Chapitre 3 : Boucles simples

Boucle `while`

Syntaxe

La boucle `while` (*while* signifiant *tant que* en anglais) permet d'exécuter plusieurs fois des instructions *similaires* sous 'condition'. Elle s'écrit de la manière suivante :

```
Instruction 0
while condition :
    instruction 1
    instruction 2
    ...
    instruction n
Instruction n+1
```

Corps de la boucle

- Le corps de la boucle est l'ensemble des instructions exécutées (`instruction 1` à `instruction n` ci-dessus).
  - Ces instructions doivent être indentées (**Très important**).
  - Elles sont exécutées tant que `condition` est vraie.

Condition de maitien dans la boucle

- `condition` doit devenir fausse à un moment donné (**sinon la boucle est infinie**)
- il doit donc y avoir des instructions dans le corps qui changent la valeur de `condition`
- Les variables intervenant dans `condition` doivent être initialisées avant l'instruction `while condition :`

Exemple de boucle infinie

```
In [ ]: i=1
        while i>0 :
            print(i)
            i=i+1
        print('message jamais affiche')
```

Pour répéter une instruction un certain nombre de fois il faut un compteur

Exemple d'une boucle affichant les nombres de 1 à 10

```
In [ ]: compteur = 1
        while compteur <= 10 :
            print(i)
            i = i + 1
```

Quelques remarques sur l'exemple (mais qui s'appliquent dans le cas général) : * l'initialisation de la variable (`compteur`) intervenant dans la condition est faite AVANT la boucle ; * l'incrémentation de la variable `compteur` dans le corps de la boucle permet `condition` de devenir fausse et donc la sortie de la boucle (ce qui n'est pas vrai dans l'exemple d'une boucle infini ci-dessus) ; * bien estimer la condition d'arrêt et les valeurs des variables en sortie : dans l'exemple, la condition devient fausse lorsque `compteur` est supérieur à `10` et en sortie `compteur` vaut 11 et non 10 !

Tests unitaires

Tests unitaires

Les tests unitaires permettent de vérfier la validité du programme développé, *"Est-ce que le programme fournit a le comportement attendu par le cahier des charges ?"*. Pour le vérifier, il faut prévoir **plusieurs tests significatifs** pour tester le programme développé.

La fonction `assert`

- syntaxe : `assert condition, message`
- la fonction `assert` permet de verifier la validité d'une condition, et réagit différemment selon que cette dernière est vraie ou fausse :
- **si la condidition est vraie** : l'appel de la fonction `assert` est totalement transparente ; le programme continue son déroulement ;
- **si la condition est fausse** : l'appel de la fonction`assert` engendre l'affichage d'un message d'erreur accompagné du message `message` et le déroulement du programme est arrêté.

Par exemple :

```
In [ ]: n=-1
        assert n>=0, 'Le nombre n\'est pas positif !'
```

```
In [ ]: n=-1
        assert n>=0, 'Le nombre n\'est pas positif !'
```

Vocabulaire corps, condition de maintien dans la boucle, condition d'arrêt, itération, tests unitaires.