

TP 3 Python – Boucles conditionnelles (while)

1. Boucles while en Python

Pour écrire des instructions qui se répètent tant qu'une certaine condition est vérifiée, Python dispose du mot-clé `while`.

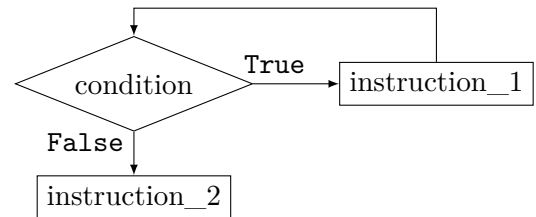
En Python :

```
1 while condition:
2     instruction_1
3 instruction_2
```

En pseudo-code :

Tant que condition :
 Faire instruction_1
Fin Tant que
instruction_2

Diagramme de flux :



Remarque. L'instruction 1 est exécutée en boucle tant que `condition` vaut `True`. Chaque exécution du bloc dans la boucle est appelée une **itération**.

Exemple

Décrire l'état du système (valeur des variables et de la condition)
à chaque itération de la boucle ci-contre :

```
1 x = 1
2 while x < 6:
3     x = x + 2
4 x = x / 2
```

⚠ **Boucle infinie** : Si la condition n'est jamais fausse, la boucle se répète indéfiniment ! Par exemple :

```
1 x = 1
2 while x < 2:
3     x = x - 1
```

2. Application À la division euclidienne d'entiers naturels

On suppose que `a` et `b` sont des variables déjà affectées avec des valeurs entières positives ($a, b \in \mathbb{N}$), et $b \neq 0$. On veut écrire un programme qui permette de connaître le quotient et le reste de la division euclidienne de `a` par `b`.

Une méthode naïve pour ce faire consiste à retirer `b` à `a` jusqu'à obtenir un entier assez petit, le reste. Le quotient est alors le nombre de fois qu'on a dû retirer `b` à `a`.

```
1 quotient=0
2 reste = a
3 while reste >= b:
4     reste = reste - b
5     quotient = quotient + 1
6 reste = a
```

3. Exercices

Exercice 1. On considère les deux programmes suivants :

```
1 a = 7.5
2 if a > 3:
3     a = a - 1
```

```
1 a = 7.5
2 while a > 3:
3     a = a - 1
```

1. Tracer le diagramme de flux correspondant à chacun des scripts. Que vaut `a` à la fin du programme de gauche ?
2. Simuler à la main l'exécution du programme de droite, en détaillant chaque itération (on pourra par exemple faire un tableau).

3. Donner une valeur initiale de **a** pour laquelle les deux programmes ont le même état final.

Exercice 2. Déterminer à l'aide d'un programme Python le plus petit entier n tel que $1 + \frac{1}{2} + \frac{1}{3} + \dots + \frac{1}{n}$ soit strictement supérieur à 10

Exercice 3. On peut montrer que quand l'entier n tend vers $+\infty$, la somme $S_n = 1 - \frac{1}{2} + \frac{1}{3} - \frac{1}{4} + \dots + \frac{(-1)^n}{n}$ tend vers $\ln(2)$. Utiliser un programme Python pour déterminer la plus petite valeur de n pour laquelle S_n est proche de $\ln(2)$ à moins de 10^{-5} .

Exercice 4. Des chercheurs cultivent un champignon en laboratoire. Au départ, ils disposent de seulement 10 g de ce champignon. Chaque jour, la masse de champignon augmente de 20%, puis les chercheurs prélèvent 1 g pour analyse.

Combien de jours faudra-t-il pour que la masse de champignon dépasse 500 g ?

Problème : Conjecture de Syracuse

Une suite de Syracuse est une suite d'entiers naturels définie par un premier terme $u_0 \in \mathbb{N}^*$ et la relation de récurrence :

$$\text{pour tout } n \in \mathbb{N}, \quad u_{n+1} = \begin{cases} \frac{u_n}{2} & \text{si } u_n \text{ est pair} \\ 3u_n + 1 & \text{si } u_n \text{ est impair} \end{cases}$$

1. Calculer à la main les 15 premiers termes de la suite de Syracuse si $u_0 = 12$.
Que remarquez-vous sur le comportement de la suite après avoir atteint la valeur 1 ?

La conjecture de Syracuse, énoncée en 1928 et ***pas encore démontrée***, est la suivante :

Quel que soit le nombre entier u_0 choisi, la suite atteint 1 : il existe un rang $n \in \mathbb{N}$ tel que $u_n = 1$.

- On appelle **temps de vol** de la suite le rang du premier terme de la suite qui vaut 1.
 - On appelle **altitude de vol** de la suite la valeur maximale prise par les termes de la suite.
2. Donner le temps de vol et l'altitude de vol de la suite de Syracuse commençant en $u_0 = 12$.
 3. Écrire une fonction Python `Suivant_Syracuse(u)`, qui prend en entrée un entier naturel u , qui correspond à un terme de la suite de Syracuse, et qui renvoie le terme qui le suit.
 4. Testez votre fonction avec plusieurs valeurs de u pertinentes.
 5. Écrire une fonction `temps_de_vol(u0)` qui prend en entrée un premier terme $u_0 \in \mathbb{N}^*$ et renvoie le temps de vol de la suite de Syracuse de premier terme u_0 .
 6. Testez votre fonction avec $u_0 = 12$. La boucle conditionnelle peut-elle être infinie ?
 7. Écrire un programme Python qui détermine le premier entier $u_0 \in \mathbb{N}^*$ pour lequel le temps de vol est supérieur ou égal à 200.
 8. Écrire une fonction `altitude_de_vol(u0)` qui prend en entrée un premier terme $u_0 \in \mathbb{N}^*$ et renvoie l'altitude de vol de la suite de Syracuse de premier terme u_0 .