**Python IF, ELSE, ELIF, Nested IF & Switch Case Statement**

### What are Conditional Statements?

Conditional Statement in Python perform different computations or actions depending on whether a specific Boolean constraint evaluates to true or false. Conditional statements are handled by IF statements in Python.

## What is If Statement? How to Use it?

In Python, If Statement is used for decision making. It will run the body of code only when IF statement is true.

When you want to justify one condition while the other condition is not true, then you use "if statement".

Syntax:

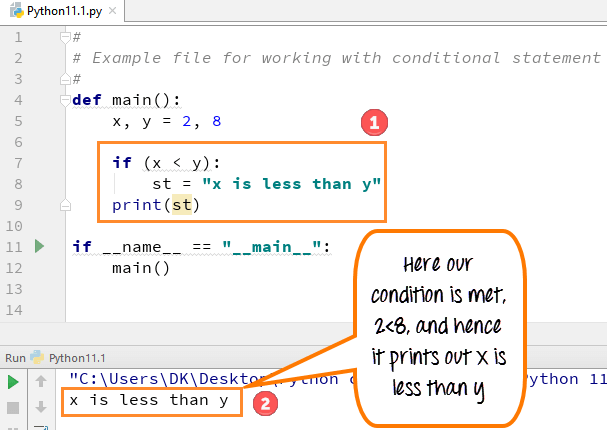
if expression

Statement

else

Statement

Let see an **example**-

[](https://www.guru99.com/images/Pythonnew/Python11.1.png)

#

#Example file for working with conditional statement

#

def main():

x,y =2,8

if(x < y):

st= "x is less than y"

print(st)

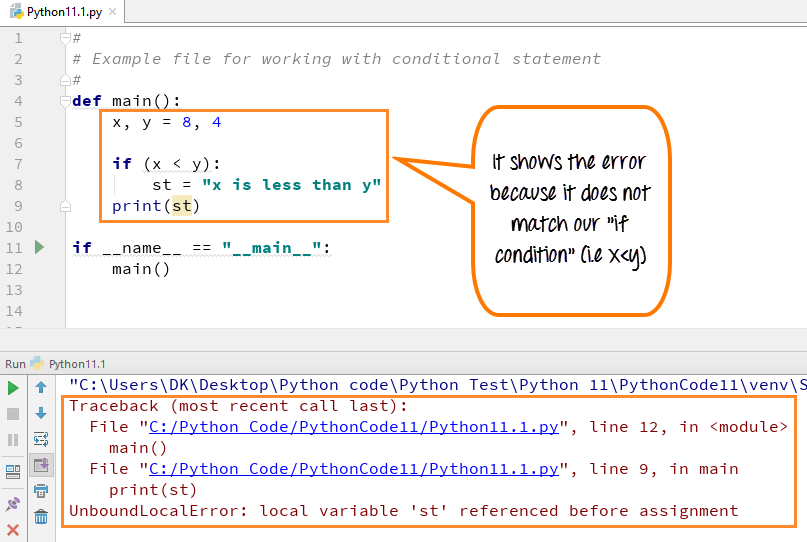
if \_\_name\_\_ == "\_\_main\_\_":

main()

* Code Line 5: We define two variables x, y = 2, 8
* Code Line 7: The if Statement checks for condition x<y which is **True** in this case
* Code Line 8: The variable st is set to "x is less than y."
* Code Line 9: The line print st will output the value of variable st which is "x is less than y",

## What happen when "if condition" does not meet

In this step, we will see what happens when your "if condition" does not meet.

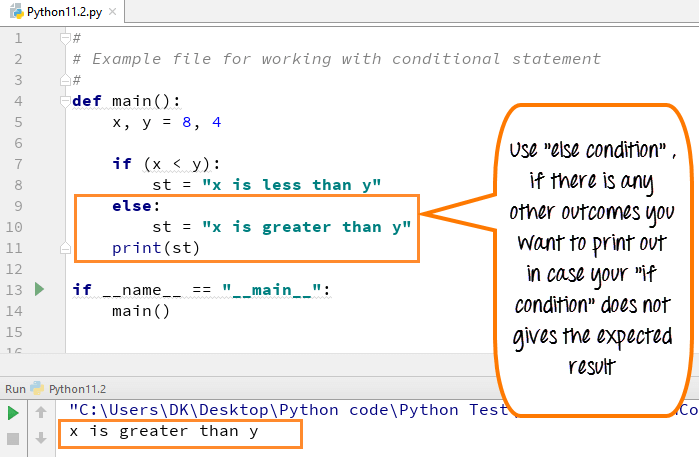
[](https://www.guru99.com/images/Pythonnew/Python11.2.png)

* Code Line 5: We define two variables x, y = 8, 4
* Code Line 7: The if Statement checks for condition x<y which is **False** in this case
* Code Line 8: The variable st is **NOT** set to "x is less than y."
* Code Line 9: The line print st - is trying to print the value of a variable that was never declared. Hence, we get an error.

## How to use "else condition"

The "else condition" is usually used when you have to judge one statement on the basis of other. If one condition goes wrong, then there should be another condition that should justify the statement or logic.

**Example**:

[](https://www.guru99.com/images/Pythonnew/Python11.3.png)

#

#Example file for working with conditional statement

#

def main():

x,y =8,4

if(x < y):

st= "x is less than y"

else:

st= "x is greater than y"

print (st)

if \_\_name\_\_ == "\_\_main\_\_":

main()

* Code Line 5: We define two variables x, y = 8, 4
* Code Line 7: The if Statement checks for condition x<y which is **False** in this case
* Code Line 9: The flow of program control goes to else condition
* Code Line 10: The variable st is set to "x is **greater** than y."
* Code Line 11: The line print st will output the value of variable st which is "x is greater than y",

## When "else condition" does not work

There might be many instances when your "else condition" won't give you the desired result. It will print out the wrong result as there is a mistake in program logic. In most cases, this happens when you have to justify more than two statement or condition in a program.

An **example** will better help you to understand this concept.

Here both the variables are same (8,8) and the program output is **"x is greater than y",** which is **WRONG**. This is because it checks the first condition (if condition), and if it fails, then it prints out the second condition (else condition) as default. In next step, we will see how we can correct this error.

A switch statement is a multiway branch statement that compares the value of a variable to the values specified in case statements.

Python language doesn’t have a switch statement.

Python uses dictionary mapping to implement switch statement in Python

**Example**

function(argument){

switch(argument) {

case 0:

return "This is Case Zero";

case 1:

return " This is Case One";

case 2:

return " This is Case Two ";

default:

return "nothing";

};

};

For the above switch statement Alternative in Python

def SwitchExample(argument):

switcher = {

0: " This is Case Zero ",

1: " This is Case One ",

2: " This is Case Two ",

}

return switcher.get(argument, "nothing")

if \_\_name\_\_ == "\_\_main\_\_":

argument = 1

print (SwitchExample(argument))

**Python 2 Example**

Above codes are Python 3 examples, If you want to run in Python 2 please consider following code.

# If Statement

#Example file for working with conditional statement

#

def main():

x,y =2,8

if(x < y):

st= "x is less than y"

print st

if \_\_name\_\_ == "\_\_main\_\_":

main()

# How to use "else condition"

#Example file for working with conditional statement

#

def main():

x,y =8,4

if(x < y):

st= "x is less than y"

else:

st= "x is greater than y"

print st

if \_\_name\_\_ == "\_\_main\_\_":

main()

# When "else condition" does not work

#Example file for working with conditional statement

#

def main():

x,y =8,8

if(x < y):

st= "x is less than y"

else:

st= "x is greater than y"

print st

if \_\_name\_\_ == "\_\_main\_\_":

main()

# How to use "elif" condition

#Example file for working with conditional statement

#

def main():

x,y =8,8

if(x < y):

st= "x is less than y"

elif (x == y):

st= "x is same as y"

else:

st="x is greater than y"

print st

if \_\_name\_\_ == "\_\_main\_\_":

main()

# How to execute conditional statement with minimal code

def main():

x,y = 10,8

st = "x is less than y" if (x < y) else "x is greater than or equal to y"

print st

if \_\_name\_\_ == "\_\_main\_\_":

main()

# Nested IF Statement

total = 100

#country = "US"

country = "AU"

if country == "US":

if total <= 50:

print "Shipping Cost is $50"

elif total <= 100:

print "Shipping Cost is $25"

elif total <= 150:

print "Shipping Costs $5"

else:

print "FREE"

if country == "AU":

if total <= 50:

print "Shipping Cost is $100"

else:

print "FREE"

#Switch Statement

def SwitchExample(argument):

switcher = {

0: " This is Case Zero ",

1: " This is Case One ",

2: " This is Case Two ",

}

return switcher.get(argument, "nothing")

if \_\_name\_\_ == "\_\_main\_\_":

argument = 1

print SwitchExample(argument)

### Summary:

A conditional statement in Python is handled by if statements and we saw various other ways we can use conditional statements like if and else over here.

* "if condition" – It is used when you need to print out the result when one of the conditions is true or false.
* "else condition"- it is used when you want to print out the statement when your one condition fails to meet the requirement
* "elif condition" – It is used when you have third possibility as the outcome. You can use multiple elif conditions to check for 4th,5th,6th possibilities in your code
* We can use minimal code to execute conditional statements by declaring all condition in single statement to run the code
* If Statement can be nested

### What is Loop?

Loops can execute a block of code number of times until a certain condition is met. Their usage is fairly common in programming. Unlike other programming language that have For Loop, while loop, dowhile, etc.

### What is For Loop?

For loop is used to iterate over elements of a sequence. It is often used when you have a piece of code which you want to repeat "n" number of time.

### What is While Loop?

While Loop is used to repeat a block of code. Instead of running the code block once, It executes the code block multiple times until a certain condition is met.

## How to use "While Loop"

While loop does the exactly same thing what "if statement" does, but instead of running the code block once, they jump back to the point where it began the code and repeats the whole process again.

Syntax

while expression

Statement

**Example**:

#

#Example file for working with loops

#

x=0

#define a while loop

while(x <4):

print(x)

x = x+1

Output

0

1

2

3

* Code Line 4: Variable x is set to 0
* Code Line 7: While loop checks for condition x<4. The current value of x is 0. Condition is true. Flow of control enters into while Loop
* Code Line 8: Value of x is printed
* Code Line 9: x is incremented by 1. Flow of control goes back to line 7. Now the value of x is 1 which is less than 4. The condition is true, and again the while loop is executed. This continues till x becomes 4, and the while condition becomes false.

## How to use "For Loop"

In Python, "for loops" are called **iterators.**

Just like while loop, "For Loop" is also used to repeat the program.

But unlike while loop which depends on condition true or false. "For Loop" depends on the elements it has to iterate.

**Example**:

#

#Example file for working with loops

#

x=0

#define a while loop

# while(x <4):

# print x

# x = x+1

#Define a for loop

for x in range(2,7):

print(x)

Output

2

3

4

5

6

For Loop iterates with number declared in the range.

For example,

**For Loop** for x in range (2,7)

When this code is executed, it will print the number between 2 and 7 (2,3,4,5,6). In this code, number 7 is not considered inside the range.

For Loops can also be used for a set of other things and not just number. We will see thin in next section.

## How to use For Loop for String

In this step, we will see how "for loops" can also be used for other things besides numbers.

**Example**:

#use a for loop over a collection

Months = ["Jan","Feb","Mar","April","May","June"]

for m in Months:

print(m)

Output

Jan

Feb

Mar

April

May

June

Code Line 3: We store the months ("Jan, Feb , Mar,April,May,June") in variable Months

Code Line 4: We iterate the for loop over each value in Months. The current value of Months in stored in variable m

Code Line 5: Print the month

## How to use break statements in For Loop

Breakpoint is a unique function in For Loop that allows you to break or terminate the execution of the for loop

**Example**:

#use a for loop over a collection

#Months = ["Jan","Feb","Mar","April","May","June"]

#for m in Months:

#print m

# use the break and continue statements

for x in range (10,20):

if (x == 15): break

#if (x % 2 == 0) : continue

print(x)

Output

10

11

12

13

14

In this example, we declared the numbers from 10-20, but we want that our for loop to terminate at number 15 and stop executing further. For that, we declare break function by defining (x==15): break, so as soon as the code calls the number 15 it terminates the program Code Line 10 declare variable x between range (10, 20)

* Code Line 11 declare the condition for breakpoint at x==15,
* Code Line 12 checks and repeats the steps until it reaches number 15
* Code Line 13 Print the result in output

## How to use "continue statement" in For Loop

Continue function, as the name indicates, will terminate the current iteration of the for loop BUT will continue execution of the remaining iterations.

**Example**

#use a for loop over a collection

#Months = ["Jan","Feb","Mar","April","May","June"]

#for m in Months:

#print m

# use the break and continue statements

for x in range (10,20):

#if (x == 15): break

if (x % 5 == 0) : continue

print(x)

Output

11

12

13

14

16

17

18

19

Continue statement can be used in for loop when you want to fetch a specific value from the list.

In our example, we have declared value 10-20, but between these numbers we only want those number that are NOT divisible by 5 or in other words which don't give zero when divided by 5.

So, in our range (10,11, 12….19,20) only 3 numbers falls (10,15,20) that are divisible by 5 and rest are not.

So except number 10,15 & 20 the "for loop" will not continue and print out those number as output.

* Code line 10 declare the variable x for range (10, 20)
* Code line 12 declare the condition for x divided by 5=0 continue
* Code line 13 print the result

## How to use "enumerate" function for "For Loop"

Enumerate function in "for loop" does two things

* It returns the index number for the member
* And the member of the collection that we are looking at

**Example**:

Enumerate function is used for the numbering or indexing the members in the list.

Suppose, we want to do numbering for our month ( Jan, Feb, Marc, ….June), so we declare the variable i that enumerate the numbers while m will print the number of month in list.

#use a for loop over a collection

Months = ["Jan","Feb","Mar","April","May","June"]

for i, m in enumerate (Months):

print(i,m)

# use the break and continue statements

#for x in range (10,20):

#if (x == 15): break

#if (x % 5 == 0) : continue

#print x

Output

0 Jan

1 Feb

2 Mar

3 April

4 May

5 June

When code is executed the output of the enumerate function returns the months name with an index number like (0-Jan), (1- Feb), (2- March), etc.

* Code Line 3 declares the list of months [ Jan, Feb,…Jun]
* Code Line 4 declares variable i and m for For Loop
* Code Line 5 will print the result and again enter the For Loop for the rest of the months to enumerate

## Pratical Example

Let see another example for For Loop to repeat the same statement over and again.

|  |  |
| --- | --- |
| **Python loop** | **Working Code for all exercises** |
| Code for while loop | x=0  while (x<4):  print (x)  x= x+1 |
| For Loop Simple Example | x=0  for x in range (2,7):  print (x) |
| Use of for loop in string | Months = ["Jan","Feb","Mar","April","May","June"]  for m in (Months):  print (m) |
| Use break-statement in for loop | for x in range (10,20):  if (x == 15): break  print (x) |
| Use of Continue statement in for loop | for x in range (10,20):  if (x % 5 == 0): continue  print (x) |
| Code for "enumerate function" with "for loop" | Months = ["Jan","Feb","Mar","April","May","June"]  for i, m in enumerate (Months):  print (i,m) |

## How to use for loop to repeat the same statement over and again

You can use for loop for even repeating the same statement over and again. Here in the example we have printed out word "guru99" three times.

Example: To repeat same statement number of times, we have declared the number in variable i (i in 123). So when you run the code as shown below, it prints the statement (guru99) that many times the number declared for our the variable in ( i in 123).

for i in '123':

print ("guru99",i,)

Output

guru99 1

guru99 2

guru99 3

Like other programming languages, Python also uses a loop but instead of using a range of different loops it is restricted to only two loops "While loop" and "for loop".

* While loops are executed based on whether the conditional statement is true or false.
* For loops are called iterators, it iterates the element based on the condition set
* Python For loops can also be used for a set of various other things (specifying the collection of elements we want to loop over)
* Breakpoint is used in For Loop to break or terminate the program at any particular point
* Continue statement will continue to print out the statement, and prints out the result as per the condition set
* Enumerate function in "for loop" returns the member of the collection that we are looking at with the index number

**Python 2 Example**

Above codes are Python 3 examples, If you want to run in Python 2 please consider following code.

# How to use "While Loop"

#Example file for working with loops

#

x=0

#define a while loop

while(x <4):

print x

x = x+1

#How to use "For Loop"

#Example file for working with loops

#

x=0

#define a while loop

# while(x <4):

# print x

# x = x+1

#Define a for loop

for x in range(2,7):

print x

#How to use For Loop for String

#use a for loop over a collection

Months = ["Jan","Feb","Mar","April","May","June"]

for m in Months:

print m

#How to use break statements in For Loop

#use a for loop over a collection

#Months = ["Jan","Feb","Mar","April","May","June"]

#for m in Months:

#print m

# use the break and continue statements

for x in range (10,20):

if (x == 15): break

#if (x % 2 == 0) : continue

print x

#How to use "continue statement" in For Loop

#use a for loop over a collection

#Months = ["Jan","Feb","Mar","April","May","June"]

#for m in Months:

#print m

# use the break and continue statements

for x in range (10,20):

#if (x == 15): break

if (x % 5 == 0) : continue

print x

#How to use "enumerate" function for "For Loop"

#use a for loop over a collection

Months = ["Jan","Feb","Mar","April","May","June"]

for i, m in enumerate (Months):

print i,m

# use the break and continue statements

#for x in range (10,20):

#if (x == 15): break

#if (x % 5 == 0) : continue

#print x

Output

0

1

2

3

2

3

4

5

6

Jan

Feb

Mar

April

May

June

10

11

12

13

14

11

12

13

14

16

17

18

19

0 Jan

1 Feb

2 Mar

3 April

4 May

5 June