

Iteration - while loops

# Iteration - for and while loops

Iteration occurs when we instruct the computer to carry out a task over and over again by repeating a section of code.

The piece of code is called a loop.

Python provide built in support for two different kinds of iteration statements/loops.

- The **while** loop
- The **for** loop

# While Loops

A while loop keeps repeating while a certain condition is true.

It stops when the condition is false. If this doesn't happen then the loop could go on forever.

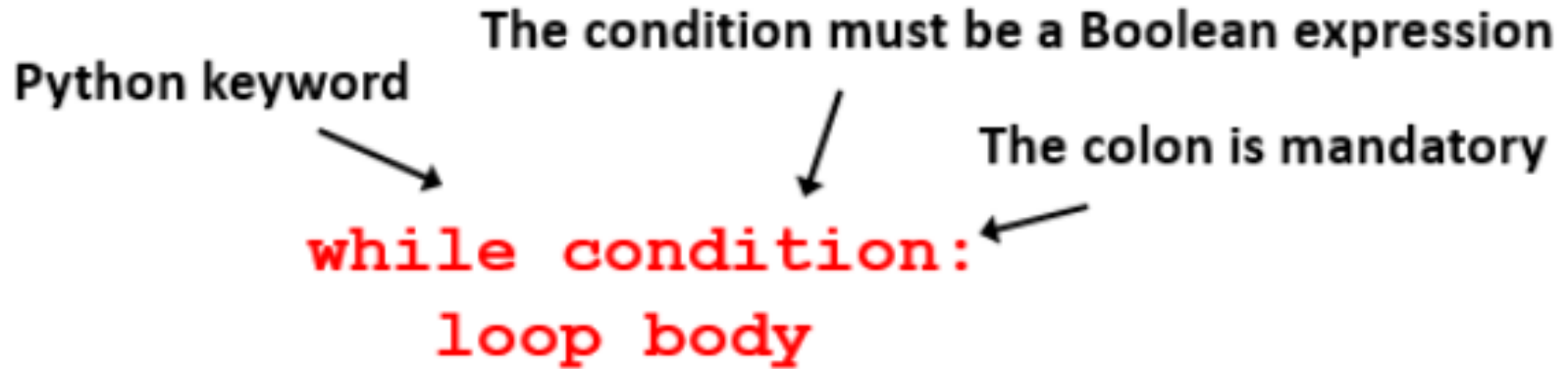
```
counter = 0
while counter < 7:
    print(counter)
    counter+=1
print("I'm glad that loop is finished!")
```

<https://repl.it/join/kfzhdbyr-suzannelinnane>

# The while loop

Python keyword      The condition must be a Boolean expression      The colon is mandatory

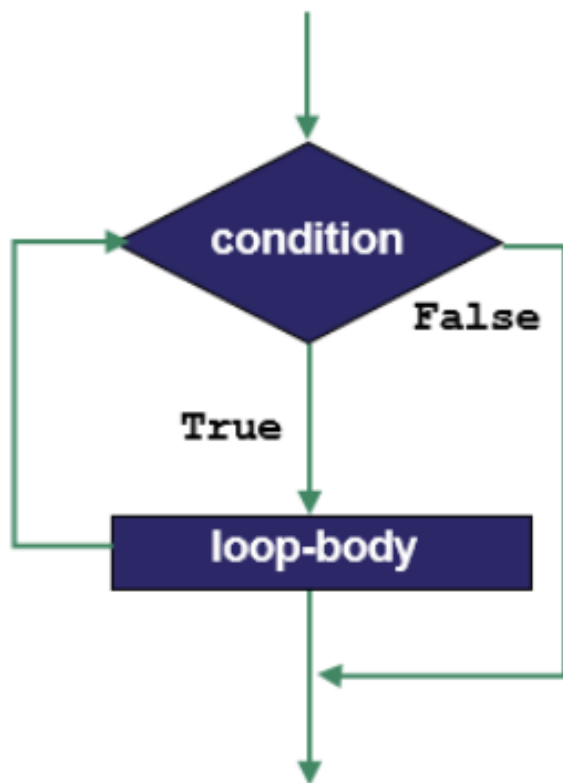
**while condition:**  
    **loop body**

A diagram illustrating the syntax of a Python while loop. It shows the code 'while condition:' followed by an indented 'loop body'. Three annotations with arrows point to specific parts: 'Python keyword' points to 'while', 'The condition must be a Boolean expression' points to 'condition', and 'The colon is mandatory' points to the colon after 'condition'.

The while loop begins with the keyword `while`, followed by a condition (made up by the programmer!)

If the result of the statement is `True`, then the code in the loop body is executed.

When Python reaches the last line of the loop body, it loops back to the condition which is evaluated again. This continues until the result of the condition is `False`.



*Flow chart illustration of while loop*

# The while loop

It is the programmer's responsibility to ensure that the loop body contains a line of code that will cause the loop condition to eventually become False.

Otherwise the loop will never terminate. Such loops are called infinite loops.

It is also possible for the loop body never to be executed.

This happens when the condition evaluates to False before the first iteration.

```
1.  # Simple while loop
2.
3.  # Initialise a loop counter
4.  counter = 1
5.
6.  # Loop 10 times
7.  while counter <= 10:
8.      print("Hello World")  # Display a message
9.      counter = counter + 1 # Increment the counter
10.
11. # This line is only executed once
12. print("Goodbye")
```

*Simple while loop demo.*

# Uses of while loops

Sometimes repetition is required that is not for a fixed number of iterations.

For example checking a user password until it is correct.

This is often referred to as a sentinel controlled loop and is frequently used for validation (ways to check something is correct).



# Example:

Enter an exam mark which must be between 0 to 100.

If the user enters a mark that falls outside this range, then they are asked to re-enter it until it is inside the range.

```
mark = float(input("Enter an exam mark  
between 0 and 100: "))  
while mark < 0 or mark > 100:  
    print("Error, mark outside range.")  
    mark = float(input("Re-enter mark: "))  
print("You entered", mark, "- well done!")
```

<https://repl.it/join/xiajaynb-suzannelinnane>

Note the following in the code above:

- The loop control variable `mark` is created and given a variable type `float`, otherwise it would be considered to be a string and the condition could not be applied.
- The use of the Boolean operator `or` in the `while` loop and the colon at the end.
- The indented block of two lines that forms the `while` clause i.e., the code that is executed if the `mark` entered is outside the range.
- How the loop control variable `mark` is changed within this block so that the condition can be re-tested and at some point, will evaluate to `False`, allowing the loop to end.
- The use of the sentinel controlled `while` loop means it may run 0 times (if the grade is within range at first), or many times, if the user keeps entering the wrong grade.

Example: Enter numbers that are to be added together. When the user enters 0, the loop terminates

```
num = int(input("Enter a number, 0 to  
finish: "))  
total = num  
while num != 0:  
    num = int(input("Enter another number,  
0 to finish: "))  
    total += num  
print("The total is: ", total)
```

Here is some sample input and output:

```
Enter a number, 0 to finish: 7  
Enter another number, 0 to finish: 9  
Enter another number, 0 to finish: 8  
Enter another number, 0 to finish: 66  
Enter another number, 0 to finish: 0  
The total is: 90
```