Loop

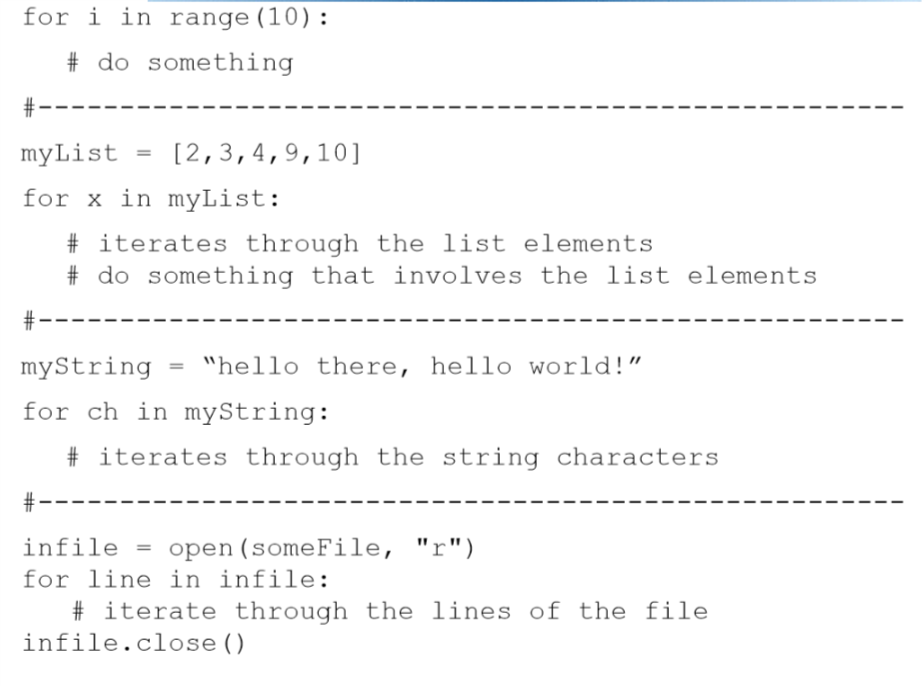
Objectives:

1. Interactive loops
2. Sentinel loops
3. Nested loops
4. Post-test loops (the condition comes after the body of the loop)
5. Loop and a half
6. Break statement
7. Continue statement

Good sentences:

1. The while loop requires us to manage the loop variable I by initializing it to 0 before the loop and incrementing it at the bottom of the body.
2. The outer while loop iterates once for each line of the file
3. For each iteration of the outer loop, the inner for loop iterates as many times as there are numbers on the line.

The type of for loop:



Getting out of an infinite loop

1. Control + c
2. Control + alt + delete
3. Push the reset button

Interactive loop:

Interactive loop allows a user to repeat certain portions of a program on demand.

At each iteration of the loop, ask the user if there is more data to process. We need to preset it to ‘yes’ to go through the loop the first time.

Sentinel loop:

A sentinel loop continues to progress data **until reaching a special value** that signals the end.

This special value is called **the sentinel**.

The first item is the sentinel, the loop terminates and no data is processed. This is sometimes called the **priming read**, since it gets the process started.

P: you can use the string to store the value, once the string that you input is empty, the program is finished.

Nested loop:

When designing nested loop:

1. Design the outer loop without worrying about what goes inside
2. Design what goes inside, ignoring the outer loop
3. Put the pieces together, preserving the nesting

Post-test loop:

Input validation: set a condition to get an invalid input

// The program keeps asking the correct input / another value when the user types an incorrect input

Conclusion:

1. **when the condition test comes after the body of the loop it is called a post-test loop**
2. a post-test loop always executes the body of the code at least once
3. Python doesn’t have a built-in statement to do this (such as do … while … in C language), but we can do it with a lightly modified while loop (by using a break)

Eg :

**While True:**

**xStr = input (“Enter a number (<Enter> to quit):”)**

**if xStr == “”:**

**break**

loop and a half:

The loop exit (break) is in the middle of the loop body. This is what we mean by a loop and a half.

while True:

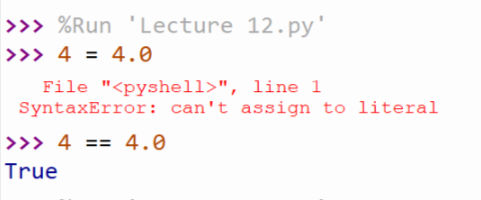
# get next data item

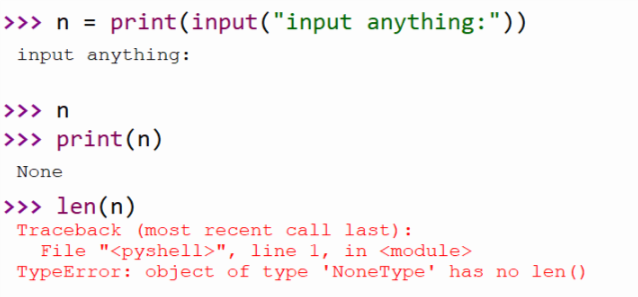
# if the item is the sentinel: break

# process the item

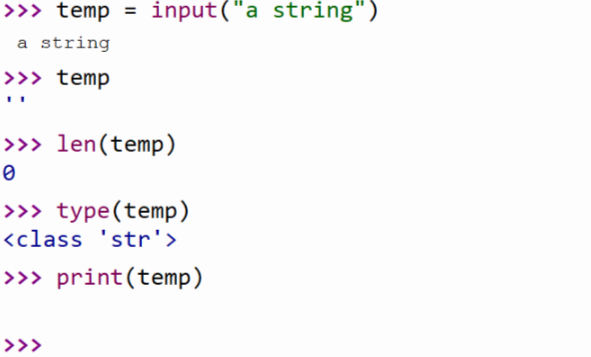
Break statement

Continue statement





This one is different with tutor, the input function returns a value which is NONE, and I still print it out. Thus, n becomes a NONE value instead of the empty string.



It is something, but it is empty, it is not None

**Empty is also something**

**None means different with the empty**

If I don’t use round function for some float number, there would exist situation of the mixture of some integer and some float number. It is because the integer places and the decimal place are store in different memory location. There are simple structure and difficult structure. When the number is just integer, it just uses the simple structure, otherwise it uses difficult structure for the float number.