6.1.3 La boucle `while`

La boucle `for` s'utilise lorsqu'on connaît à l'avance le nombre de répétitions à effectuer: soit un nombre entier, soit un ensemble de valeurs contenus dans un *iterable* dont le nombre de valeurs est fixe. On parle de boucle **bornée**.

Mais il arrive fréquemment qu'on doive répéter des instructions un *certain* nombre de fois, qui n'est pas connu à l'avance. On a donc besoin d'une boucle **non bornée**, `while` en Python, qui s'éxécutera tant qu'une condition est réalisée et qui stoppera dès que cette condition ne le sera plus.

Si jamais cela arrive...

{: .center}

1. Un premier exemple

> Combien de fois doit-on plier une feuille de papier pour que son épaisseur dépasse la hauteur de la Tour Eiffel (324 m)?

Sous réserve que cela soit techniquement possible, on va donc partir d'une feuille de papier classique, d'une épaisseur de 0,1 mm, puis on va plier, plier, plier... sachant que l'épaisseur va doubler à chaque pliage.

Il est donc nécessaire d'utiliser une boucle pour répéter ces pliages (c'est-à-dire les calculs successifs de l'épaisseur de papier). Répeter, d'accord, mais combien de fois? Justement, on ne le sait pas, puisque c'est la question !!!

En revanche, on sait qu'on doit s'arrêter de plier dès que l'épaisseur de papier sera supérieure à 324 m. Autrement dit, on doit repéter **tant que l'épaisseur est inférieure à 324 m**.

On obtient donc le code:

"'python linenums='1' epaisseur = 0.0001 nombre_pliages = 0

while epaisseur < 324: epaisseur = 2 * epaisseur nombre_pliages += 1

print("Il faut", nombre_pliages, "pliages.") "'

2. La syntaxe

!!! abstract "La boucle `while`"

```
**Syntaxe générale:**
```python linenums='1'
while expression:
    *instructions à répéter*
```

Remarques:
```

- `expression` doit renvoyer une valeur **booléenne** : une égalité, comparaison, appartenar
- `while` repète le bloc d'instructions à répéter tant que cette expression est égale à `Tru
- il faut terminer la ligne commençant par `while` par `:` ;
- le bloc d'instructions à répéter (le corps de la boucle) doit être indenté.
<!-- - le bloc d'instructions à répéter **doit contenir une modification de l'expression boo

3. Les dangers de la boucle `while`

3.1 Ne JAMAIS ENTRER dans la boucle

!!! danger “No Entrance” Le programme suivant ne va pas faire grand chose… à part afficher `fini!`. Voyez-vous pourquoi? `python linenums='1'  toto = 0  while toto >= 1:  toto = toto + 1  print("ce texte ne s'écrira jamais")  print("fini!")`

3.2 Ne JAMAIS SORTIR de la boucle

Je vous rassure tout de suite, aucun programmeur professionnel n’est passé à côté de ce piège…

!!! danger “No Exit” Que va faire le programme suivant?

```python linenums='1'
toto = 1
while toto < 10:
    toto = 2
    print("ok")
print("fini!")
```

!!! warning “Éviter une boucle infinie” Il faut **toujours** s’assurer que le corps de la boucle contienne une instruction qui fera en sorte qu’à un moment donné l’expression booléenne prendra la valeur `False`.

`![](../images/while_meme.jpg){.center}`

4. Exercices

{{ initexo(0) }}

!!! example “{{ exercice() }} : pydéfi” === “Énoncé” SW IV : On passe en vitesse lumière : https://pydefis.callicode.fr/defis/VitesseLumiere/txt{:target="_blank"}

```
    Comment compter le nombre de passages dans la boucle?

=== "Correction"
    {{ correction(False,
    "
    ```python linenums='1'
```
```

```
    x = 997
    y = 312
    z = 663
    compteur = 0
    while 10*x > y:
        compteur += 1
        x = (y * z) % 10000
        y = (3 * z) % 10000
        z = (7 * z) % 10000
    print(x, y, z)
    print(compteur)
    # on obtient 501 9461 5409
    # 23 passages
    ```
    "
) }}
```

!!! example "{{ exercice() }}" === "Énoncé" Trouver le plus petit nombre entier $n$ tel que $2^n$ soit supérieur à 1 milliard. === "Correction" {{ correction(False, " python linenums='1' n = 1 while 2**n < 10**9: n = n + 1 print('trop petit') print('trouvé : ',n) " ) }}

!!! example "{{ exercice() }}" === "Énoncé" Demander à l'utilisateur de taper la lettre S (puis sur la touche Entrée). Recommencer tant qu'il n'a pas obéi.

```
    On utilisera la fonction `input` pour récupérer une saisie clavier (voir [ici](https://c

=== "Correction"
    {{ correction(False,
    "
    ```python linenums='1'
            touche = ''  #chaine de caractère vide
            while touche != 'S':
                touche = input('appuyez sur S s il vous plaît ')

            print('ouf, merci !')
            ```
    "
) }}
```

!!! example "{{ exercice() }} : pydéfi" === "Énoncé" L'algorithme du
```

professeur Guique : https://pydefis.callicode.fr/defis/Algorithme/
txt{:target="_blank“} ===”Correction" {{ correction(False, " python
linenums='1' a = 1 b = 4 c = 3 k
= 1 n = 0 while k < 1000-n: a = b
b = c + a c = -4*c - 3*a - b n = a + b
k += 1 print(a, b, c) ") }}

!!! example “{{ exercice() }}” === “Énoncé” On considère le code ci-dessous:
python linenums='1' for k in range(5): print("Scooby-doo")

```
    Ré-écrire ce code en utilisant une boucle `while`.

=== "Correction"
    {{ correction(False,
    "
    ```python linenums='1'
    compteur = 0
    while compteur < 5:
        compteur += 1
        print('Scooby-doo')
    ```
    "
) }}

{.center width=40%}
```