TASK-8 Implement Python generator and decorators.

Aim: Write a Python Program to implement Python generator and decorators

Algorithm:

1. Define Generator function

   Define the function number_sequence (start, end, step=1)

2. Initialize current value:

   Set current to the value of start

3. Generate sequence

   * while current is less than or equal to end
     1. Yield the current value of current
     2. Increment current by step

4. Get user Input

   - Read the starting number (start) from user input
   - Read the ending number (end) from user input
   - Read the step value (step) from user input

5. Create Generator object

   create a generator object by calling number_sequence (start, end, step) with user provided values

6. Print Generated sequence

   - Iterate over value produced by the generator object
   - Print each value

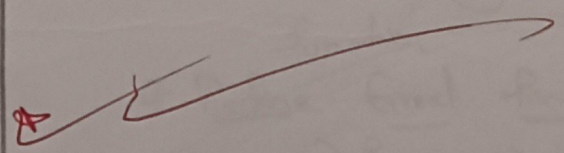
output

Enter the starting number : 1
Enter the ending number : 50
Enter the step value : 5

1
6
11
16
21
26
31
36
41
46

Program

```
def number_sequence (start ,end step =1);
    current =start
    while create <= end;
        yield current
        corrent += step
    start =int(input ("Enter the starting number:"))
    end = int(input("Enter the ending number:"))
    setP= int(input("Enter the step value:" ).)
# create the generator
sequence_generator = number_sequence(start ,end, step)
# Print the generated sequence of numbers
for number in sequence-generator
    Print # (number)
```



Result: Thus the c.Program is Python generator and decorators is executed Verified successfully.

Task-8.2 Imagine you are working on a messaging application that needs to formate

Aim: To Produce a default sequence of number start from 0 ending 10.

Algorithm:

1. create Decorators:
   - Define uppercase_decorator to convert the result of a function to uppercase
   - Define lowercase_decorator to convert the result of a function to lowercase
2. Define function
   - Define shout function to return the input text Apply @ uppercase-decorator to this function
   - Define whisper function to return the input text Apply @ lowercase_decorator to this function
3. Define Greet function

   Define greet function that:

   1) Accepts a function (func) as input
   2. Calls this function with the text "Hi" I am created by a function Passed as an argument
   3. Prints the result.

output

0
1
2

4. Create the Program

1. Call greet (shout) to Print the greeting in uppercase
2. Call greet (whisper) to Print the greeting in lowercase

Program

```
def uppercase_decorator (func):
    def wrapper (text):
        return func (text).upper ()
    return wrapper
def lowercase_decorator (func):
    def wrapper (text):
        return func (text). lower ()
    return wrapper
@ uppercase_decorator
def shout (text):
    return text
@ lowercase_decorator
    def whisper (text):
    return text:
def greet (func):
    greeting = func ("Hi, I am created by a function
        Passed as an argument.")
    Print (greeting)
    greet (shout)
    greet (whisper)
```

EL TECH - CSE	
EX NO.	8
PERFORMANCE (5)	5
RESULT AND ANALYSIS (3)	3
VIVA VOCE (3)	3
RECORD (4)	4
TOTAL (15)	15
SIGN WITH DATE	

Result: Thus the Python Program to implement Python generator and decorators was successfully executed the output was verified.