# 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...


## 1. 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:

Script Python

```python
epaisseur = 0.0001
nombre_pliages = 0

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

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

# 2. 2. La syntaxe

### La boucle `while`

**Syntaxe générale:**

Script Python

```python
while expression:
    *instructions à répéter*
```

**Remarques:**

- `expression` doit renvoyer une valeur **booléenne** : une égalité, comparaison, appartenance, etc. ;
- `while` repète le bloc d'instructions à répéter tant que cette expression est égale à `True` ;
- il faut terminer la ligne commençant par `while` par `:` ;
- le bloc d'instructions à répéter (le corps de la boucle) doit être indenté.

# 3. 3. Les dangers de la boucle `while`

## 3.1. 3.1 Ne JAMAIS ENTRER dans la boucle

### No Entrance

Le programme suivant ne va pas faire grand chose... à part afficher `fini!`. Voyez-vous pourquoi?

Script Python

```python
toto = 0
while toto >= 1:
    toto = toto + 1
    print("ce texte ne s'écrira jamais")
print("fini!")
```

## 3.2. 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...

### No Exit

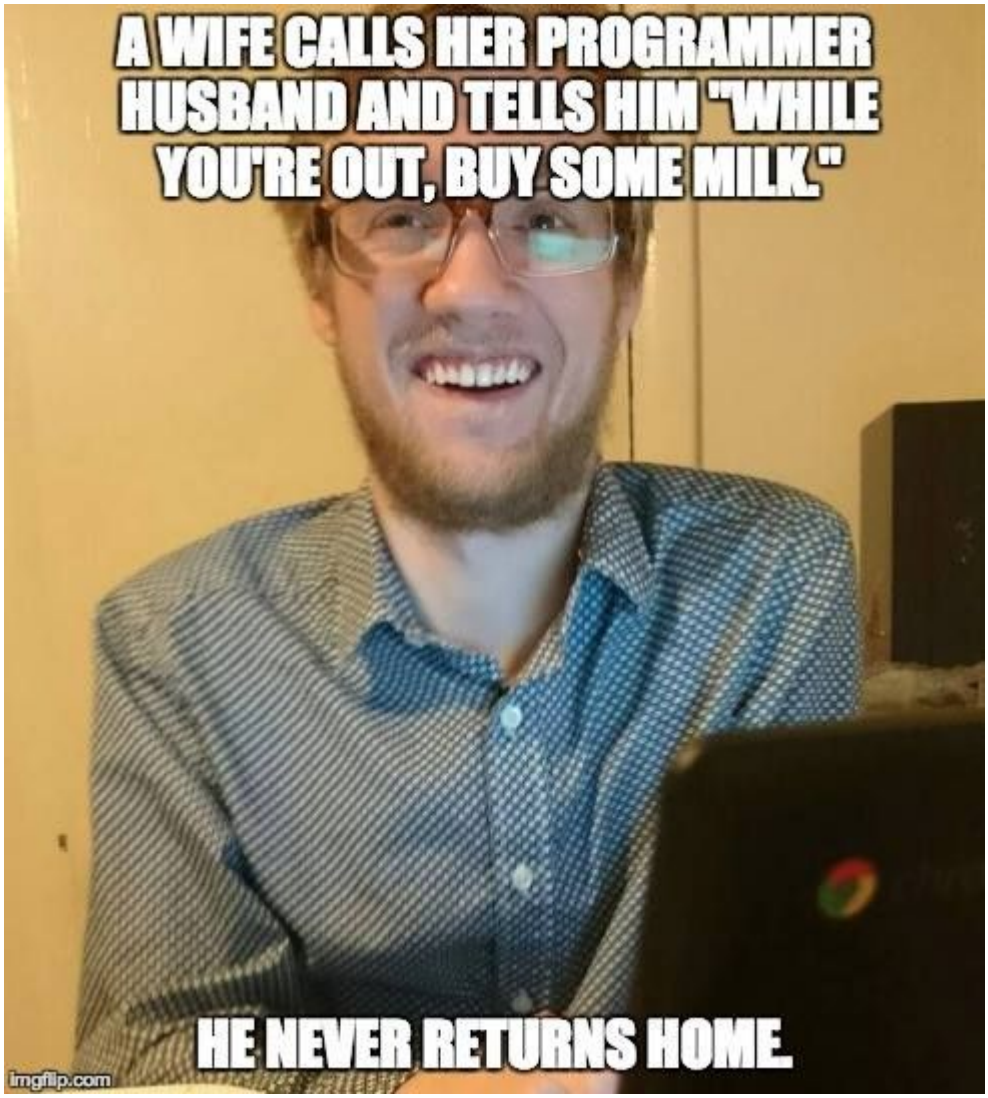
Que va faire le programme suivant?

Script Python

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

**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`.




## 4. 4. Exercices

## Exercice 1 : pydéfi

### Énoncé

SW IV : On passe en vitesse lumière : https://pydefis.callicode.fr/defis/VitesseLumiere/txt

Comment compter le nombre de passages dans la boucle?

### Correction

## Exercice 2

### Énoncé

Trouver le plus petit nombre entier $n$ tel que $2^n$ soit supérieur à 1 milliard.

### Correction

## Exercice 3

### 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).

### Correction

## Exercice 4 : pydéfi

### Énoncé

L'algorithme du professeur Guique : https://pydefis.callicode.fr/defis/Algorithme/txt

### Correction

## Exercice 5

### Énoncé

On considère le code ci-dessous:

Script Python

```
for k in range(5):
    print("Scooby-doo")
```

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

### Correction