**IF :**  it is a decision making , check if condition is true

If it is matches (true) , else means false

Syntax : If condition :

Statements

**if** num == 5 :  
 print(**"Number is 5"**)

**IF Else :**

The if..else statement evaluates test expression and will execute the body of if only when the test condition is True.

If the condition is False, the body of else is executed. Indentation is used to separate the blocks.

# Program checks if the number is positive or negative# And displays an appropriate message

num = 3

# Try these two variations as well. # num = -5# num = 0

if num >= 0:

print("Positive or Zero")

else:

print("Negative number")

**IF elif else:**

The elif is short for else if. It allows us to check for multiple expressions.

If the condition for if is False, it checks the condition of the next elif block and so on.

If all the conditions are False, the body of else is executed.

Only one block among the several if...elif...else blocks is executed according to the condition.

The if block can have only one else block. But it can have multiple elif blocks.

'''In this program,

we check if the number is positive or

negative or zero and

display an appropriate message'''

num = 3.4

# Try these two variations as well:# num = 0# num = -4.5

if num > 0:

print("Positive number")

elif num == 0:

print("Zero")

else:

print("Negative number")

**For loop :**

The for loop in Python is used to iterate over a sequence ([list](https://www.programiz.com/python-programming/list), [tuple](https://www.programiz.com/python-programming/tuple), [string](https://www.programiz.com/python-programming/string)) or other iterable objects. Iterating over a sequence is called traversal.

### **Syntax of for Loop**

for val in sequence:

loop body

empdetails = [10,**"Chandra"** ,**"AP"**,**"INDIA"**]

for emp in empdetails:  
 print(emp)

# Program to find the sum of all numbers stored in a list

# List of numbers

numbers = [6, 5, 3, 8, 4, 2, 5, 4, 11]

# variable to store the sum

sum = 0

# iterate over the list

for val in numbers:

sum = sum+val

print("The sum is", sum)

When you run the program, the output will be:

The sum is 48

**Range() :**

We can generate a sequence of numbers using range() function. range(10) will generate numbers from 0 to 9 (10 numbers).

We can also define the start, stop and step size as range(start, stop,step\_size). step\_size defaults to 1 if not provided.

The following example will clarify this.

print(range(10))

print(list(range(10)))

print(list(range(2, 8)))

print(list(range(2, 20, 3)))

****Output****

range(0, 10)

[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

[2, 3, 4, 5, 6, 7]

[2, 5, 8, 11, 14, 17]

We can use the range() function in for loops to iterate through a sequence of numbers

numbers = range(10)  
print(numbers)  
range(startposition, endposition,stepsize)  
**for** num **in** range(4,20,2):  
 print(num)

# Program to iterate through a list using indexing

genre = ['pop', 'rock', 'jazz']

# iterate over the list using indexfor i in range(len(genre)):

print("I like", genre[i])

****Output****

I like pop

I like rock

​I like jazz

numbers = [7,769,67]  
sum = 0  
**for** val **in** numbers:  
 print(**"Sum value"** + str(sum))  
 sum = sum+val  
 print(**"printing sum on each iteration"** +str(sum))

**While loop:**

The while loop in Python is used to iterate over a block of code as long as the test expression (condition) is true.

We generally use this loop when we don't know the number of times to iterate beforehand.

### **Syntax of while Loop in Python**

while test\_expression:

Body of while

# Program to add natural# numbers up to # sum = 1+2+3+...+n

# To take input from the user,# n = int(input("Enter n: "))

n = 10

# initialize sum and counter

sum = 0

i = 1

while i <= n:

sum = sum + i

i = i+1 # update counter

# print the sumprint("The sum is", sum)

When you run the program, the output will be:

Enter n: 10

The sum is 55

**Break and continue :**

In Python, break and continue statements can alter the flow of a normal loop.

Loops iterate over a block of code until the test expression is false, but sometimes we wish to terminate the current iteration or even the whole loop without checking test expression.

The break and continue statements are used in these cases.

## Python break statement

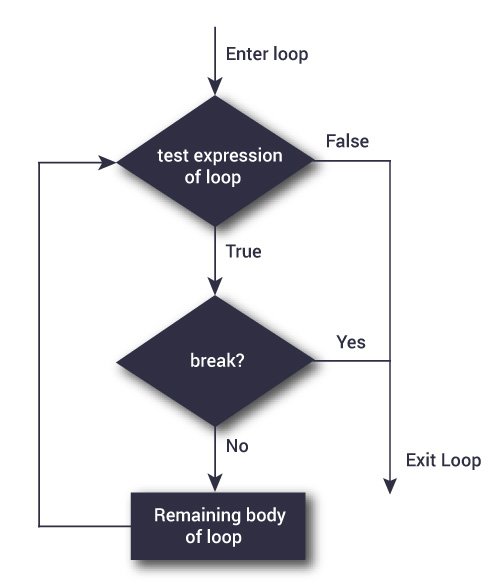
The break statement terminates the loop containing it. Control of the program flows to the statement immediately after the body of the loop.

If the break statement is inside a nested loop (loop inside another loop), the break statement will terminate the innermost loop.

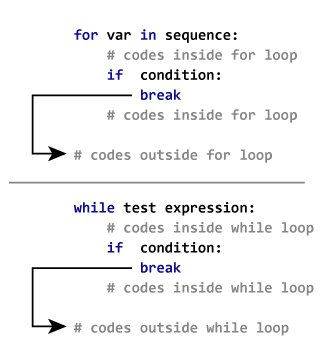
### **Syntax of break**

break

### **Flowchart of break**

Flowchart of break statement in Python

The working of break statement in [for loop](https://www.programiz.com/python-programming/for-loop) and [while loop](https://www.programiz.com/python-programming/while-loop) is shown below.

Working of the break statement

### **Example: Python break**

# Use of break statement inside the loop

for val in "string":

if val == "i":

break

print(val)

print("The end")

****Output****

s

t

r

The end

In this program, we iterate through the "string" sequence. We check if the letter is i, upon which we break from the loop. Hence, we see in our output that all the letters up till i gets printed. After that, the loop terminates.

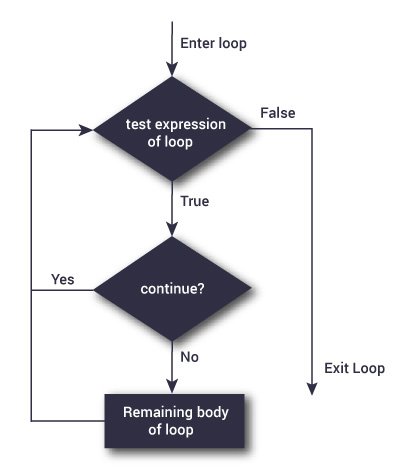
## Python continue statement

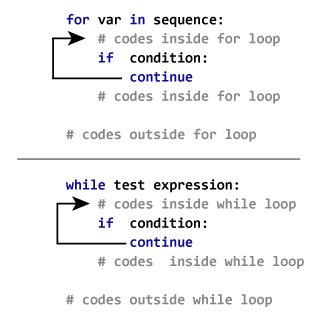
The continue statement is used to skip the rest of the code inside a loop for the current iteration only. Loop does not terminate but continues on with the next iteration.

### **Syntax of Continue**

continue

### **Flowchart of continue**

Flowchart of continue statement in Python

The working of the continue statement in for and while loop is shown below.How continue statement works in python

### **Example: Python continue**

# Program to show the use of continue statement inside loops

for val in "string":

if val == "i":

continue

print(val)

print("The end")

****Output****

s

t

r

n

g

The end

This program is same as the above example except the break statement has been replaced with continue.

We continue with the loop, if the string is i, not executing the rest of the block. Hence, we see in our output that all the letters except i gets printed.

*# break ,continue*names = [**"Chandra"**,**"Ravi"**,**"Prem"**,**"Karthik"**]  
**for** name **in** names:  
 **if**(name == **"Chandra"**) :  
 *# print(name)* **continue** print(name)  
 print(**"Address"**)  
 print(name)  
  
*# pass***for** name **in** names :  
 **pass**