TD 5 (programmation) - Fonctions et boucles.

Exercice 1 (`break` & `continue`)

Qu'affiche le programme ci-dessous ?

```
if __name__ == "__main__":
    i = 7
    while i < 1000:
        i = i + 1
        print(i)
        if i < 10:
            continue
        if i % 5 == 0:
            i = 2 * i
        if i % 17 == 0:
            break
```

Exercice 2

Écrivez une fonction qui renvoie la somme des chiffres d'un naturel qu'on lui passe en paramètre. Par exemple, `somme(9650165489)` devra renvoyer 9+6+5+0+1+6+5+4+8+9 = 53. Vous ne pouvez pas utiliser le transtypage.

Indices

À quoi correspond le reste d'une division par 10 ?

Exercice 3 (Moyenne)

(a) Écrivez un programme qui permet à l'utilisateur de saisir autant de nombres entiers qu'il veut et qui affiche la moyenne des nombres rentrés. Quand l'utilisateur a terminé, il écrit `"stop"` au lieu d'un nombre, et c'est à ce moment-là qu'on lui affiche la moyenne.

(b) Modifiez le programme pour qu'il redemande les nombres tant qu'ils ne sont pas compris entre 0 et 20.

Exercice 4 (Racine cubique)

Pour cet exercice, vous devez vous passer du transtypage.

(a) Écrivez une fonction `racine_cubique` qui prend un nombre naturel en argument et qui renvoie la partie entière de la racine cubique de ce nombre, c'est-à-dire le plus grand naturel dont le cube est inférieur ou égal à ce nombre. Par exemple, `racine_cubique(8)` renverra 2, et `racine_cubique(29)` renverra 3 car $3^3 = 27 \leq 29$ mais $4^3 = 64 > 29$.

(b) Écrivez une fonction `est_cube(n)`, où `n` est un naturel, qui vérifie si `n` est un cube. Par exemple, `est_cube(27)` renvoie `True` et `est_cube(4)` renvoie `False`.

(c) Adaptez ces deux fonctions pour qu'elles fonctionnent aussi pour les entiers négatifs.

Exercice 5

Écrivez un programme qui demande à l'utilisateur de deviner un nombre entre 1 et 100 généré aléatoirement à l'aide de la fonction `randint` du module `random`. Le nombre d'essais doit être limité à un certain `n` fixé au préalable. L'affichage devra être le suivant :

```
$ python3 jeu.py
Tentez de deviner un nombre entre 1 et 100 (essais restants: 10): 50
Raté, le nombre à trouver est plus grand
Tentez de deviner un nombre entre 1 et 100 (essais restants: 9): 75
Raté, le nombre à trouver est plus grand
...
```

Si l'utilisateur a échoué après le nombre d'essais imparti, on affichera à la fin du programme : `"Vous avez perdu, le nombre à trouver était ..."`.

Exercice 6

(a) Écrivez un programme qui fait jouer l'utilisateur à "Pierre / papier / ciseaux" contre l'ordinateur. L'utilisateur rentre son choix sous la forme d'une chaîne de caractères (ou `"stop"` pour arrêter). À la fin de l'exécution du programme, celui-ci doit afficher les scores (par exemple : "5 victoires, 3 défaites" pour l'utilisateur). L'ordinateur jouera au hasard à chaque partie.

(b) Une fois le programme fonctionnel, modifiez-le pour forcer l'utilisateur à rentrer une chaîne interprétable, c'est-à-dire soit un des trois choix valides, soit la chaîne `"stop"` ; si l'utilisateur rentre autre chose, il faut lui redemander son choix tant qu'il n'est pas valide.

(c) Si vous avez résolu l'exercice à l'aide de nombreux `if`, trouvez une solution plus concise. Vous pouvez en cas de besoin utiliser la méthode `index` des listes : `lst.index(valeur)` renvoie la position de la première occurrence de `valeur` dans la liste `lst`.

Indices

L'ordre (pierre, ciseaux, papier) est un ordre cyclique : pierre bat ciseaux, ciseaux bat papier, et papier bat pierre.