Subject: Python Programming

Module Number: 1.3

Module Name: Decision Making and Looping in Python

Version Code:PP4, PP5
Released Date: 4-OCT-2019

Introduction to Python

Syllabus:

- Introduction: Introduction to Python, Setting up the environment, Installing Python, Running python program, Python's execution model, Guidelines on how to write good, The Python culture, A note on the IDEs
- Built-in Data Types: Numbers, Immutable sequences, Mutable sequences, Set types,
- **Mapping types – dictionaries, The collections module, Final considerations**
- **Iterating and Making Decisions: Conditional programming, Looping, Putting this all together.**

Decision Making and Looping in Python

Aim:

To discuss about the Decision Making Statements and Looping in Python Programming

Objectives:

The Objectives of this module are to:

To learn about the Decision making statements in Python

To understand the concepts about Looping techniques in Python

Outcomes

At the end of the module, you are expected to :

Students will learn and code using the concept of decision making statements

Students will learn and code using the concept of Looping

Decision Making and Looping in Python

CONTENT :

- Decision Making Statements :
    - If statements
    - If…else statements
    - Nested if statements
- Looping techniques used in Python Programming :
    - While Loop
    - For loop

Decision Making and Looping in Python

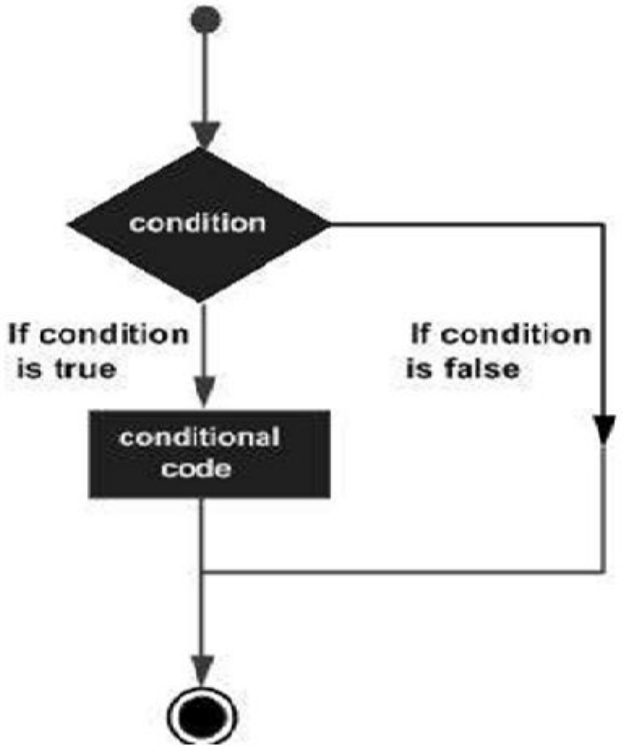
- Decision making is the anticipation of conditions occurring while execution of the program and specifying actions taken according to the conditions.
- Decision structures evaluate multiple expressions which produce TRUE or FALSE as outcomes.
- The action has to determine and the statements to be executed if the outcome is TRUE or FALSE otherwise.
- Types of decision making statements:-
  - If statements
  - If…else statements
  - Nested if statements

Decision Making and Looping in Python

Types of Decision making	Description
If statement	An if statement consists of Boolean expression followed by one or more statement.
If …else statement	An if statement can be followed by optional else statement which executes when the Boolean expression is False.
Nested if else statement	In a nested **if** construct, you can have an **if...elif...else** construct inside another **if...elif...else** construct.

If statements:

The statements under if clause are only executed if and only if the condition prescribed becomes true, else the statements are ignored.

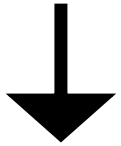


Syntax:-

```
if (condition):
        statement1
        statement2
        …..
        statement n
```

Example of If statements:

```
abc= 10
if (abc == 10) :
    print("hii how are you??")
    print(" hello world")
```



Output:
hii how are you??
hello world

```
a =  7
b =  6
if ( a == b ) :
    print("both are equal")
print ("good bye")
```

Output:-
good bye

If - else statements:

An else statement can be combined with an if statement. The block of code under the if statements are executed if the condition given for if becomes true.

The else statements are optional. There could be at most one else statement followed by the if statement and is executed when the if condition is false.

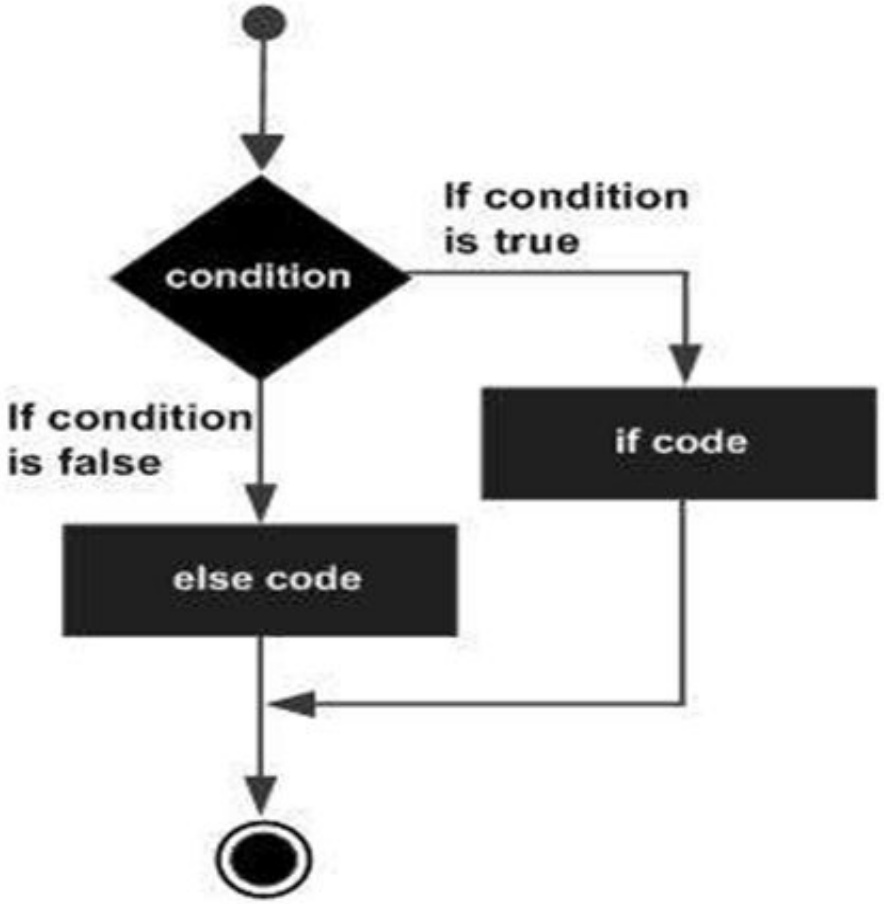
Note:-

The values except 0 are considered as true and 0 is considered as false.

If - else statements:

Syntax:-

```
if ( condition ):
        statement1
        statement2
        …..
        statement n
else:
        statement1
        statement2
        …..
        statement n
```



Example of If - else statements:

```
var=100
if  var:
    print(“it is true")
    print(var)
else:
    print(“it is false")
    print(var)
```

Output:-

```
it is true
 100
```

```
d={"vehicle": "car", "Model":" hatchback"}
key = input("Enter the key to be checked :-")
if key in d.keys():
      print("key value is :")
      print( d[key] )
else:
      print( “ key is not present “ )
```

Output:-

```
Enter the key to be checked :-vehicle
key value is :
car
```

elif statements:

The **elif** statement allows you to check multiple expressions for TRUE and execute a block of code as soon as one of the conditions evaluates to TRUE.

Similar to the **else**, the **elif** statement is optional. However, unlike **else**, for which there can be at most one statement, there can be an arbitrary number of **elif** statements following an **if**.

elif statements:

Syntax:-

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

Example of If - else statements:

```
 var = 100
if var == 200:
        print ("1- Got a true expression value")
elif var == 150:
        print("2- Got a true expression value")
elif var == 100:
        print("3- Got a true expression value")
else:
        print("4-Got a false expression value")
print( "Good bye")
```

Output:-

3-Got a true expression value

Nested if - else statements:

There may be a situation when you want to check for another condition after a condition resolves to true. In such a situation, you can use the nested **if** construct.

In a nested **if** construct, you can have an **if...elif...else** construct inside another **if...elif...else** construct.

Nested if - else statements:

Syntax:-

```
if expression1:
        statement(s)
        if expression2:
            statement(s)
        elif expression3:
            statement(s)
        elif expression4:
            statement(s)
        else:
            statement(s)
 else:
        statement(s)
```

Nested if - else statements (Example):

```
 var = 100
if var < 200:
    print("Expression value is less than 200")
    if var == 200:
        print("It is 200")
    elif var == 100:
        print("it is 100")
    else:
        print("It is not 100")
else:
    print("Could not find true expression")
print("Good bye")
```

Output:-

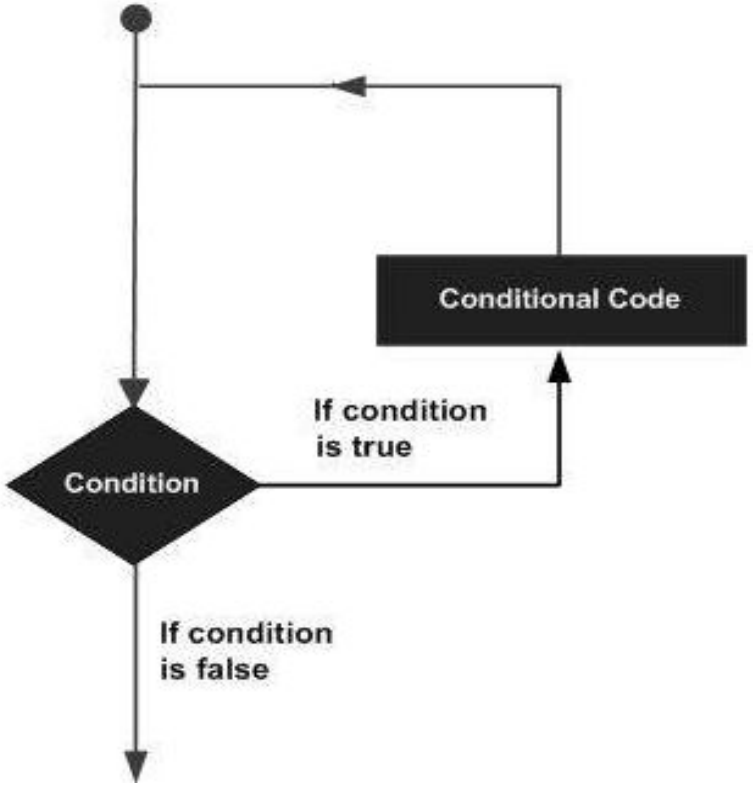
Expression value is less than 200

It is 100

Good bye

Looping in Python

A loop statement allows us to execute a statement or group of statements multiple times. The following diagram illustrates a loop statement.

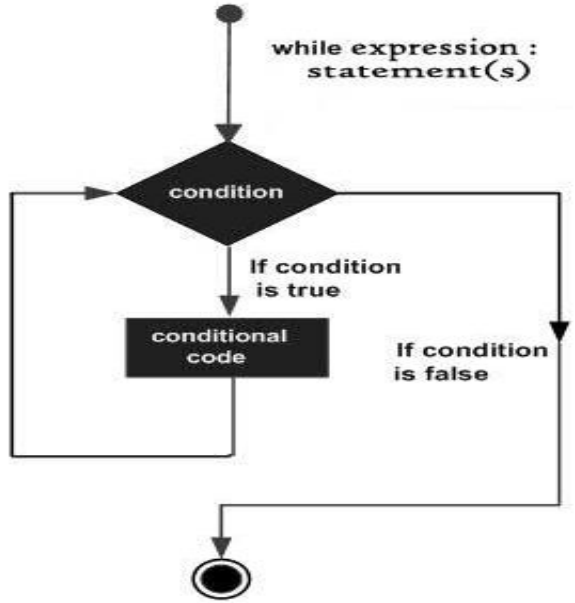


while loop:

- The statements under the while loop are executed until as long as the condition remains true.
- When the condition becomes false, program control passes to the line immediately following the loop.
- The loop statement is mandatory to be incremented or decremented, or else the statements would be executed to infinite times.

Syntax of While loop

```
while expression:
    statement(s)
```



Flow Diagram of While Loop

while loop:

Example of While loop

```
count = 0
while (count < 9):
    print ('The count is:', count)
    count = count + 1
print ("Good bye!“)
```

Out put:

```
The count is: 0
The count is: 1
The count is: 2
The count is: 3
The count is: 4
The count is: 5
The count is: 6
The count is: 7
The count is: 8
Good bye!
```

Else in while loop:

Using else Statement with Loops:- Python supports to have an **else** statement associated with a loop statement. The **else** statement is executed when the condition becomes false.

Example of else with while loop

```
count = 0
while(count < 5):
    print(count, "is less than 5")
    count = count + 1
else:
    print(count, "is not less than 5")
```

Output :

0 is less than 5

1 is less than 5

2 is less than 5

3 is less than 5

4 is less than 5

5 is not less than 5 (when x =5 the controls executes the else statements)

Break Statement :

Break statement: The statements can be stopped from executing even if the condition of the while loop is true by using break statement.

Example:

```
i=1
while(i<6):
    print(i)
    if(i==3):
        break
    i+=1
```

Output:-

```
1
2
3
```

Note: Exit the loop when i is 3.

Continue Statement:

Continue statement: The entire statement of the while loop (or iteration) can be stopped from executing and continuing the next.

Example of Continue statement :

```
i = 0
while i < 6:
        i += 1
  if i == 3:
       continue
  print(i)
```

Output:-

1
2
4
5
6

Continue the next iteration if i is 3

for loop

for loop is used for iterating over a sequence (i.e., either a list, a tuple, a dictionary, a set, or a string).

Break statement:- With the **break** statement we can stop the loop before it has looped through all the items.

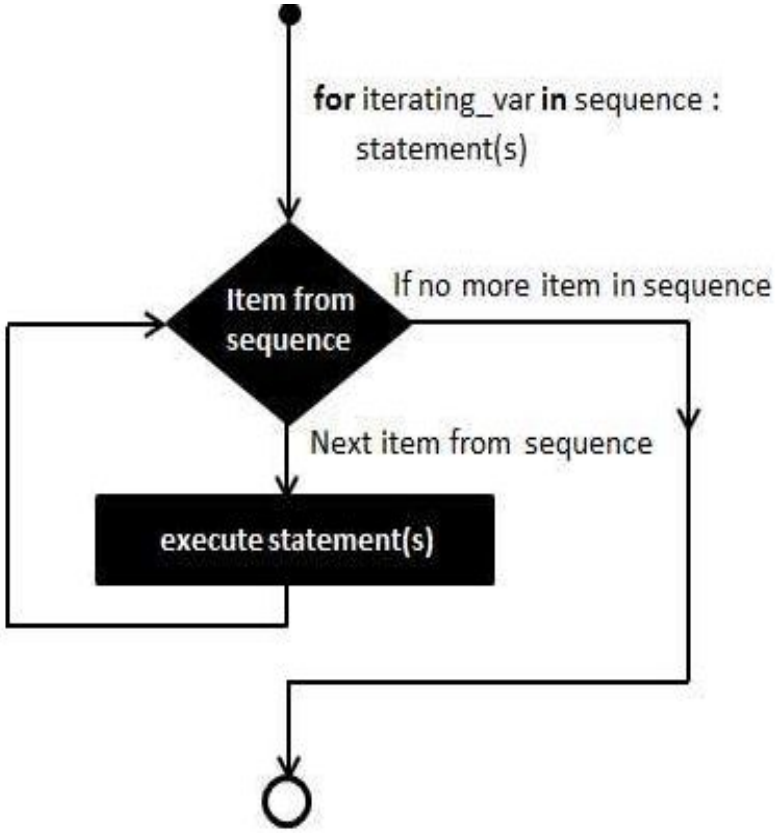
Continue Statement:- With the **continue** statement we can stop the current iteration of the loop, and continue with the next.

for loop

Syntax of for loop

for iterating_var in sequence:

    statements(s)



Flow Diagram of for loop

for loop

Example of for loop:-

```
fruits = ["Apple","Banana","Cherry"]
for x in fruits:
    print(x)
```

Output:-

```
Apple
Banana
Cherry
```

for loop

Using range() in for loop: The **range()** function returns a sequence of numbers, starting from 0 by default, and increments by 1 (by default), and ends at a specified number.

Example

```
for x in range(6):
    print(x)
else:
    print("Finally finished!")
```

Output:-

```
0
1
2
3
4
5
Finally finished
```

Else in for loop

Using else Statement with Loops:- Python supports to have an **else** statement associated with a loop statement. The **else** statement is executed when the condition becomes false.

Example of else with while loop

```
for x in range(5):
    print(x, "is less than 5")
else:
    print(x, "is not less than 5")
```

Output :

0 is less than 5

1 is less than 5

2 is less than 5

3 is less than 5

4 is less than 5

5 is not less than 5 (when x =5 the controls executes the else statements)

Nested loop

Python programming language allows to use one loop inside another loop.

Syntax of nested for loop :

```
for iterating_var in sequence:
    for iterating_var in sequence:
        statements(s)
    statements(s)
```

syntax for a nested while loop

```
while expression:
    while expression:
        statement(s)
    statement(s)
```

Self Assessment Question

1. Which one of the given options will be the output for the following code?

```
x = ['ab', 'cd']
 for i in x:
        i.upper()
print(x)
```

a. [‘ab’, ‘cd’].

b. [‘AB’, ‘CD’].

c. [None, None].

Answer: [‘ab’, ‘cd’].

Self Assessment Question

2. What is the output of the following?

```
for i in  range(2):
    print i
for i in range(4,6):
    print i
```

Answer: Output:

0
1
4
5

Assignment

1. Write a python code to find if the given number is prime or not.
2. Write a python code to find LCM and GCM of a given list.

Summary

- Discussed about the decision making with example
- Discussed about the looping concepts in python with example

NURTURE Education Solutions TOMORROW'S HERE

Document Links

Topics	URL	Notes
Decision Making in Python	https://www.tutorialspoint.com/python/python_decision_making.htm	This link explains different Decision Making Statements in Python
If statement	https://www.tutorialspoint.com/python/python_if_statement.htm	The flow diagram of if statement has been taken from this link and also explains about the if statement concepts
If else	https://www.tutorialspoint.com/python/python_if_else.htm	The flow diagram has been taken from this link and also explains about the if else statement
Nested if else	https://www.tutorialspoint.com/python/nested_if_statements_in_python.htm	This link explain about the nested if else statement

Document Links

Topics	URL	Notes
Loop	https://www.tutorialspoint.com/python/python_loops.htm	The flow diagram of looping has been taken from this link
While loop	https://www.tutorialspoint.com/python/python_while_loop.htm	The flow diagram of while loop has been taken from this link
For loop	https://www.tutorialspoint.com/python/python_for_loop.htm	The flow diagram of for loop has been taken from this link
Looping in Python	https://www.tutorialspoint.com/python/python_loops.htm	This link explains about the looping in python

Video Links

Topics	URL	Notes
Loop in python	https://www.youtube.com/watch?v=zFvoXxeoosI	This link explains about the loop in python