17/09/25 Task 8: Implement python generator and decorators

Aim: write a python program to implement Python generator and decorator

Algorithm:-

1. Define Generator function:
   Define the function number-sequence
2. Initialize current value:
   set current to the value to start
3. Generate sequence:
   - while current is less than & equal to end:
   - Yeild the current value of current
   - Increment current by step
4. Get user Input:
   - Read the starting number from the user input
   - Read the ending number from the user input
   - Read the step value from user input.
5. Create Generator Object
   - Create a generator object by calling number-sequence
6. Print Generated sequence:
   - Iterate over the values produced by generator object
   - Print each value.

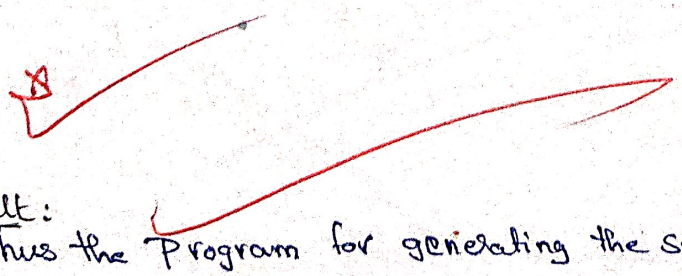
Program:

```
def number_sequence (start, end, step=1)
    Current = start
    While current <= end;
        Yeild current
        Current + = step
    Start = int (input ("Enter the starting number :"))
    end = int (input ("Enter the ending number :"))
    Step = int (input ("Enter the step value "))
#Create the generator
sequence_generator = number_sequence (start, end, step)
```

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
```

```
for the number in sequence generator:
    Print (number)
```



Result:

Thus the Program for generating the sequence of numbers was Successfully verified.

Task No : 8 11b)

Aim : To write the python program my-generator using loop statements.

Algorithm :

1. Start function :
   - Define the function my-generator (n) that takes a parameter n
2. Initialize counter : Set values to 0
3. Generate values : while values is less than n :
   - yeild current Object
   - Increment by value :
4. Create Generator object
   - call my. generator (11) to create a generator object
5. ~~Generator~~ Iterate and print values :
   - For each value procedure by the generator object.

Program :

```
def my-generator(n):
#initialize counter
value =0
#loop until counter is less than n
While value <n:
  #produce the current value of the counter
  Yield value
#iterate over the generator object produced by my-generator
for value in my-generator (3):
  #Print each value produced by generator
  print (value)
```

EX NO.	5
PERFORMANCE (5)	5
RESULT AND ANALYSIS (3)	5
VIVA VOCE	5
	15

Result :

Thus the python program my-generator using loop statements was successfully executed

Output: 0
1
2

Task No: 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 of a function to uppercase
2. Define functions:
   - Define Shout function to return the input text
3. Define greet function:
   - Accepts a function (name) as input
   - Print the result
4. Execute the program
   - call greet to print text.

Program:

```
def upper_case_decorator(func):
    def wrapper(text):
        return func(text).upper()
    return wrapper
def lower_case_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)
```

Result:

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

Output:

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

hi, t am created by a fenction passed as an argument