1. What are the main functional differences between a ‘while’ and a ‘for’?

A ‘while’ keeps looping back to the start of a block of statements; as long as the loop remains a true value. In contrast, the ‘for’ loop is designed for the repetition in a set order and will go through it as many times as stated. The ‘while’ can also replace ‘for’ with its counter loop; but more code is needed and will run slower.

1. What’s the difference between ‘break’ and ‘continue’?

A ‘break’ will stop a loop immediately and log out of it; while ‘continue’ will lead its loop to the very beginning of a while/for loop.

1. When is a loop’s ‘else’ clause executed?

If there is one ‘else’ clause in the while/for loop, the program will keep running only if it meets the else’s condition(s). If a ‘break’ is put in the middle of a for/while loop and doesn’t fall into the ‘else’ condition; the program will stop immediately without running any ‘else’ statement.

1. How can you code a counter-based loop in Python?

To code a counter-based loop, we normally use ‘range’ in a while statement or a for loop. Other than to indicate the number of repetition needed in a ‘for’, ‘range’ iterates item on demand or anywhere you need a series of integer. This method is not preferred; however, we use it to step across all items in a sequence.

1. What can a range be used for in a for loop?

Whenever one need to state a fixed number of repetition, to switch up lists, to shorten a successive order of to scan by offset, we use ‘range’ in a ‘for’ loop.