Q: What are the main functional differences between a while and a for?

A: The while loop is a general looping statement, but the for is designed to iterate across items in a sequence or other iterable. Although the while can imitate the for with counter loops, it takes more code and might run slower.

Q: What’s the difference between break and continue?

A: The break statement exits a loop immediately (you wind up below the entire while or for loop statement), and continue jumps back to the top of the loop (you wind up positioned just before the test in while or the next item fetch in for).

Q: When is a loop’s else clause executed?

A: The else clause in a while or for loop will be run once as the loop is exiting, if the loop exits normally (without running into a break statement). A break exits the loop immediately, skipping the else part on the way out (if there is one).

Q: How can you code a counter-based loop in Python?

A: Counter loops can be coded with a while statement that keeps track of the index manually, or with a for loop that uses the range built-in function to generate successive integer offsets. Neither is the preferred way to work in Python, if you need to simply step across all the items in a sequence. Instead, use a simple for loop instead, without range or counters, whenever possible; it will be easier to code and usually quicker to run.

Q: What can a range be used for in a for loop?

A: The range built-in can be used in a for to implement a fixed number of repetitions, to scan by offsets instead of items at offsets, to skip successive items as you go, and to change a list while stepping across it. None of these roles requires range, and most have alternatives—scanning actual items, three-limit slices, and list comprehensions are often better solutions today