Lecture 13: Introduction to Computer Programming Course - CS1010

DEPARTMENT OF COMPUTER SCIENCE | 10/22/2019

Rensselaer

Announcements

- Exam 2 will now be on November 4th.
- This is 2 Weeks from now.
- Back exam will be posted by the end of this week.

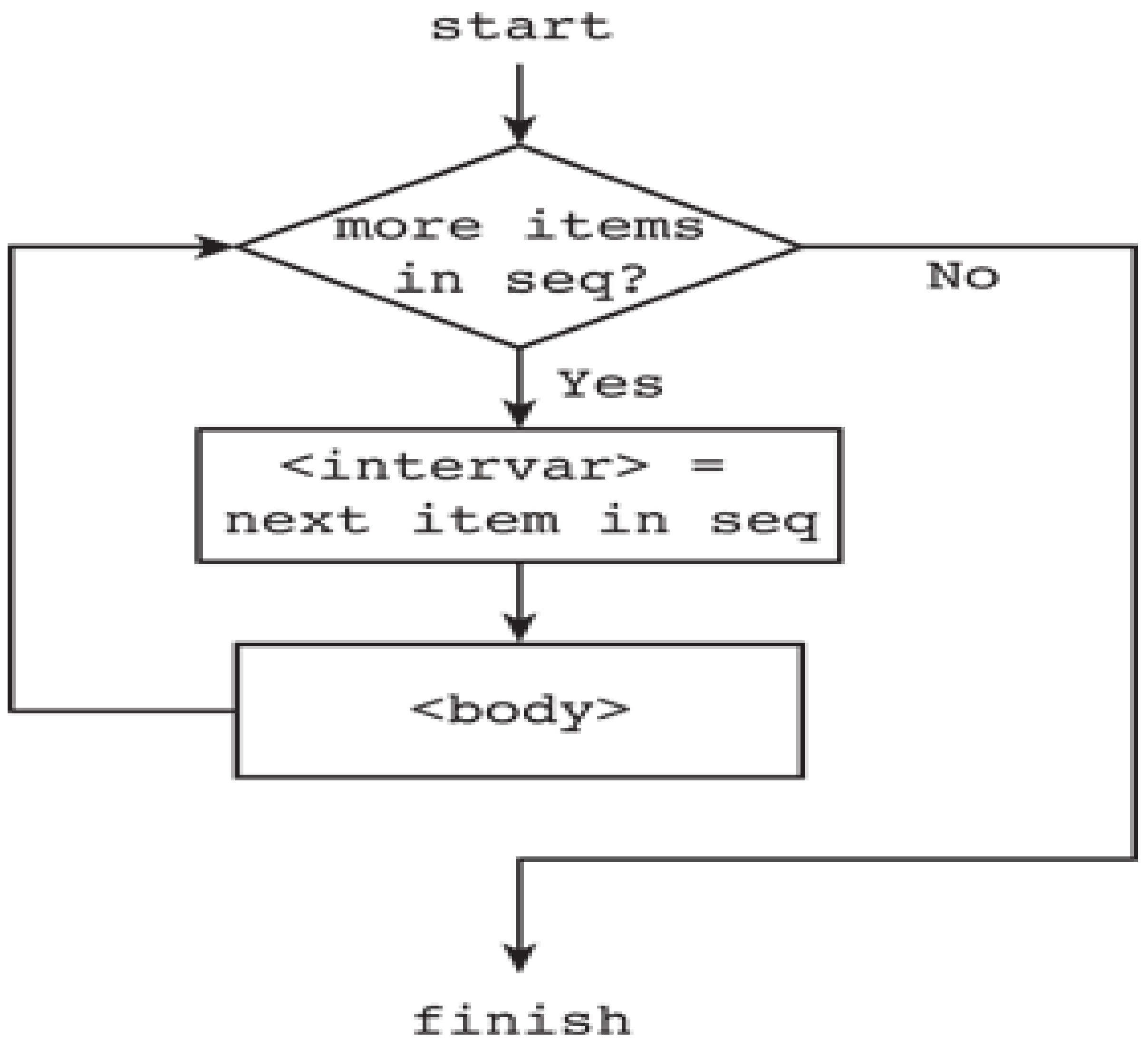
Goals for today

- Loops in general
- While Loops
- Problems on While Loops

What is a Loop

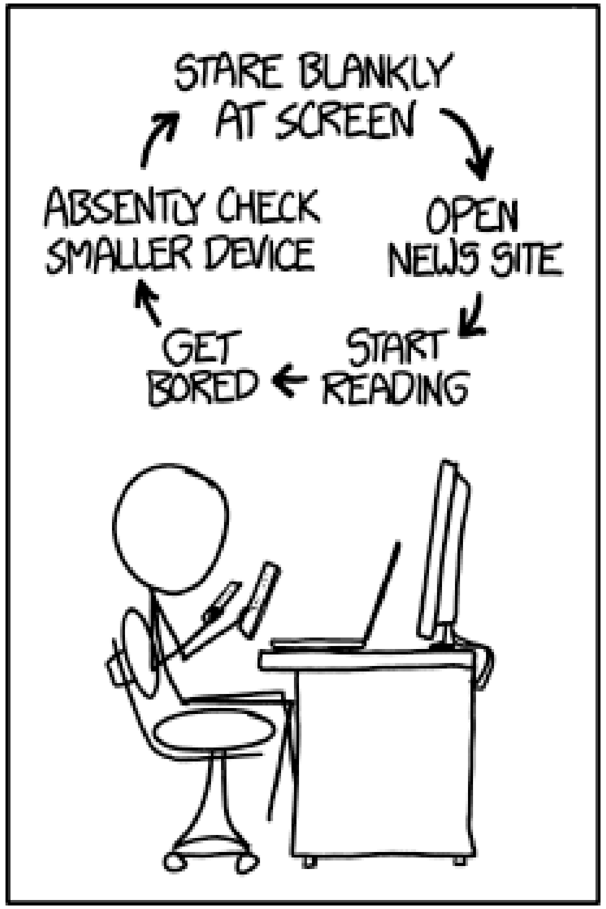
- A loop is a fundamental programming technique that is used in writing programs.
- In computer programming, a loop is a **sequence** of instructions that are repeated until a certain condition is reached.
- For example
  - We need to get an item of data and/or change it,
  - Based on some condition it is checked whether a counter has reached a prescribed number.
  - If NOT, then the next instruction in the sequence is an instruction to return to the first instruction in the sequence and repeat the sequence.
  - If the condition is reached, the next instruction tells the execution to branch outside the loop.

Loop Structure



Why do we Loop?

- Repetition
  - Generally used to access and modify information in a List
  - Allows us to repeat a block of code
  - Generally required for most sophisticated programming tasks



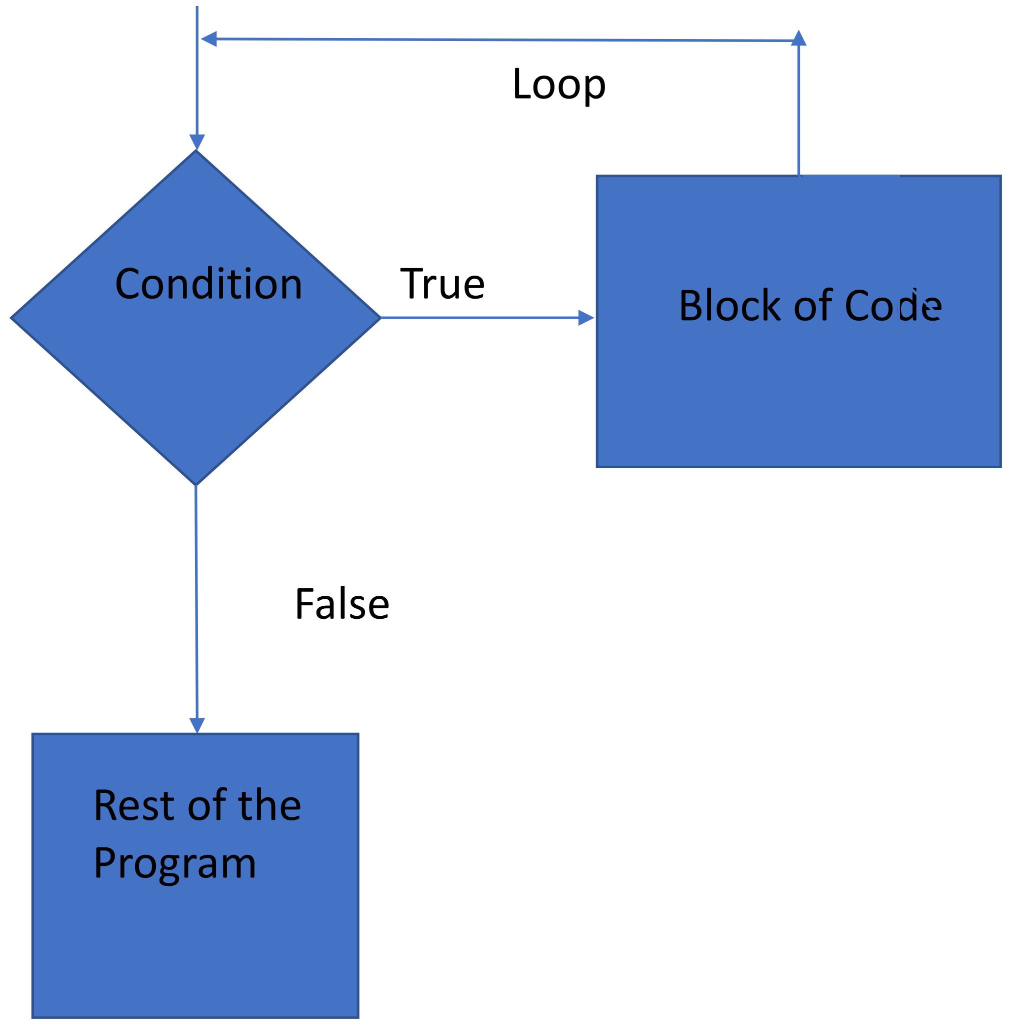
Types of Loops

- While Loops: Test a condition
- For Loops: Run for some pre-defined number of times
- Iteration by index (Another form of for Loops)
- Nested Loops

While Loops

- While loop is used to run a block of code repeatedly until a given a condition is satisfied.
- This repetition is referred to as iteration.
- When the condition becomes false, the line immediately after the loop in the code is executed.

- Execution begins



Syntax

```
while expression(s):
    statements
```

Can be combined with else:

```
while condition(s):
    statements
else:
    statements
```

Python uses indentation as its method of grouping statements.

Python Program to Illustrate While Loop

- General Procedure:
- 1. **Initialize** a counter variable
- 2. Specify the **condition** for while
- 3. Specify the required **actions**
- 4. **Increment or Decrement** the counter

- Let’s check in spyder!

Calculation of Bacteria Growth Rate

- Consider the function:

$$f(t+1) = f(t) + rf(t)$$

- This is the equation for bacteria growth such that $f(t)$ represents the population at time t. Here r is the growth rate.

- Given a certain initial population and growth rate, we want to know when will this specie double its population.

Insights from Results

- Time was updated inside the loop so its value is the value from the last iteration.
- Loop condition was population<2000:
  - The variable population exceeded 2000 within the loop
  - In the next iteration i.e. when the variable exceeded 2000, the execution stopped
- What if we want to stop exactly at 2000:
- We can say in the condition:
  - while population == 2000
- What is the issue in the above statement?

Infinite Loops

- Execution can go on forever:
  - In Spyder Go to: Console push the red square to ‘kill’ the program
- When deciding for loop condition be careful of the execution
  - Try to avoid infinite loops

```
# DO NOT RUN THIS CODE!!!!
while True:
    print("I'm stuck in an infinite loop!")
```

Loops with Lists

- We use loops to repeat a specific operation on every element of a list.
- Given a list of velocities of different falling objects in metric unit (m/sec):
- **Velocities = [0.0, 9.81, 19.62, 29.43]**
- Covert these to ft per second (multiply by 3.28). The index of the lists specifies the object.
- Print the new output for every object.

Controlling Loops

- The basic rule is that all code within the body of a loop is executed.
- Python provides controlling of Loop iteration using the following statements:
    - Break
    - Continue
    - Pass

Controlling Loops

- break, continue, and pass statements are used in loops to add additional functionality for various cases.

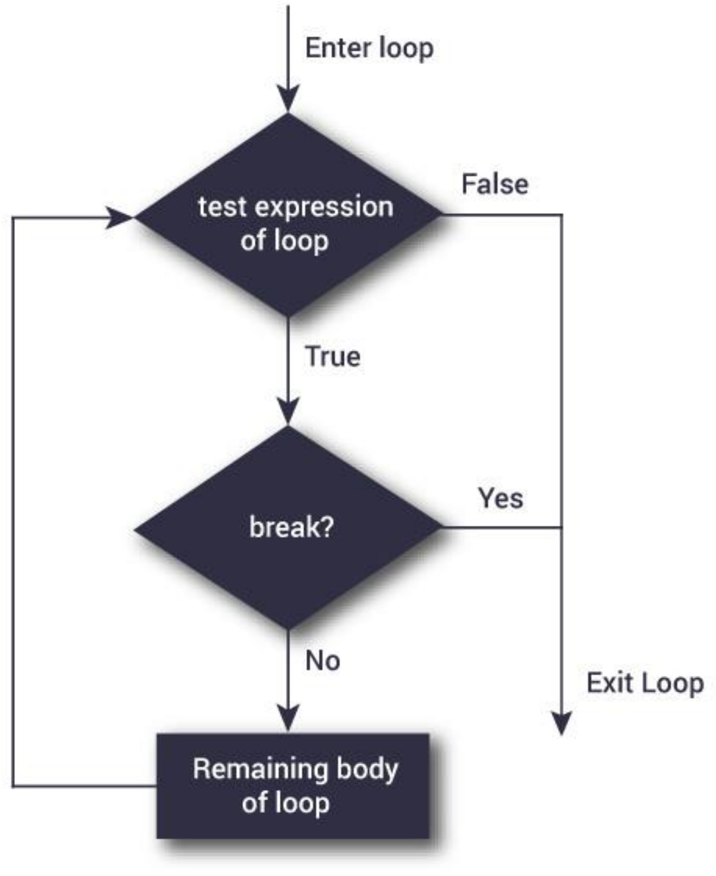
- These three statements are defined as:
    - <u>break</u>: Breaks control out of the current next enclosing loop.
    - <u>continue</u>: Takes control to the top of the next enclosing loop.
    - <u>pass</u>: Does nothing at all.

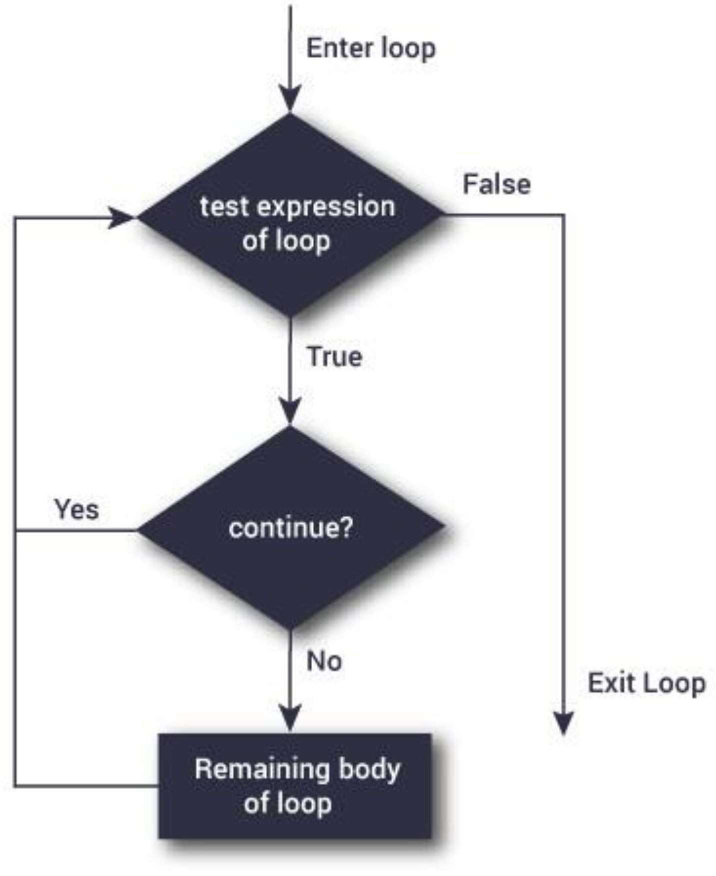
Controlling Loops Continued…

- break and continue statements can appear anywhere inside the loop’s body.
- We usually put them further nested in conjunction with an if statement to perform an action based on some condition.
- Check in Spyder!

```
while test:
    code statement
    if test:
        break
    if test:
        continue
else:
```

Flowchart of Break and Continue





More Examples of break and continue

Problem 1(a)

Write a while statement that prints integers from zero to 5.

Problem 1(b)

Write a while statement that outputs only odd integers from 0 to 10.

Nested While

- Just like if statements we can have Nested While statements
- Output of the program on the right?
  - For every value in L the rest of the values will be repeated
  - Let’s check

```
L = [2, 21, 12, 8, 5, 31]
i = 0
while i < len(L):
    j = 0
    while j < len(L):
        print(L[i], L[j])
        j += 1
    i += 1
```

Problem 1

- Write a Python program to count the number of even and odd numbers from a series of numbers.

- *Sample numbers* : numbers = (1, 2, 3, 4, 5, 6, 7, 8, 9)

- *Expected Output* :
  Number of even numbers : 5
  Number of odd numbers : 4

Problem 2

- Given a list of numbers; print the numbers and find their sum.
- Solution: Loop through the numbers.
- Keep track of the sum in each iteration using a variable.

Problem 3

- Write a Python program to find those numbers which are divisible by 7 and multiple of 5, between 1500 and 2700 (both included).

- Solution:
- Loop through all elements in the given range
- Check divisibility by 5 and 7
- Output the elements that fulfill the condition

Problem 4

- The following list represents the population of New York State (in hundreds of thousands of people) for the US Census in 1790, 1800, 1810, etc., all the way through 2010.
- census = [340, 589, 959, 1372, 1918, 2428, 3097, 3880, 4382, 5082, \
-             5997, 7268, 9113, 10385, 12588, 13479, 14830, 16782, \
-             8236, 17558, 17990, 18976, 19378]
- Write code to find the average percentage change from one decade to the next, across all decades. For example, the change from 1790 to 1800 is (589 - 340) / 340 * 100 = 73.2% and the change from 1800 to 1810 is (959 - 589) / 589 * 100 = 62.8% so the average across just these two decades is 68.0%.

Problem 5:Example of nested loop

Write a Python program to construct the following pattern, using a nested loop.

```
*
* *
* * *
* * * *
* * * * *
* * * *
* * *
* *
*
```

Solution

- Break the solution into two parts:
  - The upper triangle
  - The lower triangle
- Think of two variables:
  - The lines containing stars
  - The stars themselves
- Loop both the variables
  - Increment for the first(upper) triangle
  - Decrement for the second (lower) triangle

Next Class

- Problems on While Loops
- In Class exercise