Vidyavardhini’s College of Engineering & Technology

Department of Computer Engineering

Experiment No. 2
To implement Conditional Statements and Loop in python
Date of Performance:17/01/2024
Date of Submission:24/01/2024

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:

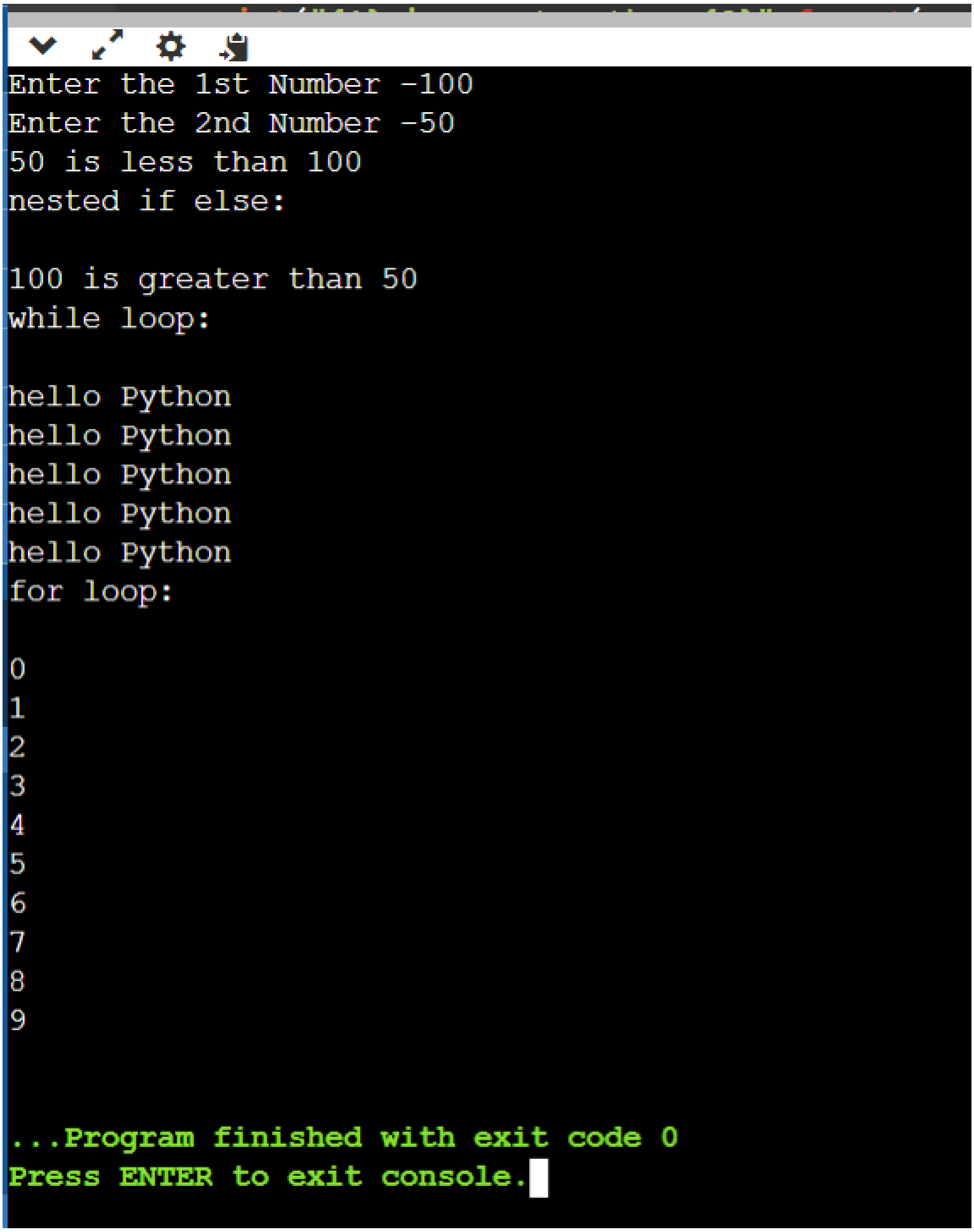
For loops are used for sequential traversal. For example: traversing a list or string or array etc. In Python, there is no C style for loop, i.e., for (i=0; i<n; i++). There is "for in" loop which is similar to for each loop in other languages. Let us learn how to use for in loop for sequential traversals.

Code:

```
# input statments


num1 = int(input("Enter the 1st Number -"))

num2 = int(input("Enter the 2nd Number -"))


#conditions statments

# 1 if statments


if num1 < num2:

    print("given condition is True")


#2 if else


if num2 > num1:

    print("{1} is greater than {0}".format(num1,num2))

else:

        print("{1} is less than {0}".format(num1,num2))


# 3 elif


if num1 < num2:

    print("given condition is True")
```

```
elif num1 == num2:
    print("its equal")
else:
     print("{0} is greater than {1}".format(num1,num2))




# while loop

count=0

while(count<5):
    count=count +1
    print("hello Python")



# for loop

n = 10
for i in range(0, n):
            print(i)
```

Output:



```
Enter the 1st Number -100
Enter the 2nd Number -50
50 is less than 100
nested if else:

100 is greater than 50
while loop:

hello Python
hello Python
hello Python
hello Python
hello Python
for loop:

0
1
2
3
4
5
6
7
8
9


...Program finished with exit code 0
Press ENTER to exit console.
```

Conclusion:

Conditional statements and loops are essential programming constructs in Python that allow us to control the flow of our code and execute certain blocks of code based on conditions or iteratively. In this implementation, we explored several of these constructs:

1. **if Statement**: The `if` statement allows us to execute a block of code only if a specified condition is true. It provides a basic decision-making capability.

2. **if..else Statement**: The `if..else` statement extends the `if` statement by providing an alternative block of code to execute when the condition evaluates to false.

3. **Nested if Statement**: Nested `if` statements involve placing one `if` statement inside another. This allows for more complex decision-making based on multiple conditions.

4. **if-elif Statements**: The `if-elif` statements provide a way to test multiple conditions. If the condition in the `if` statement is false, it proceeds to check the conditions in the subsequent `elif` statements until a true condition is found or the `else` block is executed.

5. **while Loop**: The `while` loop repeatedly executes a block of code as long as the specified condition evaluates to true. It is useful when the number of iterations is unknown beforehand.

6. **for Loop:** The `for` loop iterates over a sequence such as a list, tuple, or string and executes the block of code for each element in the sequence. It is commonly used for iterating through collections of data.

By understanding and using these constructs effectively, programmers can create more sophisticated and efficient Python programs capable of handling various scenarios and tasks.