La boucle while

La boucle conditionnelle **`while`** (*tant que* en français) **vérifie la condition de boucle au début de chaque itération**.

- Si la condition est vraie, elle effectue une nouvelle itération.
- Si la condition est fausse, la boucle s'arrête.

Attention, si la condition est mal choisie, on risque de ne jamais entrer dans la boucle, ou pire de ne jamais en sortir !

La boucle `while` permet ainsi de créer des répétitions, même si l'on ne connaît pas à l'avance le nombre de répétitions. On parle de **boucle non bornée**.

L'écriture d'une boucle non bornée nécessite trois étapes :

1. Initialisation de la variable de contrôle.
2. Vérification de la condition.
3. Mise à jour de la variable de contrôle.

Dans l'exemple ci-dessous, nous écrivons l'équivalent d'une boucle for avec une boucle `while`. Dans une boucle for toutes les affectations de la variable i sont gérées par la fonction `range()`. Avec le `while` c'est à nous de nous en occuper.
(En pratique, il est fortement recommandé d'utiliser une boucle `for` quand cela est possible, elles sont sujettes à moins d'erreurs de programmation que les boucles `while`.)

```
i = 1                  # initialisation
while i < 7:           # condition d'arrêt
    print(i)
    i = i + 2          # Mise à jour de la variable de contrôle
# affiche successivement 1 3 5
```

Dans l'exemple ci-dessous, la boucle `while` continue tant que le contenu de la variable `mdp` est différent de `"P@ssw0rd"`. Quand `mdp` est contient `"P@ssw0rd"`, la boucle s'arrête.

```
mdp = input("Password ? ")            # initialisation
while mdp != "P@ssw0rd":              # condition d'arrêt
    print("Bad password! Please retry.")
    mdp = input("Password ? ")        # Mise à jour de la variable de contrôle
print("Welcome !")
```

Les codes ci-dessous sont des exemples de **boucles infinies**.

- La variable `i` commence à 0 et est incrémentée à chaque itération. La condition de boucle n'est donc jamais fausse.

```
i = 0
while i >= 0: # !!!! BOUCLE INFINIE !!!!
    print(i)
    i = i + 1
```

- La variable `i` commence à 10 et mais ne change pas à chaque itération. La condition de boucle n'est donc jamais fausse.

```
i = 10
while i >= 0: # !!!! BOUCLE INFINIE !!!!
    print(i)
    i = i
```

Exercice 1 : Predict, run, investigate.

Sans exécuter, qu'affichent les codes suivants ? Vérifier en exécutant le code.

`# Code 1`
`i = 10`
`while i > 2:`
`    print('cool')`
`    i = i / 2`
	`# Code 2`
	`i = 1`
	`compteur = 0`
	`while i < 64:`
	`    print(compteur)`
	`    compteur = compteur + 1`
	`    i = i + i`

`# Code 3`
`i = 1`
`while i**2 <= 100:`
`    print(i)`
`    i = i + 1`
	`# Code 4`
	`i = 0`
	`j = 0`
	`while i <= 4:`
	`    while j < 3:`
	`        j = j + 1`
	`        print('j=', j)`
	`    print('i=', i)`
	`    i = i + 1`

Exercice 2 : Modify.

1) En modifiant le code 3 de l'exercice 1, écrire un programme qui trouve le plus petit entier n tel que $n^3 > 10000$.

2) On dispose d'une feuille de papier d'épaisseur 0,1 mm. On cherche à savoir combien de fois nous devons la plier au minimum pour que l'épaisseur dépasse la hauteur de la tour Eiffel (324 mètres) ? En modifiant le code 2 de l'exercice 1, écrire un programme pour répondre à cette question.

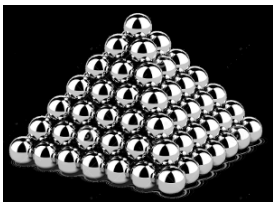
Exercice 3 : Make.

1) Écrivez un programme qui fait jouer l'utilisateur au ni oui, ni non : il rentre un texte jusqu'à saisir "oui" ou "non", ce qui déclenche la fin du jeu.

2) a) Écrivez un programme qui fait saisir un nombre à l'utilisateur jusqu'à ce que ce nombre soit inférieur ou égal à 100.
b) Ensuite, améliorez votre programme pour que le nombre saisi soit compris entre 50 et 100 inclus.

3) Aujourd'hui un appartement vaut 100 000€. Sa valeur augmente de 1% chaque année.
a) Écrire un programme en Python pour connaître sa valeur au bout de 10ans.
b) Écrire un programme en Python pour savoir au bout de combien d'années sa valeur aura doublé.

4) Inès veut construire une pyramide à base carrée comme sur la photo. La pyramide sur la photo a 7 étages.



Inès a 1000 billes. Combien d'étages au maximum aura sa pyramide ? Écrire un programme en Python pour répondre au problème.

5) Nous allons écrire un programme qui choisi aléatoirement un nombre entier et tente de le faire deviner au joueur.

- **a)** Générer aléatoirement un entier entre 1 et 100 tous deux inclus. Rechercher en ligne ou dans les précédents TPs.
- **b)** Pour la 1ère version du jeu, le programme continue tant que le nombre n'est pas deviné par l'utilisateur. Afficher un message quand le joueur trouve le nombre.
- **c)** Pour la 2nde version du jeu, quand le joueur n'a pas la bonne réponse, lui indiquer si le nombre recherché est plus grand ou plus petit que sa dernière réponse.
- **d)** Pour la 3nde version du jeu, limiter le nombre d'essais du joueur à 7. Au bout de 7 essaies infructueux, le joueur perd.
- **e)** Pour la 4ème version du jeu, demander à l'utilisateur un nombre n, générer le nombre aléatoire entre 1 et n inclus et limiter le nombre d'essais du joueur à `math.ceil(math.log2(n))`.

6) Jeu de Nim - le duel des bâtonnets.

Partant d'un tas de 20 bâtonnets, deux joueurs s'affrontent chacun leur tour en enlevant 1, 2 ou 3 bâtonnets. Le vainqueur est celui qui prend le dernier bâtonnet. Voici un exemple de partie :

Itération ...	1	2	3	4	5	6	7	8	9	10	Fin
C'est à ... de jouer.	J1	J2	J1	J2	J1	J2	J1	J2	J1	J2	J2 Gagne !
Il reste ... bâtonnets.	20	19	16	13	12	10	8	5	4	3	0

Voici les grandes lignes du programmes :

- Le nombre de bâtonnets est initialisé à 20.
- Le premier joueur à jouer est le joueur 1.
- Tant que le nombre de bâtonnets n'est pas égal à 0 :
  - Un nombre doit être demandé au joueur. Tant que ce nombre n'est pas égal à 1, 2 ou 3, on lui redemande. Une fois validé, on diminue le nombre de bâtonnets.
  - En fin de boucle, si la partie n'est pas finie, le joueur doit changer.
- La boucle finie, le joueur ayant pris le dernier bâtonnet gagne.

- **a)** Coder ce jeu.
- **b)** Pour la 2nde version du jeu, en début de partie, offrir la possibilité de jouer à 1 joueur contre une intelligence artificielle. Dans ce cas de figure, les coups du joueurs 2 sont choisis aléatoirement.