



The Islamic University of Gaza
Faculty of Engineering
Computer Engineering Department



Introduction to Computers

LAB #5

Loops

Eng. Hashem M. Hejazy

Eng. Mohammed O. AlHabbash

FALL 2026

Introduction

A **loop** can be used to tell a program to execute statements **repeatedly**. In other words, A loop statement allows us to execute a statement or block of statements **multiple times**. Python provides two types of loop statements: **while** loops and **for** loops.

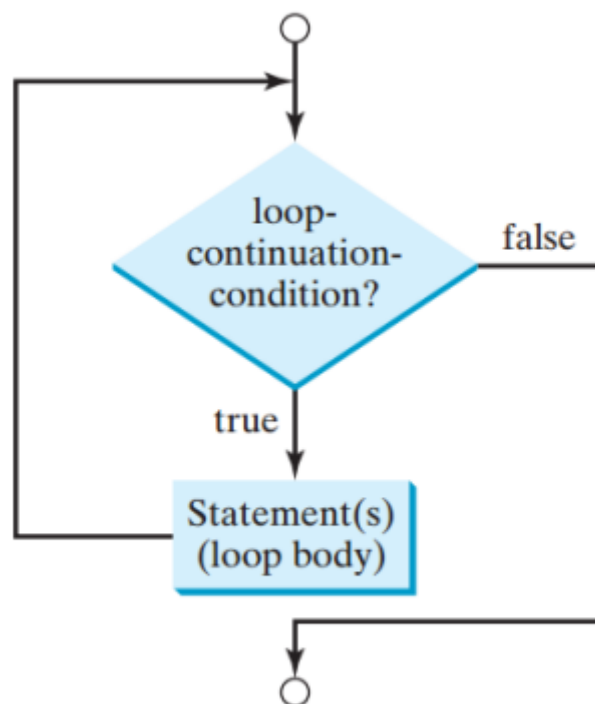
while loop

A **while** loop executes statements repeatedly as long as a **condition remains true**. **while** loop is a condition-controlled loop, it is controlled by a true/false condition. It tests the condition before executing the loop body. The syntax for the **while** loop is:

```
while loop-continuation-condition:  
    # Loop body  
    Statement(s)
```

Remember: All the statements indented by the same number of character spaces after a programming construct are considered to be part of a single block of code.

Here is the flowchart for the **while** loop:



Example #1

Write a Python program that prints "Hi Python!" 100 times to the console:

```
count = 0
while count < 100:
    print("Hi Python!")
    count = count + 1    # or use count += 1
print("Done!")
```

When the condition is tested and the result is false, the loop body will be skipped and the first statement after the while loop will be executed.

Example #2

Write a Python program that prints the numbers from 0 to 9.

```
count = 0
while count < 10:
    print(count)
    count += 1
print("Done!")
print(count)    # prints 10
```

Example #3

Write a Python program that prints the sum of numbers from 1 to 10.

```
sum = 0
i = 1
while i <= 10:
    sum = sum + i    # or use sum += i
    print(i)
    i += 1
print(f"sum is {sum}!")
```

Indent mistake

Suppose the loop is mistakenly written as follows:

```
sum = 0
i = 1
while i <= 20:
    sum = sum + i
    print(i)
i += 1    # Indent mistake here
print(f"sum is {sum}!")
```

Here the statement `i = i + 1` is not in the loop body. This loop is **infinite** because `i` is always `1` and `i <= 20` will always be true.

Note that the entire loop body must be indented inside the loop.

for loop

A **for** loop iterates through each value in a **sequence**. **for** loop is a **counter-controlled** loop. A while loop can be written as follows:

```
i = initialValue # Initialize loop-control variable
while i < endValue:
    # Loop body
    i += 1 # Adjust loop-control variable
```

A for loop can be used to simplify the preceding loop:

```
for i in range(initialValue, endValue):
    # Loop body
```

Here is the example that prints `"Hi Python!"` 100 times, but written using for loop this time.

```
for i in range(0, 100):
    print("Hi Python!")
```

range function

There are three versions of **range** function:

1. **range(a, b)**: returns the sequence of integers from **a** to **b - 1**.
2. **range(b)**: returns the sequence of integers from **0** to **b - 1**. It is the same as **range(0, b)**.
3. **range(a, b, k)**: returns the sequence of integers from **a** to **b - 1** using the **step value k** (Difference between each number in the sequence).

```
for v in range(3, 9, 2):  
    print(v)
```

Output:

```
3  
5  
7
```

Example #4

Suppose the input is **2 3 4 5 0** (one number per line). What is the output of the following code?

```
number = 0  
sum = 0  
for count in range(5):  
    number = int(input("Enter an integer: "))  
    sum += number  
print(f"sum is {sum}!")  
print(f"count is {count}!")
```

break statement

You can use the keyword **break** in a loop to immediately **terminate** a loop. A **break** could be used in a **while** or **for** loop.

The following example illustrates the use of **break** keyword, if the sum of numbers is greater than or equal to 100 then **break** the loop and print the result:

```
sum = 0
number = 0
while number < 20:
    number += 1
    sum += number
    if sum >= 100:
        break
print(f"The number is {number}")
print(f"The sum is {sum}")
```

The output is:

```
The number is 14
The sum is 105
```

continue statement

You can use the **continue** keyword in a loop **skip all the remaining statements in the current iteration**. When it is encountered, it ends the current iteration, and program control goes to the end of the loop body. In other words, **continue** keyword **breaks out of an iteration**, while the **break** keyword breaks out of a loop. **continue** could be used in a **while** or **for** loop.

The following example prints the numbers from 0 to 100 except those numbers that divide by 7:

```
for i in range(100):
    if i % 7 == 0:
        continue
    print(f"{i} is {i}!")
```

Infinite loops

A loop becomes an **infinite loop** if a condition never becomes **False**. You must pay attention when using **while** loops because a loop may never end. Such a loop is called an **infinite loop**.

```
count = 0
while count < 5:
    print(f"Count is {count}!")
```

In the previous example, the condition is always **True**, so the loop never ends. To solve this problem, we need to increment the **count** variable every iteration, to make the condition **False** eventually.

Example #5

Write a program that sums the integer numbers entered by the user. The program requests the user to enter an integer until 0 is entered:

```
num = int(input("Enter numbers, zero ends loop: "))
sum = 0
while num != 0:
    sum += num
    num = int(input("Enter a number: "))
print(f"Sum is {sum}")
```

Example #6

Write a program that asks the user to type 10 integers, then prints the smallest value of the entered integers:

```
num = input("Enter a number: ")
min = num
for i in range(9):
    num = input("Enter a number: ")
    if num < min:
        min = num
print(f"Minimum number is {min}")
```

Example #7

Write a program to print the following pattern

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

```
star= "*"
for i in range(1,6):
    print((star+" ")*i)
```

Homework

1. Write a python program to let the user to **enter** the **start** number and the **end** number, then find the sum of the numbers (**including the start value only without the end value**) that **are divisible by 5**.
2. Write a program that reads integers from the user until the value 0 is entered. The program must determine and display the smallest integer that was entered. Assume at least one non-zero value will be entered.
3. Write a program that prompts the user to enter a positive integer and then determines whether the number is prime or not, displaying an appropriate message. (A prime number is divisible only by 1 and itself).
4. Write a program to print the following pattern.

```
1
2  2
3  3  3
4  4  4  4
5  5  5  5  5
```