Python - Decision Control: if, else, elif

By default, statements in the script are executed sequentially from the first to the last. If the processing logic requires so, the sequential flow can be altered in two ways:

- Conditional execution: a block of one or more statements will be executed if a certain expression is true.
- Repetitive execution: a block of one or more statements will be repetitively executed as long as a certain expression is true.

In this chapter you will learn the decision control if, else, elif statements in Python. The comparison and logical operators are useful in controlling flow of the program.

if Condition

Python uses the if keyword to implement decision control. Python's syntax for executing a block conditionally is as below:

Syntax:

```
if [boolean expression]:
   statement1
   statement2
   ...
   statementN
```

Any Boolean expression evaluating to True or False appears after the if keyword. Use the : symbol and press Enter after the expression to start a block with increased indent. One or more statements written with the same level of indent will be executed if the Boolean expression evaluates to True.

To end the block, decrease the indentation. Subsequent statements after the block will be executed out of the if condition. The following example demonstrates the if condition.

```
>>>> if 10<100:
...      print("10 is less than 100")
```

In the above example, the expression 10<100 evalutes to True, so it will execute the block. The if block starts from new line after :. All the statements under the if condition start with an increased indentation. Above, if block contains only one statement.

Output
10 is less than 100

Now, let's take another example. Assume that we have to write a Python program that calculates the amount payable from price and quantity inputs by the user and applies a 10% discount if the amount exceeds 1000.

Calculation and application of the discount is to be done only if the amount is greater than 1000, hence, the process is placed in a block with increased indent, following the conditional expression.

The print() statement is written after the conditional block is over, hence, it will be executed if the expression is false (i.e. the amount is not greater than 1000), as well as after applying the discount if the expression is true (i.e. the amount is greater than 1000).

```
price=int(input("Enter Price: "))
qty=int(input("Enter Quantity: "))
amt=price*qty
if amt>1000:
    print ("10% discount is applicable")
    discount=amt*10/100
    amt=amt-discount
print ("Amount payable: ",amt)
```

Two scenarios of running the code are shown below. In the first one, inputs for price and quantity are such that the amount is greater than 1000. The payable amount after applying the discount will be displayed.

Output
Enter Price: 100
Enter Quantity: 20
10% discount is applicable
Amount payable: 1800.0

In the second case, inputs are such that the expression doesn't become true. Hence, the print statement will display the payable amount, which will not have a discount.

Output
Enter Price: 10
Enter Quantity: 20
Amount payable: 200

else Condition

Along with the if statement, the else condition can be optionally used to define an alternate block of statements to be executed if the boolean expression in the if condition is not true.

Syntax:

```
if [boolean expression]:
   statement1
   statement2
   ...
   statementN
else:
   statement1
   statement2
   ...
   statementN
```

As mentioned before, the indented block starts after the : symbol, after the boolean expression. It will get executed when the condition is true. We have another block that should be executed when the if condition is false. First, complete the if block by backspace and write else, put add the : symbol in front of the new block to begin it and add the required statements in the block. De-dent the else block and enter the statements that should be executed, irrespective of the boolean expression being true or false.

```
>>>> if 10>100:
...     print("10 is greater than 100")
... else:
...     print("10 is less than 100")
```

Output

10 is less than 100

In the above example, the if condition 10>100 is False, so the Python will execute the else block.

elif Condition

Use the elif condition is used to include multiple conditional expressions between if and else.

Syntax:

```
if [boolean expression]:
   [statements]
elif [boolean expresion]:
   [statements]
elif [boolean expresion]:
   [statements]
elif [boolean expresion]:
   [statements]
else:
   [statements]
```

The elif block is executed if the specified condition is true.

```
>>>> x=10
>>>> if x==1:
...     print('X is 1')
... elif x==5:
...     print('X is 5')
... elif x==10:
...     print('X is 10')
... else:
...     print('X is something else')
```

Output

X is 10

In the above example, multiple elif conditions are applied between if and else. Python will execute the elif block whose expression evaluates to true. If multiple elif conditions become true, then the first elif block will be executed.