

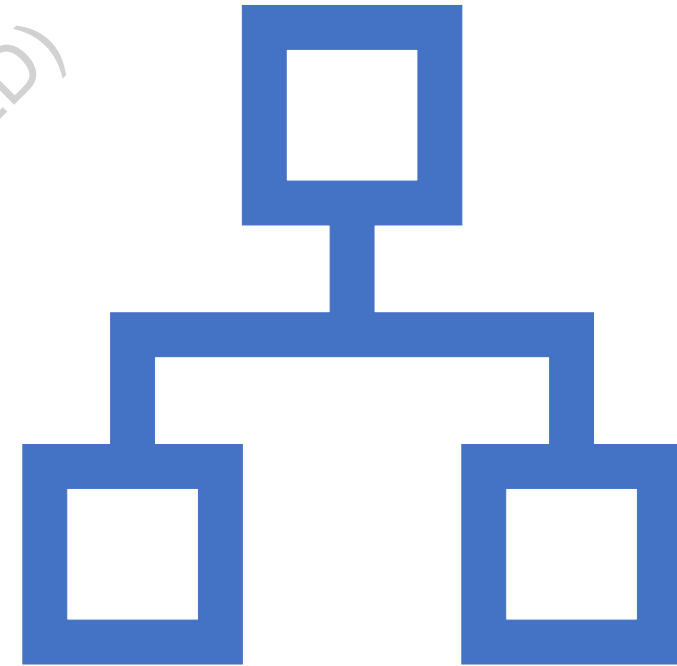


# Looping Structures in Python

By

Prof. Muhammad Iqbal Bhat

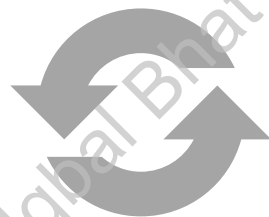
Government Degree College Beerwah



# Topics



Looping Structures in  
Python



For loop



While loop

Prof. M. Iqbal Bhat (JKHED)

Prof. M. Iqbal Bhat (JKHED)

# Looping Structures in Python:



Looping structures are an essential part of programming and are used to automate repetitive tasks.



In Python, looping structures are used to perform repeated operations on a set of data or a sequence of values.



Looping structures are efficient ways to perform repetitive tasks without writing the same code repeatedly.



They enable us to perform the same operation on a large dataset or a sequence of values, which would be difficult and time-consuming to do manually.



Looping structures give us more control over the flow of our program.



There are two types of loops in Python:

**for loop**

**while loop**

# For loop

for loop is used to iterate over a sequence of elements in Python

The sequence can be any iterable object such as a list, tuple, set, dictionary, or string

for loop iterates through each element of the sequence and executes the block of code for each element

The syntax of for loop in Python

```
for variable in sequence:
```

**variable** is the variable that takes the value of the current element in each iteration

sequence is the sequence of elements to be iterated over

# Examples of for loop:



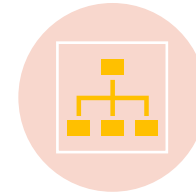
iterating over a list  
using for loop

```
numbers = [1, 2, 3, 4, 5]
for num in numbers:
    print(num)
```



iterating over a range  
of numbers using for  
loop

```
for i in range(1, 6):
    print(i)
```



iterating over a list  
with its index using  
enumerate() function

```
fruits = ['apple', 'banana', 'cherry']
for i, fruit in enumerate(fruits):
    print(i, fruit)
```

# While Loop:

---

A while loop is used to execute a block of code repeatedly until a given condition is satisfied.

---

The syntax of a while loop is:

```
while condition
# block of code
```

---

The while loop requires relevant variables to be ready before the loop starts. For example, a counter variable that is incremented or decremented in each iteration.

---

It is important to ensure that the condition in the while loop will eventually become False to avoid an infinite loop

---

To exit a while loop prematurely, we can use the break statement

---

# Examples of while loop:

- Printing numbers from 1 to 10 using a while loop:

```
num = 1
while num <= 10:
    print(num)
    num += 1
```

# Examples of while loop:

- Reversing a string using a while loop:

```
string = "Hello"  
index = len(string) - 1  
while index >= 0:  
    print(string[index], end="")  
    index -= 1
```



# Examples of while loop:

- Finding the factorial of a number using a while loop:

```
n = 5
factorial = 1
while n > 0:
    factorial *= n
    n -= 1
print(factorial)
```

Prof. M. Iqbal Bhat (JKHED)

# Examples of Loops:

- Checking if a number is prime or not using a for loop and conditional statements:

```
num = int(input("Enter a number: "))
is_prime = True
for i in range(2, num):
    if num % i == 0:
        is_prime = False
        break
if is_prime:
    print(num, "is a prime number")
else:
    print(num, "is not a prime number")
```

# Examples of Loops:

- Printing the Fibonacci series up to a given number using a while loop

```
num = int(input("Enter a number: "))  
a, b = 0, 1  
while b <= num:  
    print(b, end=' ')  
    a = b  
    b = a + b
```

Prof. M. Iqbal Bhat (JKHED)

Prof. M. Iqbal Bhat (JKHED)

# Break, Continue, and Pass Statements in Python

Prof. M. Iqbal Bhatt (JKHED)



# Break, Continue, and Pass Statements in Python

In Python, break, continue, and pass are control flow statements that help in altering the normal execution of a loop.

They are primarily used in loops like for and while, to control the flow of the program.

# Break Statement

The break statement is used to stop the execution of a loop prematurely. When the break statement is encountered in a loop, the program jumps out of the loop and continues executing the next statement after the loop.

The syntax of the break statement is as follows:

```
while expression:  
    statement(s)  
    if condition:  
        break
```

Example:

```
for i in range(1, 11):  
    if i == 5:  
        break  
    print(i)
```

# Continue Statement

The continue statement is used to skip the current iteration of a loop and move on to the next iteration. When the continue statement is encountered in a loop, the program skips the rest of the code in the loop for the current iteration and continues with the next iteration.

The syntax of the `continue` statement is as follows:

```
while expression:  
    statement(s)  
    if condition:  
        continue
```

Example:

```
for i in range(1, 11):  
    if i == 5:  
        continue  
    print(i)
```

# Pass Statement

The pass statement is used as a placeholder when a statement is required syntactically, but you do not want any command or code to execute. It is often used as a placeholder for functions or loops that will be implemented in the future.

The syntax of the `pass` statement is as follows:

```
while expression:  
    statement(s)  
if condition:  
    pass
```

Example:

```
for i in range(1, 11):  
    if i == 5:  
        pass  
    else:  
        print(i)
```

Prof. M. Iqbal Bhat (JKHED)



# Example of break statement:

- Find the sum of all the numbers in a list until a negative number is encountered using break statement

```
numbers = [2, 5, 1, 7, 3, -4, 6, 8]
```

```
sum = 0
```

```
for num in numbers:
```

```
    if num < 0:
```

```
        break
```

```
    sum += num
```

```
print("Sum of positive numbers:", sum)
```

# Example of break statement:

- Search for an element in a list until it is found using a break statement

```
fruits = ["apple", "banana", "cherry", "orange",  
"kiwi", "melon", "mango"]
```

```
search = input("Enter a fruit to search: ")
```

```
for fruit in fruits:
```

```
    if fruit == search:
```

```
        print(search, "found in the list")
```

```
        break
```

```
else:
```

```
    print(search, "not found in the list")
```

# Example of break statement:

- Usage of break statement in a program to check whether a number is prime or not

```
num = int(input("Enter a number: "))

for i in range(2, num):
    if num % i == 0:
        print(num, "is not a prime number")
        break
else:
    print(num, "is a prime number")
```

# Example of continue statement:

- Skip printing the even numbers in a list using continue statement

```
numbers = [2, 5, 1, 7, 3, 4, 6, 8]
```

```
for num in numbers:  
    if num % 2 == 0:  
        continue  
    print(num)
```

Prof. M. Iqbal Bhat (JKHED)

Prof. M. Iqbal Bhat (JKHED)

# Example of pass statement:

- If you have written some code but it's incomplete or you haven't decided how to implement it yet, you can use pass statement to indicate that the code block is empty.

```
if condition1:  
    # TODO: Add implementation later  
    pass  
elif condition2:  
    # TODO: Add implementation later  
    pass  
else:  
    # TODO: Add implementation later  
    pass
```

Questions?

Prof. M. Iqbal Bhat (JKHEP)

