Task:- 8 implement python generator and decoitors

Aim:- Write a python program to implement python generator and decorators.

Algorithm:-

1. Define Generator function
- Define the function number - Sequences (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
- Yield the current value of current
- 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_sequences (start, end, step) with user-provided values.
6. print Generator Sequence
- Iterate over the values produced by the Generator
- print each value

Program

```
def number-sequence (start, end, step=1):
Current = start
while current <= end:
 Yield current
 Current t= step
Start = int (intput ("Enter the starting number:"))
End = int (intput ("Enter the ending number:"))
Step = int (intput ("Enter the step value:"))
```

```
# create the generator
sequence_generator = number_sequence(start, end, step)
# print the generated sequences of numbers
for number in sequence_generator:
    print(number)
```

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

Ex:- 8.1(b)

Aim:- To write python program my-generator using loop statem. cut

Algorithm:-

1. Start function
- Define the function my-generator(n) that takes a parameter n
2. Initialize Counter
- Set value to 0
3. Generate values:
- while value is less than n
- Yield the Current value
- Increment value by 1
4. Create generator object
- Call my-generator(11) to Create a generator object
5. Iterate and print value
- for each value produced by the generated object

8.1(b) program

```
def my-generator(n):
# Initialize Counter
value = 0
# loop untill Counter is less than n
while value < n:
# produce the current value of the Counter.
~~while value : n~~
yield value
# increment the Counter
value += 1
# iterate over the generator object produced by my-generator
for value in my-generator(3):
# print each value produced by generator
print(value)
```

output :-

0
1
2

Task 8.2

Aim:- To write a python program using functions they decorate by Converting the text case

Algorithm:-

1. Create decorators
   - Define uppercase_decorators to Convert the result to a function to upper case
   - Define lower case_decorators to convert the result to function to lowercase
2. Define functions
   - Define should function to return the input text Apply @upper case-decorator of this function.
   - Define whisper function to return the input text. Apply @lowercase_decorator to this function.
3. Define Greet function.
   - Accepts a function (ffunc) as input.
   - Calls this function with the text "Hi I am created by a function
4. Execute the program
   - Call greet(shout) to print the greeting in uppercase

Program:-

```
def upper case - decorator (func):
def wrapper(text):
  return func(txt).upper()
return wrapper
def lower.case - decoration (func):
def wrapper (text):
  return func (txt).lower()
return wrapper
```

Output:

HI, I AM CREATED BY A FUNCTION PASSED AS AN ARGUMENT
hi, i am created by a function passed as an Argument

```
@uppercase_decorator
def shout(text):
    return text
@lowercase_decorator
def Showwhisper(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 (3)	5
VIVA VOCE (3)	5
RECORD (4)	
TOTAL (15)	
WITH DATE	15

Result:- Thus, the python program to implement python generator and decorators was successfully executed and the output was verified.