Task-8: Implement Python generators and decorators.

Aim:

Write a Python Program to implement Python generator & decorators.

8.1.a Produce a sequence of numbers when provided with start, end and step values.

Algorithm:

1. Define the function number_sequence(start, ~~step~~ end, step=1).
2. Set current to the value start.
3. while current is less than or equal to end:
   i. Yield the current value of current.
   ii. increment current by step.
4. Get user Input:
   i. Read the starting number (start), ending number (end), step value (step) from the user.
5. Create a generator values produced by the generator objects calling number_sequence(start, end, step).
6. Iterate over the values produced by the generator object.
7. Print each value.

Program:

```
def number_sequence(start,end,step):
    current = start
    while current <= end:
        yield current
        current += step
start = int(input("Enter the starting number:"))
end = int(input("Enter the ending number:"))
step = int(input("Enter the step value:"))
```

Output

Enter the Starting number: 1

Enter the ending number 150

Enter the step value: 5

1

6

11

16

21

26

31

36

41

46

```
Sequence_generator= number_sequence(start, end, step)
for number in sequence_generator:
    Print(number).
```

8.1(b) Produce a default numbers starting from 0, ending at 10, and with a step of 1 if no values are provided.

Algorithm:

1. Define the function my_generator(n) that takes a parameter n.
2. Set value to 0.
3. while value is less than n:
   - Yield the current value.
   - Increment value by 1.
4. Call my_generator(11) to create a generator object.
5. For each value produced by the generator object:
   - Print value.

8.1.(b) Program:-

```
def my_generator(n):
    value=0
    while value<n:
        yield value
        value+=1
for value in my_generator(3):
    Print(value).
```

8.2 Algorithm:-

1. Define uppercase_decorator, lowercase_decorator to convert the result of a function to uppercase and lower case.
2. Define short function to return the input text. Apply @upper_case_decorator to this function.
3. Define whisper function to return the input text. Apply @lower_

output

0
1
2

Output:-

HI, I AM CREATED BY A FUNCTION PASSED AS AN ARGUMENT.

hi, i am created by a ~~create~~ function passed as an argument.

case_decorator to this function.

4. Accept a function (func) as input.
5. Call function text "Hi, I am created by the function passed as as argument".
6. Call greet(short) to print the greeting in uppercase
7. 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 warapper(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)
```

VELTECH	
EX No.	8
PERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	5
VIVA VOCE (5)	5
RECORD (5)	5
TOTAL (20)	20
SIGN WITH DATE	

Result:-

Thus the Python program to implement Python generator and decorators was successfully excuted and the output was verified.