Variables et boucles

Python comme une calculatrice

1. Lancer le programme PYZO. Taper les commandes suivantes dans la fenêtre *Shells* et noter ce qu'il se passe (le symbole ↵ désigne la touche Entrée, par la suite on ne l'indiquera plus) :

```
5+3↵
2-9↵
7+3*4↵
(7+3)*4↵
3**3↵
3**0.5↵
5/2↵        ⚠ L'opération / est la division usuelle
5//2↵       ⚠ L'opération // est la division euclidienne (division entière)
```

2. Les commandes suivantes définissent et utilisent des variables (regarder ce qui se passe dans la fenêtre Workspace au fur et à mesure) :

```
x = 10↵
x = x+1↵
largeur = 20↵
hauteur = 5*9.3↵
v = largeur*hauteur↵
print(v)↵
largeur = 10↵
print(v)↵
type(largeur)↵
type(hauteur)↵
type(v)↵
```

Un premier programme : volume d'un cylindre

3. Aller dans le menu *Fichier* puis choisir *Nouveau* (ou faire simplement *Ctrl+N*).
Immédiatement, enregistrer le fichier avec *Ctrl+S*, aller dans le menu *Documents*, cliquer sur *Nouveau Dossier*, créer ainsi un dossier *TPinfo* puis le sélectionner, créer un nouveau dossier *TP1* puis choisir comme nom de programme *volume_cylindre.py* (par exemple) ;
Taper le programme suivant dans l'éditeur :

```
from math import pi

def volume_cylindre(r,h):
    v = pi*r**2*h
    return v

print("Le volume du cylindre de rayon 2 et de hauteur 3 est")
print(volume_cylindre(2,3))
print("Le volume du cylindre de rayon 1 et de hauteur 2 est")
print(volume_cylindre(1,2))
```

Enregistrer le programme avec *Ctrl+S* puis lancer le programme avec *Ctrl+F5* et observer les résultats affichés.

Sur le même principe : aire d'un triangle

4. Écrire un programme (on pourra par exemple l'appeler *triangle.py*) qui done l'aire d'un triangle dont les côtés ont pour longueur a, b et c en utilisant la formule :

$$S = \sqrt{p(p-a)(p-b)(p-c)} \quad \text{avec} \quad p = \frac{a+b+c}{2}$$

On procèdera de la manière suivante :

- On commencera par écrire `from math import sqrt`, on disposera ainsi de la fonction `sqrt` pour calculer la racine carrée ;
- On définira une fonction `aire_triangle(a,b,c)` ;
- Dans cette fonction, on commencera par définir `p` puis `S` avec les formules ci-dessus ;
- On renverra alors `S`.

Utiliser cette fonction pour afficher l'aire d'un triangle dont les côtés ont pour longueurs respectives 3, 5 et 7. Noter le résultat obtenu.

Réponse : _____

Boucles *while*

L'instruction **while** dans un programme signifie « répéter tant que... »

5. Écrire le programme suivant (on pourra par exemple l'appeler **premier_while.py**) :

```
k = 0
while k<=5:
    k = k+1
    print(k)
```

Enregistrer le programme, exécuter et observer le résultat.

6. Modifier le programme de la manière suivante :

```
k = 0
while k<=5:
    k = k+1
print(k)
```

Enregistrer le programme, exécuter et observer la différence avec le programme précédent.

7. Modifier le programme de la manière suivante :

```
k = 0
while k<=5:
    print(k)
    k = k+1
```

Enregistrer le programme, exécuter et observer la différence avec le programme précédent.

Une somme avec une boucle *while*

8. En s'inspirant de ce qui précède, écrire un programme qui utilise une boucle *while* pour calculer la somme des nombres de 1 à 10. On utilisera comme dans ce qui précède une variable **k** qui servira de compteur mais aussi une variable **s** à laquelle on ajoutera successivement toutes les valeurs de **k**. Faire afficher le résultat et noter la réponse obtenue.

Réponse : _____

9. Modifier ce programme afin d'obtenir une fonction **somme(n)** qui calcule et renvoie la somme des nombres de 1 à n. Noter la valeur affichée avec l'instruction **print(somme(20))**.

Réponse : _____

Encore des boucles *while*

10. Tester le code suivant et comprendre à quoi correspond l'affichage obtenu.

```
n = 147
while n>0:
    print(n%10)
    n = n//10
```

11. Modifier le programme pour qu'il calcule la somme des chiffres du nombre n. À l'aide de ce programme, déterminer la somme des chiffres du nombres 9998345667.

Réponse : _____

12. En s'inspirant de ce qui précède, écrire une fonction de la forme :

```
def somme_chiffres(n):
    ...
    return s
```

qui calcule et renvoie la somme des chiffres du nombre n.

13. Vérifier que `somme_chiffres(1035)` donne bien la valeur attendue.

14. Écrire sur le même principe une fonction `somme_chiffres_cube(n)` qui calcule et renvoie la somme des cubes des chiffres de n.

15. Utiliser la fonction `somme_chiffres_cube` pour déterminer la somme des cubes des chiffres du nombre 2346.

Réponse : _____

16. À l'aide de la fonction précédente et d'une boucle *while*, déterminer le premier nombre $n \geqslant 2$ qui est égal à la somme des cubes de ses chiffres.

Réponse : _____

Corrections

Q 4.

```
from math import sqrt

def aire_triangle(a,b,c):
    p = (a+b+c)/2
    S = sqrt(p*(p-a)*(p-b)*(p-c))
    return S

print(aire_triangle(3,5,7))
```

```
6.49519052838329
```

Q 8.

```
s = 0
k = 1
while k<=10:
    s = s+k
    k = k+1
print(s)
```

```
55
```

Q 9.

```
def somme(n):
    s = 0
    k = 1
    while k<=n:
        s = s+k
        k = k+1
    return s

print(somme(20))
```

```
210
```

Q 10. Les chiffres 7, 4 et 1 sont affichés successivement. Il s'agit des chiffres du nombre **n** de départ en commençant par le chiffre des unités.

Q 11.

```
n = 9998345667
s = 0
while n>0:
    s = s+n%10
    n = n//10
print(s)
```

```
66
```

Q 12.

```
def somme_chiffres(n):
    s = 0
    while n>0:
        s = s+n%10
        n = n//10
    return s
```

Q 13.

```
print(somme_chiffres(1035))
```

```
9
```

Q 14.

```
def somme_chiffres_cube(n):
    s = 0
    while n>0:
        s = s+(n%10)**3
        n = n//10
    return s
```

Q 15.

```
print(somme_chiffres_cube(2346))
```

```
315
```

Q 16.

```
n = 2
while n!=somme_chiffres_cube(n):
    n = n+1
print(n)
```

```
153
```