TP 6 : Domptez la boucle while !

Bienvenue, apprenti Pythoniste ! La boucle while est une créature puissante, mais capricieuse. Contrairement à la boucle for, elle ne sait pas combien de tours elle doit faire à l'avance. Elle continue jusqu'à ce qu'on lui dise d'arrêter. Aujourd'hui, tu vas apprendre à la maîtriser.

1. La boucle infinie (mais pas vraiment)

La boucle `while` exécute son code tant que la condition est vraie. Mais attention, si tu ne lui dis jamais de s'arrêter, elle tournera… à l'infini (et ton ordinateur pourrait chauffer !).

Exemple :

```
i = 0
while i < 5:
    print("Tour n°", i)
    i = i + 1  # N'oublie pas d'incrémenter, sinon c'est la boucle infinie !
```

Analyse du code :

`i = 0` : On crée une variable i à 0, qui nous servira de compteur.

`while i < 5:`: tant que i < 5, python fera les lignes suivantes. Ici, Python fait un test pour voir si i est inférieur à 5.

`print("Tour n°", i)`: Afficher : Tour n° i

`i = i + 1` : On augmente de 1 la valeur de i puis on boucle. Si on oublie cette ligne, i vaudra toujours 0 et on se retrouve dans une boucle infinie…

Remarque :

Pour `i = i + 1`, on peut aussi écrire `i +=1`.

Exercice 1 : Les 10 premiers nombres

Crée une boucle `while` qui affiche les nombres de 0 à 9.

2. Devine le nombre magique !

Les jeux interactifs, c'est toujours sympa. Ici, nous allons utiliser une boucle while pour demander à l'utilisateur de deviner un nombre secret. La boucle continue tant que le nombre n'est pas trouvé !

Exemple :

```
nombre_secret = 7
reponse = 0

while reponse != nombre_secret:
    reponse = int(input("Devine le nombre secret (entre 1 et 10) : "))
    if reponse == nombre_secret:
        print("Bravo ! Tu as trouvé.")
    else:
        print("Essaie encore.")
```

3. Tant que c'est faux... c'est faux !

Dans cette boucle, on va vérifier si l'utilisateur entre une réponse correcte. Tant qu'il fait une erreur, on lui redemande. Pratique pour valider des entrées utilisateur !

Exemple :

```
reponse = ""
while reponse != "python":
    reponse = input("Quelle est la meilleure langue de programmation ? ")
    if reponse != "python":
        print("Non, essaie encore...")
```

Exercice 2 : Le mot magique

Crée un programme qui demande à l'utilisateur un mot de passe (par exemple, "tortue"). Tant que le mot de passe n'est pas correct, la boucle continue et redemande avec un message d'erreur.

4. Jeu des "Yes or No"

Créons un petit jeu où l'utilisateur doit répondre par "oui" ou "non". La boucle `while` continue tant qu'il ne donne pas une réponse valide.

Exemple :

```
reponse = ""
while reponse not in ["oui", "non"]:
    reponse = input("Veux-tu continuer ? (oui/non) : ")

if reponse == "oui":
    print("Super, on continue !")
else:
    print("D'accord, à la prochaine !")
```

Exercice 3 : Le choix

Crée un programme où l'utilisateur doit répondre par "oui" ou "non" à une question que vous choisirez. Tant qu'il n'a pas donné de réponse valide, le programme redemande.

5. Oui, bon, hein, on va faire un peu de Maths avec les suites ou pas...

La boucle `while` est très pratique pour voir à partir de quel moment une suite dépasse une valeur seuil. C'est d'ailleurs un script qui tombe régulièrement au Bac.

Il faut bien penser à initialiser la suite (avec son premier terme) et aussi un compteur qui nous servira pour l'indice des termes de la suite.

Exercice 4 (si tu connais les suites...):

Soit la suite (u_n) définie par : $u_{n+1} = 3u_n + 2$ et son premier terme $u_0 = 5$.

À partir de quel n a-t-on $u_n > 1500$.

Exercice 4bis (si tu ne connais pas les suites...):

Tous les mois, tu mets de côtés 35 PythonCoins dans une `tirelire`. Écris un programme qui, lorsque tu atteindras la somme de 2500 PythonCoins, affichera : « tu es enfin riche !!! ».

Félicitations !

Si tu es arrivé(e) jusqu'ici, tu maîtrises désormais l'art des boucles `while`. Tu es prêt(e) à dompter Python et à écrire des programmes qui réagissent intelligemment à l'utilisateur ! 🎉