Lecture#12 Python Programming

Python Conditional Statements: IF…Else, ELIF & Switch Case

What are Conditional Statements in Python?

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.

In this lecture, we will see how to apply conditional statements in Python.

- What is If Statement? How to Use it?
- What happen when “if condition” does not meet
- How to use “else condition”
- When “else condition” does not work
- How to use “elif” condition
- How to execute conditional statement with minimal code
- Python Nested if Statement
- Switch Case Statement in Python

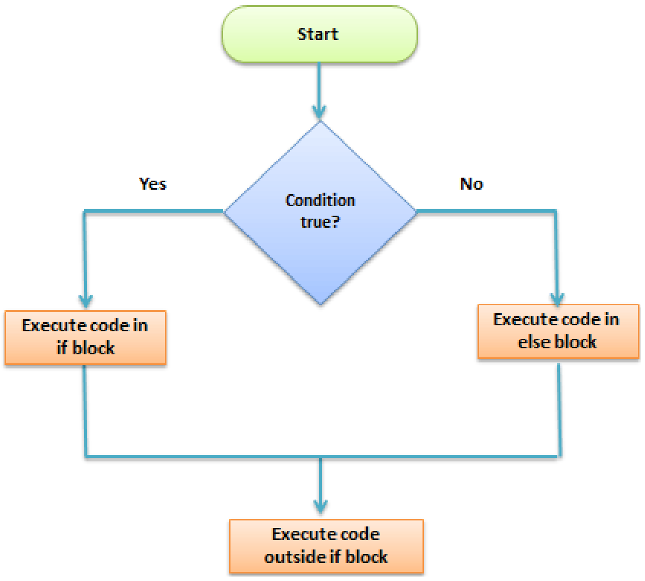
What is Python If Statement?

Python if Statement is used for decision-making operations. It contains a body of code which runs only when the condition given in the if statement is true. If the condition is false, then the optional else statement runs which contains some code for the else condition.
When you want to justify one condition while the other condition is not true, then you use Python if else statement.

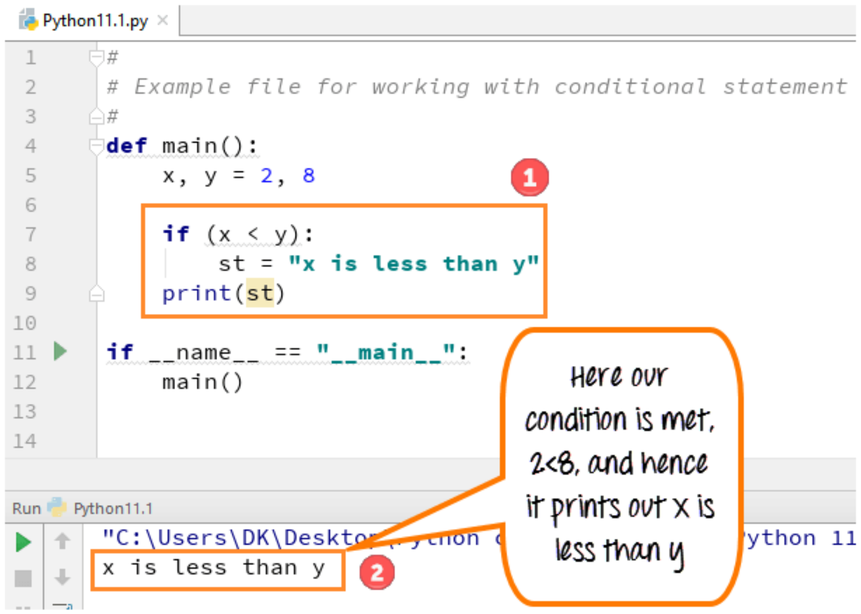
Python if Statement Syntax:

```
if expression
 Statement
else
 Statement
```

Python if…else Flowchart



Let's see an example of Python if else Statement:


Python11.1.py
1
2
Run Python11.1
Here our condition is met, 2<8, and hence it prints out x is less than y

```
#
# 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()
```

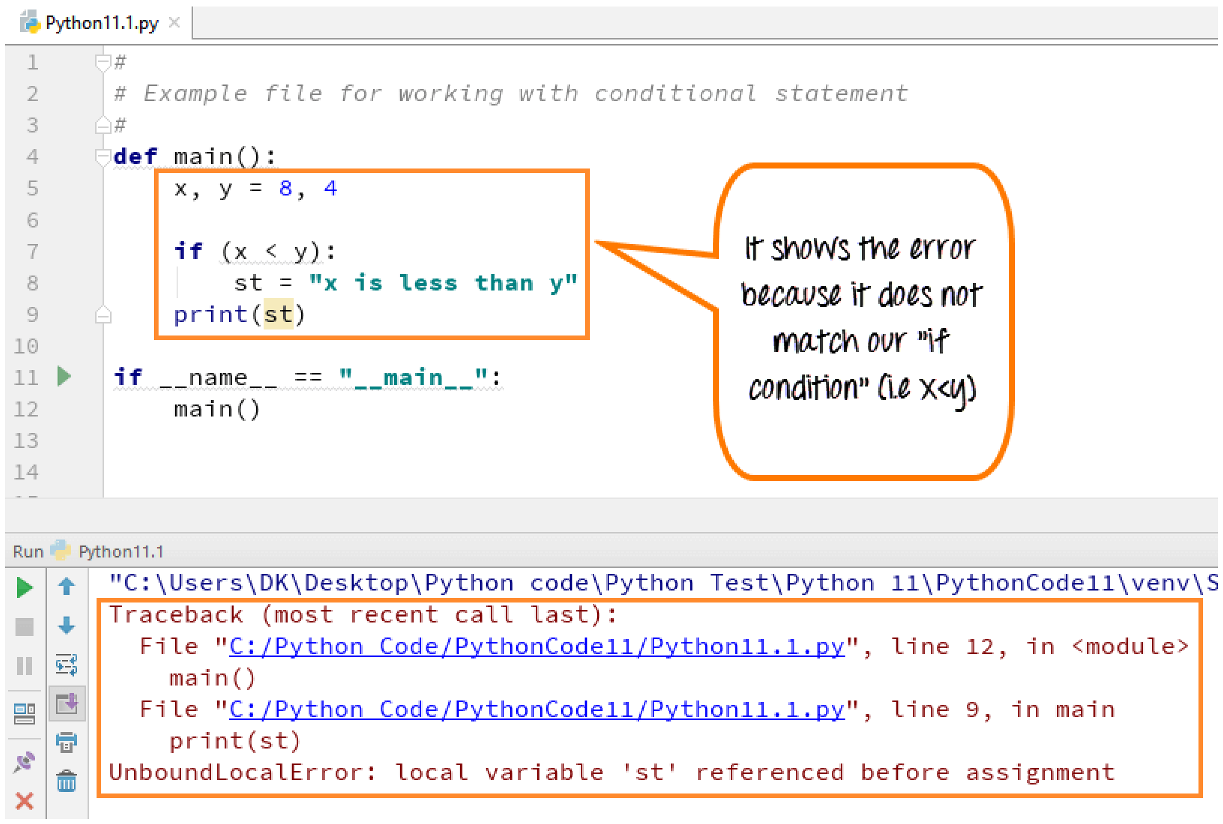
```
"C:\Users\DK\Desktop\Python c... Python 11
x is less than y
```

```
#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 in Python 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 if condition in Python does not meet.


Python11.1.py
It shows the error because it does not match our "if condition" (i.e x<y)
Run Python11.1

```python
#
# Example file for working with conditional statement
#
def main():
    x, y = 8, 4

    if (x < y):
        st = "x is less than y"
    print(st)

if __name__ == "__main__":
    main()
```

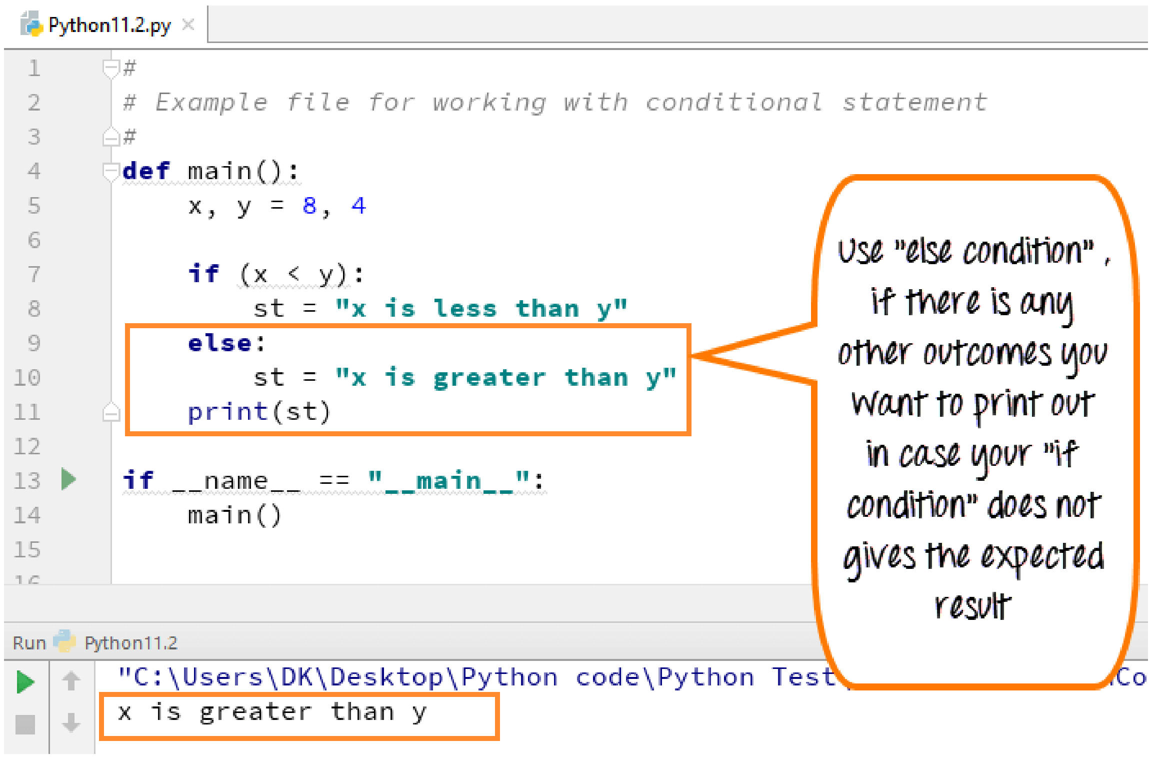
```
"C:\Users\DK\Desktop\Python code\Python Test\Python 11\PythonCode11\venv\S
Traceback (most recent call last):
  File "C:/Python Code/PythonCode11/Python11.1.py", line 12, in <module>
    main()
  File "C:/Python Code/PythonCode11/Python11.1.py", line 9, in main
    print(st)
UnboundLocalError: local variable 'st' referenced before assignment
```

- Code Line 5: We define two variables x, y = 8, 4
- Code Line 7: The if Statement in Python 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:



```
#
# 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()
```

Use "else condition", if there is any other outcomes you want to print out in case your "if condition" does not gives the expected result

```
"C:\Users\DK\Desktop\Python code\Python Test
x is greater than y
```

```
#
#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()
```

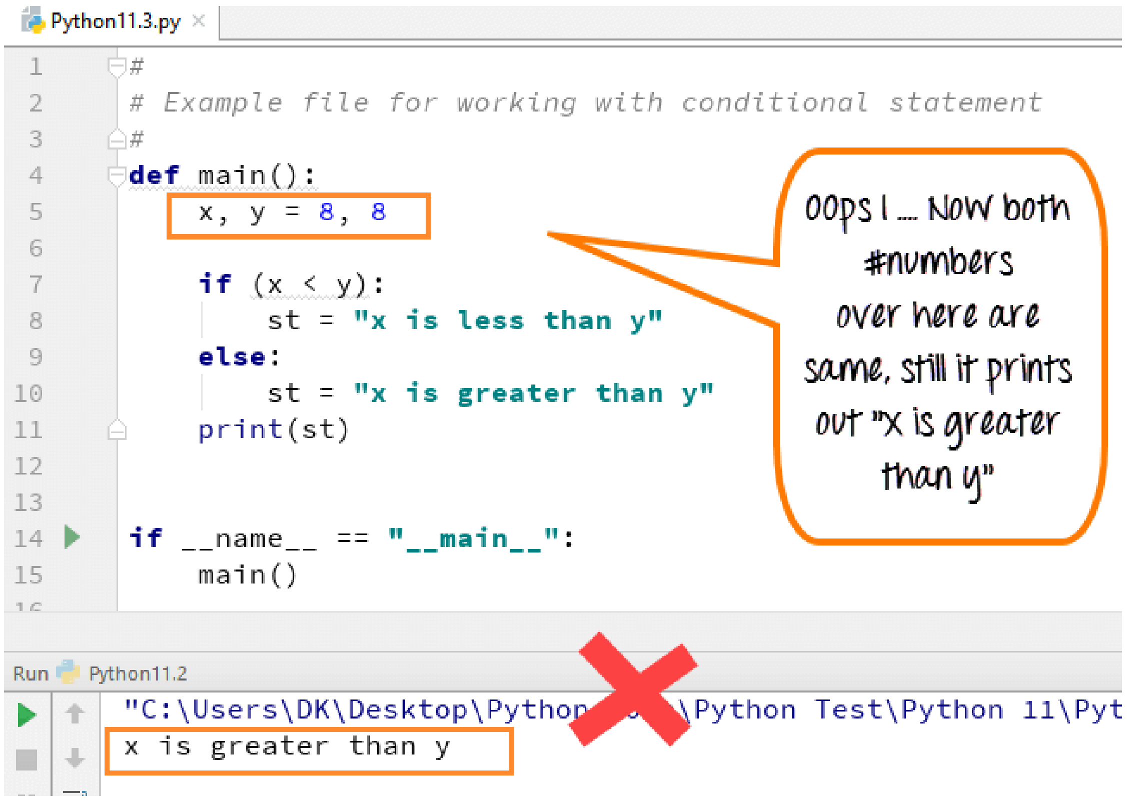
- Code Line 5: We define two variables x, y = 8, 4
- Code Line 7: The if Statement in Python checks for condition x<y which is **False** in this case
- Code Line 9: The flow of program control goes to else condition
- Code Line 10: The variable st is set to “x is **greater** than y.”
- Code Line 11: 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 in Python), 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.



```
#
# 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()
```

Python11.3.py
oops ! ... Now both #numbers over here are same, still it prints out "x is greater than y"
Run Python11.2

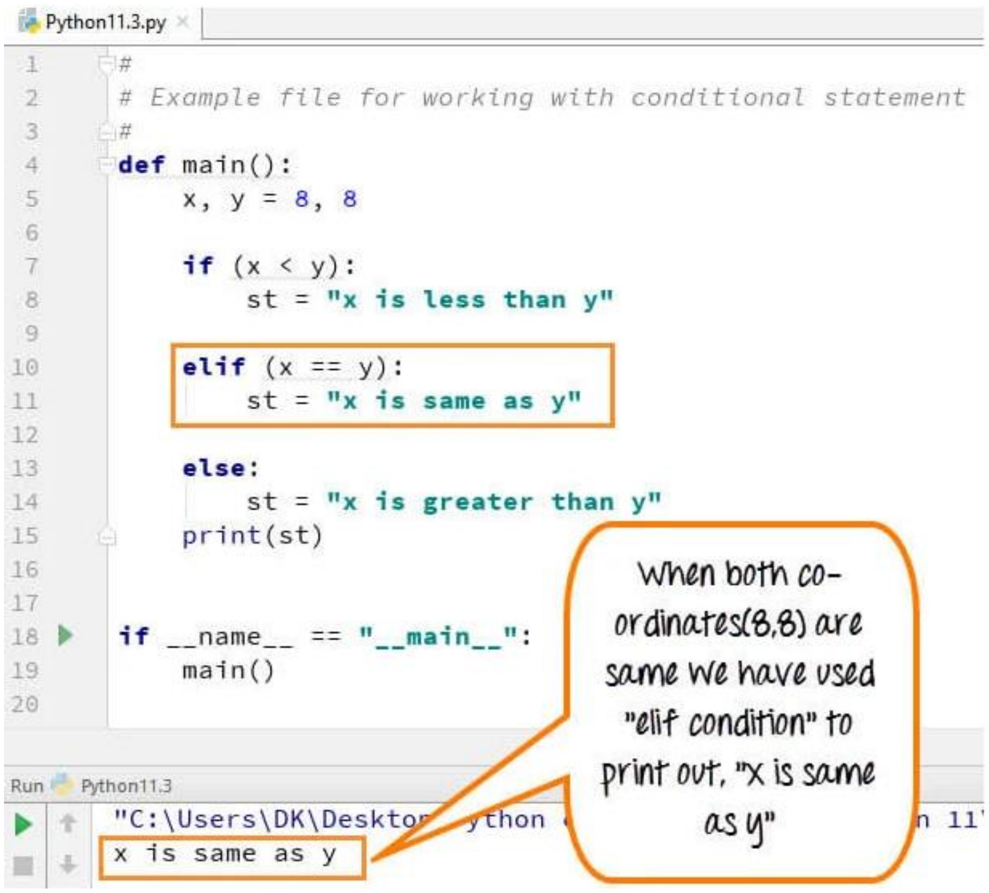
```
"C:\Users\DK\Desktop\Python      \Python Test\Python 11\Pyt
x is greater than y
```

```
#
#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)
```

How to use “elif” condition

To correct the previous error made by “else condition”, we can use **“elif”** statement. By using “**elif**” condition, you are telling the program to print out the third condition or possibility when the other condition goes wrong or incorrect.

Example



```
#
# 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()
```

```
"C:\Users\DK\Desktop ython  n 11'
x is same as y
```

When both co-ordinates(8,8) are same we have used "elif condition" to print out, "x is same as y"

```
#

#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()
```

- Code Line 5: We define two variables x, y = 8, 8
- Code Line 7: The if Statement checks for condition x<y which is **False** in this case
- Code Line 10: The flow of program control goes to the elseif condition. It checks whether x==y which is true
- Code Line 11: The variable st is set to “x is **same as** y.”
- Code Line 15: The **flow of program control exits the if Statement (it will not get to the else Statement).** And print the variable st. The output is “x is same as y” which is correct

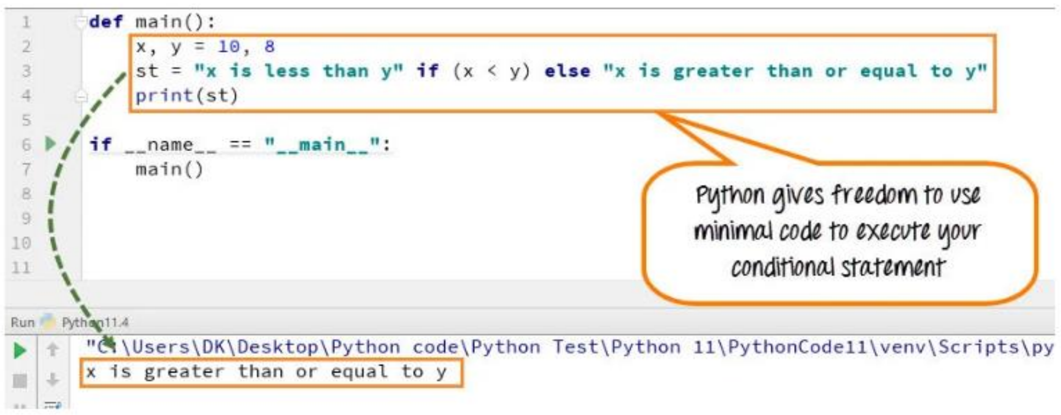
How to execute conditional statement with minimal code

In this step, we will see how we can condense out the conditional statement. Instead of executing code for each condition separately, we can use them with a single code.

Syntax

```
A If B else C
```

Example:



```
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()
```

Python gives freedom to use minimal code to execute your conditional statement
Run Python11.4

```
"C:\Users\DK\Desktop\Python code\Python Test\Python 11\PythonCode11\venv\Scripts\py
x is greater than or equal to y
```

```
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()
```

- Code Line 2: We define two variables x, y = 10, 8
- Code Line 3: Variable st is set to “x is less than y “if x<y or else it is set to “x is greater than or equal to y”. In this x>y variable st is set to **“x is greater than or equal to y.”**
- Code Line 4: Prints the value of st and gives the correct output

- Instead of writing long code for conditional statements, Python gives you the freedom to write code in a short and concise way.

Python Nested if Statement

Following example demonstrates nested if Statement Python

```
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")
```

Uncomment Line 2 in above code and comment Line 3 and run the code again

Switch Case Statement in Python

What is Switch statement?

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 Case 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 case 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))
```

Summary:

A conditional statement in Python is handled by if statements and we saw various other ways we can use conditional statements like Python if 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
- Python If Statement can be nested