Experiment No. 2
To implement Conditional Statements and Loop in python
Date of Performance: 23-02-2024
Date of Submission: 6-03-2024

Experiment No. 2

Title: To implement Conditional Statements and Loop in python

Aim: To study, and implement Conditional Statements and Loop in python

Objective: To introduce Conditional Statements and Loop in python

Theory:

1. Conditional Statements

There comes situations in real life when we need to do some specific task and based on some specific conditions and, we decide what should we do next. Similarly there comes a situation in programming where a specific task is to be performed if a specific condition is True. In such cases, conditional statements can be used. The following are the conditional statements provided by Python.

if

if..else

Nested if

if-elif statements.

Let us go through all of them.

if Statement

If the simple code of block is to be performed if the condition holds true than if statement is used. Here the condition mentioned holds true then the code of block runs otherwise not.

if..else Statment

In conditional if Statement the additional block of code is merged as else statement which is performed when if condition is false.

Nested if Statement

if statement can also be checked inside other if statement. This conditional statement is called nested if statement. This means that inner if condition will be checked only if outer if condition is true and by this, we can see multiple conditions to be satisfied.

if-elif Statment

The if-elif statement is shoutcut of if..else chain.While using if-elif statement at the end else block is added which is performed if none of the above if-elif statement is true.

2. Looping in python

Python programming language provides following types of loops to handle looping requirements. Python provides three ways for executing the loops. While all the ways provide similar basic functionality, they differ in their syntax and condition checking time.

While Loop:

In python, while loop is used to execute a block of statements repeatedly until a given a condition is satisfied. And when the condition becomes false, the line immediately after the loop in program is executed.

for in Loop:

For loops are used for sequential traversal. For example: traversing a list or string or array etc. In Python, there is no C style for loop, i.e., for (i=0; i<n; i++). There is "for in" loop which is similar to for each loop in other languages. Let us learn how to use for in loop for sequential traversals.

CODE:

```
a = input("Enter your value of a: ")

b = input("Enter your value of b: ")

#if statement
```

```
if b > a:
    print("b is greater than a")
#else if statement
elif a == b:
  print("a and b are equal")
#else statement
else:
  print("a is greater than b")


#nested if statement
i = 10
if (i == 10):

    if (i < 15):
        print("i is smaller than 15")


    if (i > 5):
        print("i is greater than 5")
#while
count = 0
while (count < 2):
    count = count + 1
    print("Hello")
```

```
#for loop

n = 5

for i in range(0, n):

    print(i)
```

RESULTS:

Conclusion: Conditional and looping statements in Python, such as if, elif, else, while, and for, enable developers to control the flow of their programs and execute code based on specific conditions or iterate over sequences of data. These constructs empower programmers to create dynamic and responsive applications by enabling decision-making and repetition. Leveraging these statements, developers can implement logic to handle diverse scenarios, iterate over collections, and automate repetitive tasks efficiently. Python's clear and expressive syntax, along with its rich set of built-in functions and libraries, make conditional and looping statements integral to writing concise, readable, and powerful code for a wide range of applications and problem domains.