Experiment 1.3

Page No.
Date

Aim: To understand the control flow statements in Python.

Theory:

A program's control flow is the order in which the program code executes. The control flow of a Python program is regulated by conditional statements, loops and function calls. Python has three types of control structures:

1. Sequential: Sequential statements are a set of statements whose execution process happens in a sequence. The problem with sequential statements is that if logic has broken in any one of the lines, then complete source code execution will break.

2. Decision control statements: The selection statement allows a program to test several conditions and execute instructions based on which condition is true. Some decision control statements are:

a. if statement - if statement is the simplest decision-making statement. The if statement contains a logical expression using which data is compared and a decision is made based on the result of the comparison. It is used to decide whether a certain statement or block of statements will be executed or not i.e. if a certain condition is true then a block of statement is executed otherwise not.

b. if-else statement - The if statement alone tells us that if a condition is true, it will execute a block of statements and if the condition is false, it won't. But what if we want to do something else if the condition is false. Here comes the else statement.

c. if... elif... else statement: An else statement can be combined with an if statement. An else statement contains the block of code that executes if the conditional expression in the if statement resolves to 0 or a False value. The else statement is an optional statement and there could be at most only one else statement following if. 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.

d. Shorthand if-else statement: This can be used to write the if-else statements in a single line where there is only one statement to be executed in both if and else block.

3. Repetition: A repetition statement is used to repeat a block of programming instructions. In python, we generally have two loops:

a. for loop: For loops are used for sequential traversal. For example: traversing a list or string or array etc.

Syntex:

for iterating_var in sequence:
    statement(s)

b. while loop: A while loop statement in Python programming language repeatedly executes a target statement as long as a given condition is true.

Syntex:

```
while expression:
    Statement(s)
```

c. nested loops: Python programming language allows to use one loop inside another loop.

Syntex:

```
for iterator_var in sequence:
    for iterator_var in sequence:
        Statement(s)
        Statement(s).
```

* Loop control statements: Loop control statements change execution from its normal sequence. when execution leaves a scope, all automatic objects that were created in that scope are destroyed. Python supports the following control statements:

a. break statement: It terminates the current loop and resumes execution at the next statement, just like the traditional break statement in C. The most common use for break is when some external condition is triggered requiring a hasty exit from a loop. The break statement can be used in both while and for loops. If you are nested loops, the break statement stops the execution of the innermost loop and start executing the next line of code after the block.

Page No.
Date

b. continue statement: It returns the control to the beginning of the while loop. The continue statement rejects all the remaining statements in the current iteration of the loop and moves the control back to the top of the loop. The continue statement can be used in both while and for loops

c. pass statement: It is used when a statement is required syntactically but you do not want any ~~comma~~ command or code to execute. The pass statement is a null operation, nothing happens when it executes. The pass is also useful in places where your code will eventually go, but has not been written yet.

Programs:

A. Program to check odd or even numbers.

```python
num = int(input("Enter a number: "))
if (num % 2) == 0:
    print("{0} is Even".format(num))
else:
    print("{0} is Odd".format(num))
```

Output:

Case 1- Number = 56

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS E:\Python> python -u "e:\Python\oddeven.py"
Enter a number: 56
56 is Even
PS E:\Python>
```

Case 2 - Number = 17

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS E:\Python> python -u "e:\Python\oddeven.py"
Enter a number: 17
17 is Odd
PS E:\Python>
```

B. Program check whether the number is positive or negative or zero.

```python
num = float(input("Enter a number: "))
if num > 0:
    print("Positive number")
elif num == 0:
    print("Zero")
else:
    print("Negative number")
```

Output:

Case 1 - Number = 89

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS E:\Python> python -u "e:\Python\positivenegative.py"
Enter a number: 89
Positive number
PS E:\Python>
```

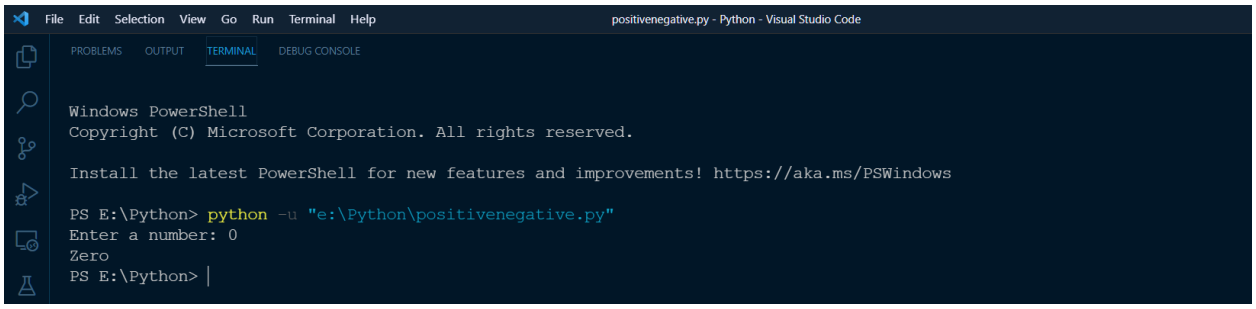
Case 2 - Number = -56

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS E:\Python> python -u "e:\Python\positivenegative.py"
Enter a number: -56
Negative number
PS E:\Python>
```

Case 3 - Number = 0



Conclusion: We have understood the control flow statements in Python.