction *somme_carres(10,2)* dans la console de Python, celui-ci ira utilisera la fonction pour fournir le résultat 104.

Voici quelques exercices pour se familiariser avec cette nouvelle notion.

Exercice 1. Considérons la fonction

```
def exercice1(a) :
return a**2-a+1
```

1. Combien de paramètres cette fonction possède-t-elle ? Le(s)quel(s) ?

2. Que renvoit la fonction *exercice1* lorsque l'on tape les instructions suivantes ?

(a)
```
>>> exercice1(2)
```

(b)
```
>>> exercice1(2.1)
```

Exercice 2. Compléter la fonction *vitesse* ci-dessous pour qu'elle renvoit la vitesse (en km/h) lorsque l'utilisateur donne une distance en kilomètre et une durée en heure.

```
def vitesse( , ):
    return
```

Exercice 3. On propose la fonction suivante :

```
from random import*
def exercice3() :
    return randint(1,6)
```

1. Quelle est l'utilité de l'instruction de la première ligne ?

2. Combien de paramètre possède cette fonction ?

3. Parmi les propositions suivante, entourer celles qui ne peuvent pas être un résultat de cette fonction

$$2 \quad ; \quad 3,1 \quad ; \quad 7 \quad ; \quad 1,412 \quad ; \quad 6 \quad ; \quad 1$$

4. Proposer un cas concret dans lequel cette fonction aurait un intérêt.