Task 8: Implement Python generator

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

~~Program~~: **Algorithm:**

1. Define Generator function:
2. initialize current value
3. Generate Sequence
4. Get user input
5. Create Generator object

8.1 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
  step = int (input ("Enter the step Value"))
  Sequence_generator = number_sequence (start, end, step)
  for number in sequence_genarator
        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
```

Task 8: Implement Python generator

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

~~Program~~ **Algorithm:**

1. Define Generator function:
2. initialize current value
3. Generate Sequence
4. Get user input
5. Create Generator object

8.1 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
step = int (input ("Enter the step value"))

Sequence-generator = number_sequence (start, end, step)

for number in sequence_genarator
        print (number)
```

Procedure a default sequence of number starting from 0 ending at 10, and with a step of 1 if no values are Provided

Algorithm:

```
1. start function
2. initialize counter
3. Generate values
4. Create Generator object
5. iterate and Print values.
```

8.1 (b) Program

```
def number_sequence (start, end)
  def my generator
   value = 0
   while value <n:
    yeild value
    value +=1
  for value in my_generator(3):
     Print(value)
```

2. Imagine you are working on a message application that needs to format message differently based on the users Peferences.

Algorithm:

1. Create Decorators
2. Define functions
3. Define Greet function
4. Execute the Program

Program

```
def uppercase_~~determi~~ decorator (func):
 def wrapper (text):
 return func(text) upper()
 return wrapper
 def wrapper
 return func (text) lower()
 return wrapper
 @ upper case - decorator
   def . shout (text)
   return text
 @ lowercase _decorator
   def shout (text):
    return text

def greet(func):

  greeting = func("Hi, I am Created by a function Passed
                        as argument")
```

output:

```
Number of lines with 'ERROR' is 2

HI, I am Created by A function Passed as
       an argument
hi, I am Created by a function Pass as an
          argument
```

```
Print (greeting)
greet (shout)
greet (whisper)
```

VEL TECH	
EX NO.	8
PERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	5
VIVA VOCE (5)	5
ECORD (5)	
(20)	15

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