# TITLE: CONTROL STATEMENT

**OBJECTIVE**:

* To understand control statements in python.

**THEORY**:

Control statement, control structure or flow control structure is a programming concept that determines the order in which the statements are executed in a program. Control statements regulates the flow of execution based on the specified conditions and loops. There are three major types of control statements:

1. **Selection statements**: These statements determine the part of the program that is to be executed based upon the condition. Examples include **if**, **else**, **elif** (else if), and **switch** statements.
2. **Iteration statements**: These statements enable the repetition of a block of code until a specified condition is met. Examples: **while**, **for**, and **do-while** loops.
3. **Jump statements**: These statements control the flow of program by jumping to a certain point in a program based on the certain condition. Example: **break**, **continue**, **return** and **pass**.

Some control statements:

* **While statements:** This is the iterative statement that executes the iteration until specified condition is true. This method is particularly used when finite iterative steps are not known. It first checks for the condition then only execute the block of code.

1. while condition:

2. # Code to execute while the condition is true

3.

* **Do……while statements:**

Alike while statement, but in this statement the iteration will execute at least once though the condition is not satisfied. In this iteration, first the code is executed then after condition is checked. Python does not have built-in do…while statement. Equivalent do…while statement in python:

1. while True:

2. # Code to execute

3. if not condition:

4. break

* **For statements:**

This is the most preferred iteration. It is used to iterate over finite elements in iterable (Eg: range, string, list, tuple, etc.).

1. for variable in iterable:

2. # Code to execute for each iteration

* **Foreach statements:**

In Python, the for statement itself serves as a foreach statement, iterating over elements in an iterable and executing the code block for each element.

1. for variable in iterable:

2. # Code to execute for each iteration

* **Nested Loops:**

Nested loops are used for handling complex iteration scenarios. A loop inside a loop is called nested loop. The "inner loop" will be executed one time for each iteration of the "outer loop".

1. for variable\_1 in iterable\_1:
2. for variable\_2 in iterable\_2:
3. #code to execute..

* **Break statements:**

Break statement is used to terminate the loop before completion if certain condition is met. It exits a loop prematurely when a certain condition is met, jumping to the statement immediately following the loop.

1. While contition:
2. #code
3. If condition:
4. break

* **Continue statement:**

It skips the rest of the code inside a loop for the current iteration and moves to the next one.

1. for variable in iterable:
2. if condition:
3. continue

* **Pass statement:**

Acts as a placeholder where syntactically some code is required but no action is intended. It essentially does nothing and allows the program to pass over it without causing an error.

1. if some\_condition:
2. pass

* **Else in for Loops:**

The else keyword in a for loop specifies a block of code to be executed when the loop is finished. Example:

1.for x in range(6):  
2.  print(x)  
3.else:  
4.  print("Finally finished!")

* **The range() function:**

To loop through a set of code a specified number of times, we can use the range() function, 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.

1. range(begin, end, step)

1. for x in range(2, 20, 3):

2.  print(x)  #output: 2,5,8,11,14,17,20

LAB ACTIVITY:

1. #inporting required libraries

2. import random

3. import msvcrt   #library to use getch

#Q1. Write a python program to find the sum of n natural numbers.

1. print("Output Q1:")

2. n=int(input("Enter the no. of terms: "))

3. sum=0

4. for i in range(n+1):

5.     sum+=i

6. print(f"The sum of {n} natural numbers is: {sum}\n")

#Q2. Write a program to read integer from user. For all non negative integers i<n, print i^2.

1. print("Output Q2:")

2. user\_input=int(input("Enter the integer: "))

3. print("Square: ",end="")

4. for i in range(user\_input):

5.     if i<user\_input:

6.         print(i\*\*2,end='\t')

#Q3. Write a python program that prints all the numbers from 0 to 6 except 0 and 6 using 'continue' statement.

1. print("\n\nOutput Q3:")

2. for i in range(7):

3.     if(i==3 or i==6):

4.         continue

5.     else:

6.         print(i,end="\t")

#Q4. Write a python program to get the fibonacci series upto n terms.

1. print("\n\nOutput Q4:")

2. n=int(input("Enter the no. of terms: "))

3. print("Fibonacci Series: ",end=" ")

4. first\_int=0

5. next\_int=1

6. for i in range(n+1):

7.     print(first\_int, end=" ")

8.     third\_int=first\_int+next\_int

9.     first\_int=next\_int

10.     next\_int=third\_int

#Q5. Write a python program to find those numbers which are divisible by 7 and multiple of 5 between 1500 and 2700 (both included).

1. print("\n\nOutput Q5:")

2. print("Numbers between 1500 and 2700 that are divisible by 7 and are multiple of 5: ",end=" ")

3. for i in range(1500,2701,5):

4.     if(i % 7 ==0):

5.         print(i,end=" ")

#Q6. Write a python program to guess a number between 1 t0 9. If user prompted right guess print" Well guessed!" otherwise prompt to guess again.

1. number=random.randint(0,9)

2. print("\n\nOutput Q6: ")

3. while True:

4.     print("Guess the number:", end=" ")

5.     guess\_int=int(input("<< "))

6.     if(guess\_int==number):

7.         print("Well Guessed !!")

8.         break

9.     else:

10.         print("Try Again !!")

#Q7. Write a python program to develop a rock paper scissor game, restart the game until the user press 'n' when the game ends.

1. print("\nOutput Q7:")

2. isTrue=True

3. while isTrue:

4.     system\_call=random.choice(['rock','paper','scissor'])

5.     your\_call=input("Scissor:Paper:Rock:???:::")

6.     system\_call=system\_call.lower()

7.     your\_call=your\_call.lower()

8.     print(f"Your Call: {your\_call}\t My Call: {system\_call}\t Result: ",end="")

9.     if (system\_call=='rock'and your\_call=='paper') or (system\_call=='paper' and your\_call=='scissor') or (system\_call=='scissor' and your\_call=='rock'):

10.         print("You Won !!")

11.     elif (system\_call==your\_call):

12.         print("It's a draw !!")

13.     else:

14.         print("Oops ! you lost.")

15.     ch=msvcrt.getch().decode('utf-8')   #convert byte-type to string

16.     if ch=='n' or ch=='N':

17.         break

#Q8. Write a python program to create a multiplicatio table from 1 to 10 of a number.

1. print("\nOutput Q8: ")

2. number=int(input("Enter the number:"))

3. print(f"Multiplication Table of {number} from 1 to 10:")

4. for i in range(1, 11):

5.     print(f"{number} X {i} = {number\*i}")

#Q9. Write a python program that accepts a word from the user and reverses it:

1. print("\nOutput Q9:")

2. word=input("Enter the word: ")

3. print("The reversed word is: ",end="")

4. for i in range(len(word),0,-1):

5.     print(word[i-1],end="")

6. print("\n")

#Q10. Python program to construct pattern:

\*

\*\*

\*\*\*

\*\*\*\*

\*\*\*\*\*

\*\*\*\*

\*\*\*

\*\*

\*

1. print("\nOutput Q10: ")

2. for i in range(1,10):

3.     if i<5:

4.         for j in range(1,i+1):

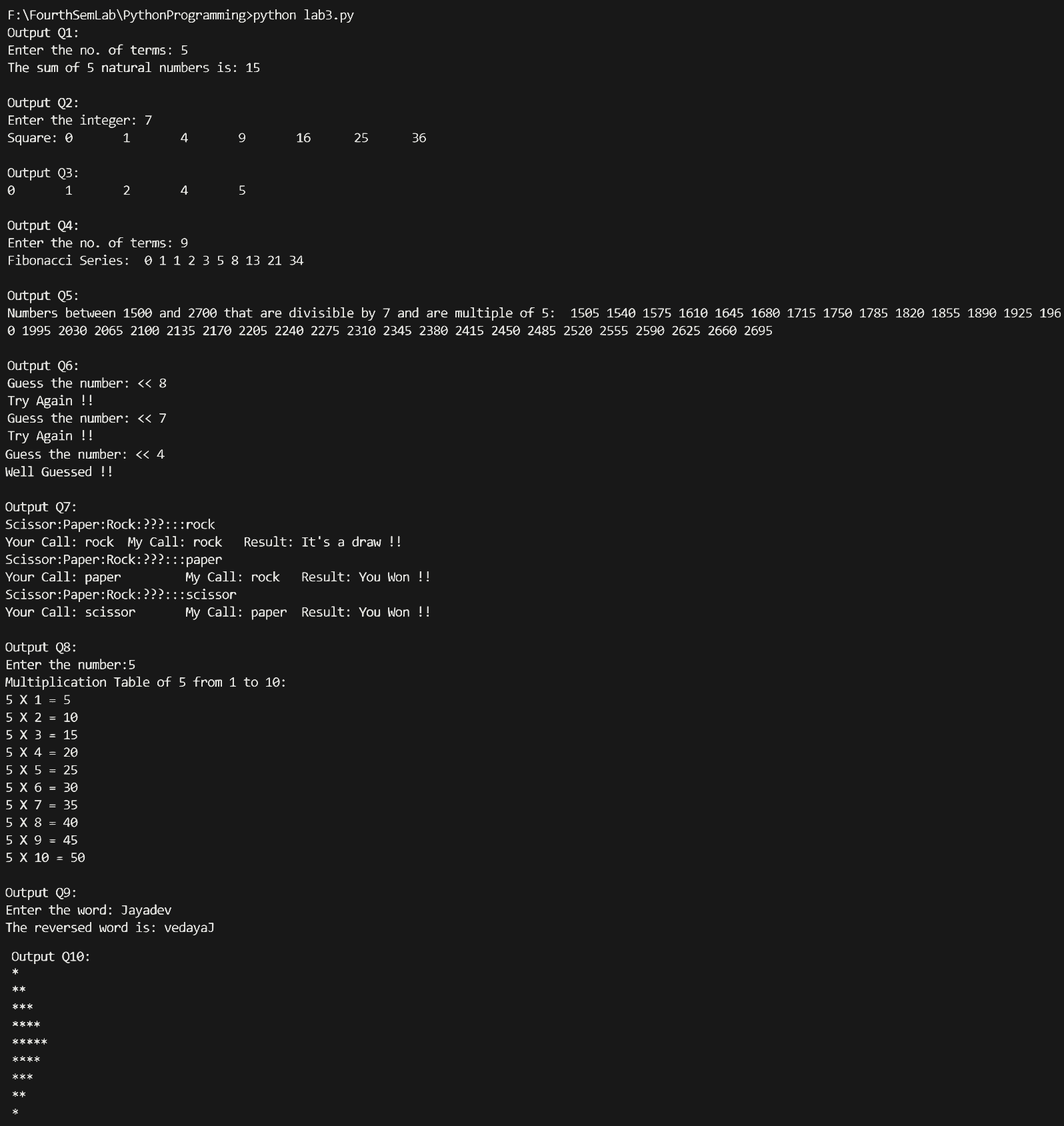
5.             print("\*",end="")

6.     else:

7.         for j in range(i,10):

8.             print("\*",end="")

9.     print("")

OUTPUT:   
  
CONCLUSION:

In this lab activity, we gained the understanding of control statements like: for and while loop, and continue, break and pass statement. These control statements are used for controlling the flow of sequence in program. The lab program and the outcome of the program are as shown above. This result clarifies our understanding of control statements with python programming.