Programming Fundamentals

Lecture 4 – Python Loops

Iresh Bandara

ROBERT GORDON UNIVERSITY ABERDEEN
TEF Gold
INFORMATICS INSTITUTE OF TECHNOLOGY

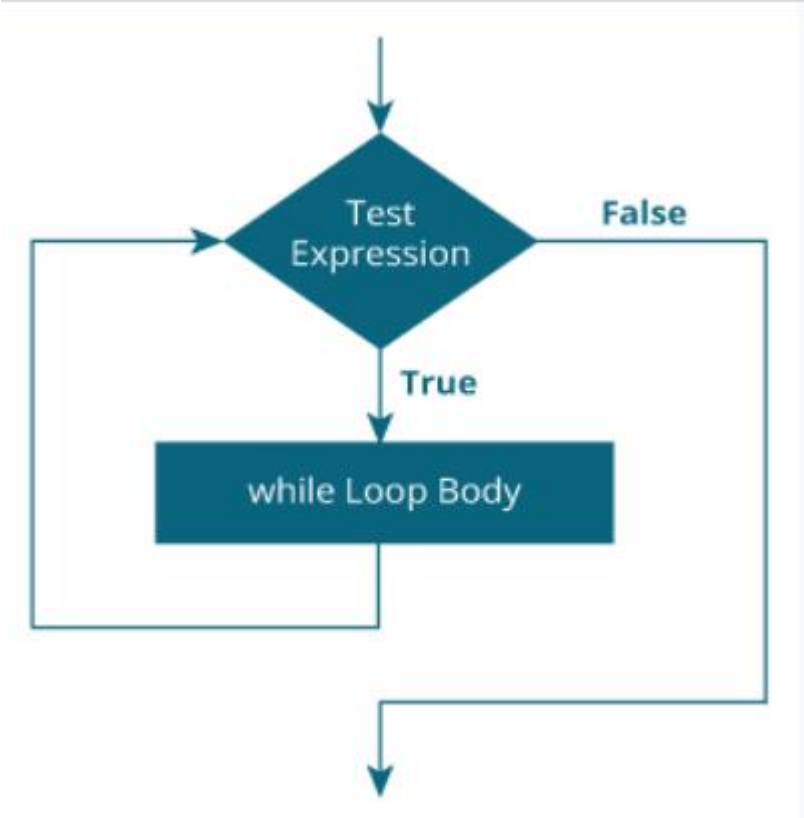
Learning Outcomes

- This lecture addresses LO1,LO2 and LO4 for the module
- On completion of this lecture, students are expected to explain and apply
  - While loops
  - Common issues
  - While - Else
  - While - Break
  - While – Continue
- Examine common issues due to While loops
- Analyse program flows based on While loops + conditions + Break + Continue

Recap: basic programming Constructs

- **Sequence**
  - Execution of set of instructions one after the other.
- **Selection**
  - An option of statements is provided and a condition is used to decide which option is chosen. **If, elif** and **else**
- **Iteration**
  - A group of statements is executed repeatedly until a condition is met - **while** loop and **for** loop

While loop



1. Condition is evaluated before enters the loop body
2. If the condition **true** only, enter the loop body
3. Execute set of sequential or non sequential statements inside the body
4. After the execution of body **1** and **2** happens again while condition is false
5. When the condition is false, loop will be terminated

Note: if the condition is false during the first evaluation, it will not enter the body

While loop structure

```
counter = 0
number = 100
while counter < 10 :
    print(number+counter)
    counter = counter + 1
print('outside')
```

Condition is true if the counter is less than 10 only. Otherwise it will not enter the body

Update counter variable within loop

Proper indentation indicates a "block" of code and it is a **must**

While loop – Common issues 1

- **Incorrect Test Condition**
- Loop body only executes if the condition is TRUE
- If bal is initialized as less than the TARGET and should grow until it reaches TARGET, Which version is correct?

```
while bal >= TARGET :
    year = year + 1
    interest = bal * RATE
    bal = bal + interest
```

```
while bal < TARGET :
    year = year + 1
    interest = bal * RATE
    bal = bal + interest
```

While loop – Common issues 2

- **Infinite loops**

```
counter = 1
while (counter != 100) :   # Runs forever
   print(counter)          # counter not updated
```

```
counter = 1
while (counter != 100) :        # counter increment 1,3,5,
   print(counter)               # 7,9,11……99,101.
   counter = counter + 2
```

- Make sure it reaches the breaking point

While loop – Common issues 3

- **Off by One error/ how many times you want to run the loop?**

```
counter = 0
while (counter < 10) :     # Q1.How many passes?
      counter = counter + 1
```

```
counter = 1
while (counter < 10) :     # Q2.How many passes?
      counter = counter + 1
```

```
counter = 0
while (counter <= 10) :     # Q3.How many passes?
      counter = counter + 1
```

```
counter = 1
while (counter <= 10) :     # Q4.How many passes?
      counter = counter + 1
```

Exercise 1

- Write a program that contains a while loop that will sum the int values entered by a user until the user enters a -1. Then print the total

- Print the pattern using while loops

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

```
row = 1
while row <= 4:
    col = 1
    while col <= row:
        print("*", end="")
        col += 1
    print()
    row += 1
```

Exercise 2

- Trace the issue and find a fix for it

```
health = 100
gain = 0
damage = 3

while health != 0:
      gain+= 1
      health -= damage
      print("current gain ", "gain")

print("Done")
```

health>0

While with Else

- Specific feature of Python
- If the condition of the while loop is false, it will execute else

```
count=0
while(count<5):                    #output: 0,1,2,3,4,outside
    print(count)
    count +=1
else:
    print("outside")
```

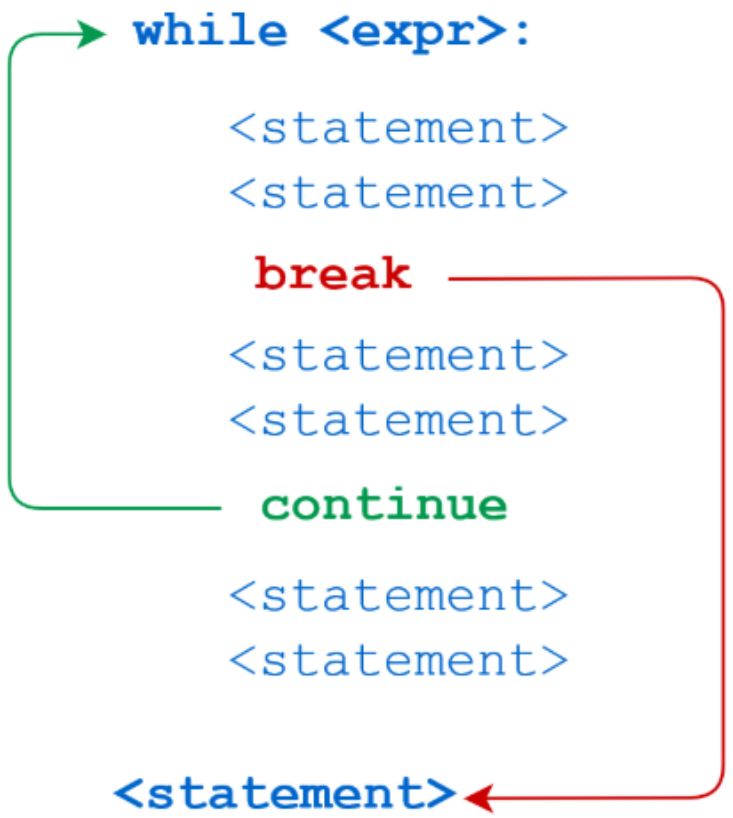
Break and Continue

- **Break** : If the developer wants to terminate the loop even before the initial condition becomes false. Break can be used with a special condition and when that special condition is true, loop will be terminated.

- **Continue**: After the execution of continue, loop stops executing the current iteration without executing statements right after continue and proceed to the next iteration. Continue also can be triggered when a specific condition is true.

Break and Continue

- Spot the difference



Break and Continue

```
var = 10
while var > 0:
    var = var -1
    if var == 5:
        continue
    print('Current value :', var)
else:
    print(“bye!“)’

#print values 9,8,7,6,4,3,2,1,bye
```

```
var = 10
while var > 0:
    var = var -1
    if var == 5:
        break
    print('Current value :', var)
else:
    print("bye!“)

#print values 9,8,7,6
```

- Check how Else is working when WHILE consist of a break or continue

Summary

- Repetition is a key construct in programming is while is one way of fulfilling it
- Main components of a while loop are the condition and the body
- While the program fulfills the condition, it executes the body. Thereafter, it exits the loop
- Common issues when handling loop : Incorrect test condition, infinite loop and off by one
- While – Else is a unique combination which is offered by Python
- Break is used to terminate the loop under a certain condition
- Continue stops the current iteration and move to the next