TASK-8 IMPLEMENT PYTHON GENERATOR AND DECORATORS
15/9/25

Aim: write a python program to implement Python generator & decorators.

8·1: write a python program that includes a generator function to produce a sequence of numbers. Produce a sequence of numbers when provided with start, end & step values

ALGORITHM:

```
1. Define generator function:
      · Define the function number_sequence
                                   (start, end, step=1).
2. intialize 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.
6. Print Generated sequence :
  · Iterate over the values produced by the
    generator object.
  · Print each value.
```

PROGRAM:

```
def number_sequence (start, end, step=1):
    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:"))
# create the generator
sequence_generator = number_sequence (start, end, step)
# print the generated sequence of numbers
    for number in sequence_generator:
        print (number)
```

8.2 : Imagine you are working on a messaging application that needs to format user's preferences. You are provided with two decorators: uppercase_decorator and lowercase_decorator. These decorators modify the behaviour of fun they decorate by converting the text to uppercase & lowercase respectively.

ALGORITHM : 1. Create Decorators :

- Define uppercase_decorator to convert the result of a function to upper case.

2. Define Functions :

- Define whisper function to return the input text.

Apply @ lowercase_decorator to this function.

3. Define Greet function :

- Accepts a function (func) as input.
- prints the result.

4. Execute the program :

- Call greet(shout) to print the greeting in uppercase.
- 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()
```

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
```

OUTPUT :

```
HI, IAM CREATED BY A FUNCTION PASSED AS AN
ARGUMENT.
hi, i am created by a function passed as an
argument.
```

(b) lower case - 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)
```

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

<u>RESULT</u> : Thus the python program to implement python generator and decorators was sucessfully executed & the output was verified.