## **Conditional statements**

In programming, very often we want to check the conditions and change the behavior of the program.

How to use Conditional Statements.

We can write programs that has more than one choice of actions depending on a variable's value.   
  
Perhaps the most well-known statement type is the if statement.   
  
You use the if statement to perform one action if one thing is true,or any number of other actions, if something else is true.   
  
We must use indentation to define that code that is executed, based on whether a condition is met.

**If Statement**

The syntax of the if statement is:  
  
if expression:  
 statement(s)

**Elif Statement**

Sometimes there are more than two possibilities, in that case we can use the elif statement  
  
It stands for "else if," which means that if the original if statement is false and the elif statement is true, execute the block of code following the elif statement.

The syntax of the if...elif statement is:

if expression1:  
 statement(s)  
elif expression2:  
 statement(s)  
elif expression3:  
 statement(s)  
else:  
 statement(s)

#### **Else Statement**

An else statement can be combined with an if statement.   
  
An else statement contains the block of code that executes if the conditional expression in the if statement resolves to 0 or a false value.   
  
The else statement is an optional statement and there could be at most only one else statement following if.

The syntax of if..else is:  
if expression:  
 statement(s)  
else:  
 statement(s)

#### Examples

This script will compare two strings based on the input from the use

# This program compares two strings.  
# Get a password from the user.  
password = raw\_input('Enter the password: ')  
# Determine whether the correct password  
# was entered.  
  
if password == 'hello':  
 print'Password Accepted'  
else:  
 print 'Sorry, that is the wrong password.'

**Another Example**

Let's show one more examples, in which will also make use of the elif statement.  
#!/usr/bin/python  
number = 20  
guess = int(input('Enter an integer : '))  
if guess == number:  
 print('Congratulations, you guessed it.')  
elif guess < number:  
 print('No, it is a little higher than that')

else:

print('No, it is a little lower than that')

**What is for loop in Python?**

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:  
 Body of for

Here, val is the variable that takes the value of the item inside the sequence on each iteration.

Loop continues until we reach the last item in the sequence. The body of for loop is separated from the rest of the code using indentation.

### **Example: Python for Loop (Try and write output)**Flowchart of for Loop in Python programming

# 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

# Output: The sum is 48

print("The sum is", sum)

## **The range() function**

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.

This function does not store all the values in memory, it would be inefficient. So it remembers the start, stop, step size and generates the next number on the go.

To force this function to output all the items, we can use the function list().

The following example will clarify this.

# Output: range(0, 10)  
print(range(10))  
  
# Output: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]  
print(list(range(10)))  
  
# Output: [2, 3, 4, 5, 6, 7]  
print(list(range(2, 8)))  
  
# Output: [2, 5, 8, 11, 14, 17]  
print(list(range(2, 20, 3)))

We can use the range() function in for loops to iterate through a sequence of numbers. It can be combined with the len() function to iterate through a sequence using indexing. Here is an example.

# Program to iterate through a list using indexing  **(Try and write output)**

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

# iterate over the list using index

for i in range(len(genre)):

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

**for loop with else**

A for loop can have an optional else block as well. The else part is executed if the items in the sequence used in for loop exhausts.

break statement can be used to stop a for loop. In such case, the else part is ignored.

Hence, a for loop's else part runs if no break occurs.

Here is an example to illustrate this.

digits = [0, 1, 5] **(Try and write output)**

for i in digits:

print(i)

else:

print("No items left.")

## **What is while loop in Python?**

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 beforehand, the number of times to iterate.

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

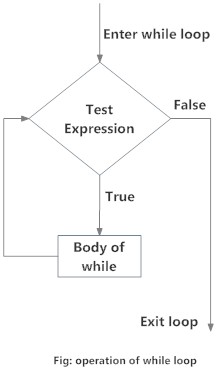
while test\_expression:  
 Body of while

In while loop, test expression is checked first. The body of the loop is entered only if the test\_expression evaluates to True. After one iteration, the test expression is checked again. This process continues until the test\_expression evaluates to False.

In Python, the body of the while loop is determined through indentation.

Body starts with indentation and the first unindented line marks the end.

Python interprets any non-zero value as True. None and 0 are interpreted as False.



### **Example: Python while Loop (Try and write output)**

# Program to add natural

# numbers upto

# 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 sum

print("The sum is", sum)

In the above program, the test expression will be True as long as our counter variable i is less than or equal to n (10 in our program).

We need to increase the value of counter variable in the body of the loop. This is very important (and mostly forgotten). Failing to do so will result in an infinite loop (never ending loop).

Finally the result is displayed.

**While loop with else**

Same as that of [for loop](https://www.programiz.com/python-programming/for-loop), we can have an optional else block with while loop as well.

The else part is executed if the condition in the while loop evaluates to False.

The while loop can be terminated with a [break statement](https://www.programiz.com/python-programming/break-continue). In such case, the else part is ignored. Hence, a while loop's else part runs if no break occurs and the condition is false.

Here is an example to illustrate this. **(Try and write output)**

# Example to illustrate

# the use of else statement

# with the while loop

counter = 0

while counter < 3:

print("Inside loop")

counter = counter + 1

else:

print("Inside else")

Here, we use a counter variable to print the string Inside loop three times.

On the fourth iteration, the condition in while becomes False. Hence, the else part is executed.

## **What is the use of break and continue in Python?**

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

Loops iterate over a block of code until 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**Flowchart of break statement in Python

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

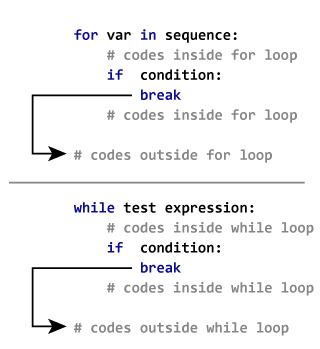
If break statement is inside a nested loop (loop inside another loop), break will terminate the innermost loop.

### **Syntax of break**

break

### **Flowchart of break**

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.



### **Example: Python break**

# Use of break statement inside loop

for val in "string":

if val == "i":

break

print(val)

print("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**How continue statement works in python

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.

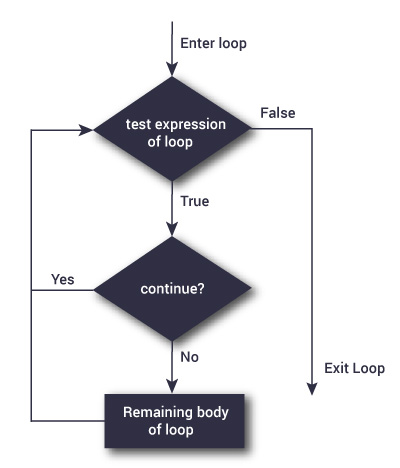
### **Syntax of Continue**

continue

The working of continue statement in for and while loop is shown below.

### **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")

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.

## 

## 

## 

## 

## **What is pass statement in Python?**

In Python programming, pass is a null statement. The difference between a [comment](https://www.programiz.com/python-programming/statement-indentation-comments) and pass statement in Python is that, while the interpreter ignores a comment entirely, pass is not ignored.

However, nothing happens when pass is executed. It results into no operation (NOP).

### **Syntax of pass**

pass

We generally use it as a placeholder.

Suppose we have a [loop](https://www.programiz.com/python-programming/for-loop) or a [function](https://www.programiz.com/python-programming/function) that is not implemented yet, but we want to implement it in the future. They cannot have an empty body. The interpreter would complain. So, we use the pass statement to construct a body that does nothing.

### **Example: pass Statement**

# pass is just a placeholder for

# functionality to be added later.

sequence = {'p', 'a', 's', 's'}

for val in sequence:

pass

We can do the same thing in an empty function or [class](https://www.programiz.com/python-programming/class) as well.

def function(args):  
 pass

class example:  
 pass