IT Department

Loops and Iteration (while loop)

Soma Soleiman Zadeh

Programming I (IT 117)

Fall 2024 - 2025

Week 9

February 12, 2025

Outline

- **Loops**
  - **Definite** Loop (**Counting** Loop)
  - **Indefinite** Loop (**Conditional** Loop)
- **while** Loop
- Which loop is better? **while** or **for** loop?

Definite Loops vs. Indefinite Loops

- There are two types of loops in Python:
  1. **Definite** (**Counting**) Loops → **for** loop
     - **Exact number of iterations** to do.
     - Iterates through the members of a set (set of numbers, characters, strings).
  2. **Indefinite** (**Conditional**) loops → **while** loop
     - Not definite number of iterations.
     - Iterates while some condition is **True**.

Why do We Need while Loop?

- Suppose we want to write a program that can **compute the sum of a series of numbers entered by the user**.
- User should be able to enter any size set of numbers.
- So what we need to know is
  - the **sum** and
  - how many numbers have been added.

Why do We Need while Loop?

- We need some sort of loop. If the user wants to enter **n** numbers, the loop should execute **n** times.
- We need to calculate and update the **sum** at each iteration.

```
n = int(input("How many numbers do you have? "))
sum = 0

for i in range(n):
    x = int(input("Enter a number: "))
    sum = sum + x
print("Sum of numbers is: ", sum)
```

```
How many numbers do you have? 5
Enter a number: 6
Enter a number: 9
Enter a number: 2
Enter a number: 7
Enter a number: 5

Sum of numbers is:  29
```

Why do We Need while Loop?

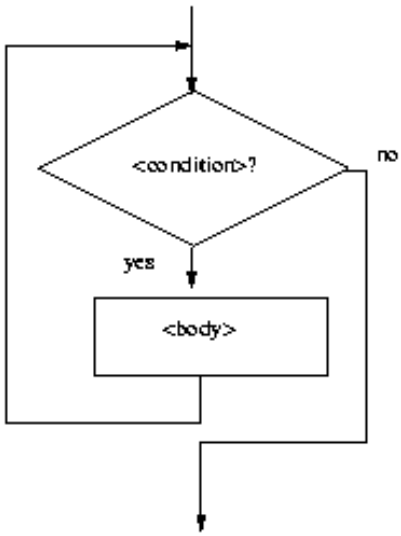
- That program is correct, but there is a point: **before executing the for loop**, you need to know how many numbers the user wants to enter.
- **What to do if we don't know how many numbers are entered by user?** and the computer takes care of counting how many numbers there are.
  - There is another tool; **while** loop.
  - The **indefinite or conditional** loop (**while** loop) is for executing the loop *as long as a condition is True*.
  - So, there is no need to know the number of iterations before executing the **while** loop.

while Loop

- A **while-loop** is used to execute a block of statement **as long as a condition is True**.
  - **condition** → a Boolean expression
  - **Body** of loop → The code block that is repeated in a loop
- General syntax of **while** loop:

```
while (condition) :
        code block
```

While Loop



- The **condition** is tested at the top of the loop. As long as the condition is True, the loop body will be executed.
- As the condition becomes False, the loop ends and the code after the loop will be executed.

While Loop

- Using **for** loop to print numbers from 0 to 10:

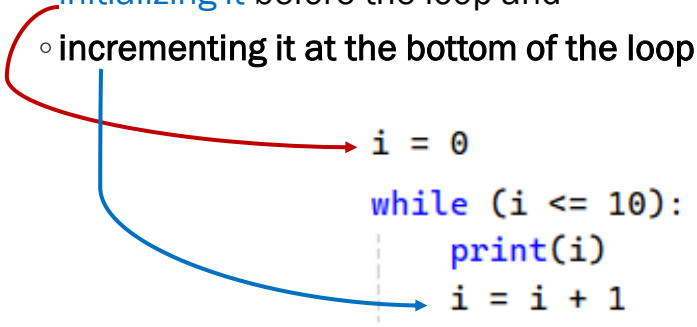
```
for i in range(11):
    print(i)
```

- Using **while** loop to print numbers from 0 to 10:

```
i = 0
while (i <= 10):
    print(i)
    i = i + 1
```

Loop Variable in While Loop

- In the **`for`** loop, increasing loop variable is handled automatically.
- The **`while`** loop requires us to manage the loop variable `i` by
  - initializing it before the loop and
  - incrementing it at the bottom of the loop body.



What happens with this code?

```
i = 0
while (i <= 10):
    print(i)
```

Output

This loop is called **infinite loop** that executes forever. Because the part of code that is for increment of looping variable (i) is missing and the condition is always True.

Solution:

```
i = 0
while (i <= 10):
    print(i)
    i = i + 1
```

```
i = 0
while (i >= 10):
    print(i)
    i = i + 1
```

Output

The body of this loop is not executed even once. Because the condition is always False.

Solution: Make sure the loop condition is in accordance with the initial value of loop variable (i) and the purpose of using loop.

While Loop – Example

- Try to write Python code to get the following output:
  - Once by using a **for** loop
  - Once by using a **while** loop

```
IT Department
IT Department
IT Department
IT Department
IT Department
```

```
for i in range(5):
    print("IT Department")
```

```
i = 0
while (i<5):
    print("IT Department")
    i = i + 1
```

What is the output of this code?

```
sum = 0
flag = True
while (flag):
    x = int(input("Enter a number: "))
    print("Your entered number is:",x)
    sum = sum + x
    message = input("Do you continue entering another number? (yes/no)")
    if (message=='no'):
        flag = False
print("Sum of entered numbers is: ", sum)
```

In this code we use a variable named **flag** to control when the **while** loop is ended. Before while loop, the **flag** is initialized to ***True***. So, in the **while (flag)**, the condition is True and the loop is always iterated until in loop body we change the value of the flag to ***False***. So, the loop is ended as its condition, **flag**, becomes ***False***.

Another Way of Writing the Previous Code

```
sum = 0
while (True):
    x = int(input("Enter a number: "))
    print("Your entered number is:",x)
    sum = sum + x
    message = input("Do you continue entering another number? (yes/no)")
    if (message=='no'):
        break
print("Sum of entered numbers is: ", sum)
```

In this code instead of using a **flag** variable to control iterations of **while** loop, we directly use the ***True*** value in the **while** condition. So, the condition is always ***True*** and the loop is executed forever until we use a **break** in the loop body to end the loop immediately.