Task 8:- implement python generator and decorators

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

8.1 Write a python program that includes a generator function to produce a seaverice of numbers the program generator should be able to

(a) Produce a seaverice of numbers when provided with start, end and step values

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

Produce a seaverice of numbers when provided with start, end, and step values

Algorithm:-

1) Define Generator function
   - Define the function number_seaverice (start, end, step=1)

2) Initialize current value
   - Set current to the value of start

3) Generate seaverice:-
   - while current is less than or eaual 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_sequence (start, end, step) with user-provided values

6) Print Generated sequence

- iterate over the values produced by the generator object
- Print each value

8.1 Program

```
def number_sequence(start, end, step=1):
    current = start
    while current <= end
    yield 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 generator sequence of numbers
    for number in sequence_generator:
        print(number)
```

OUTPUT:-

```
Enter the Starting number: 1
Enter the ending number: 50
Enter the Step value is 5
1
6
11
16
21
26
31
36
41
46
```

Produce a default seavence of numbers starting from 0, ending at 10, and with a step of 1 if no values are provided

Algorithm :-

1) Start function :-
   - Define the function my_gen 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 values
   - for each values produced by the generator object
     - Print value

8.1 (b) 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
```

OUTPUT :- 0
1
2

s/p.

```
        # 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~~

8.2 Imagine you are working on a messaging application that needs to format messages differently based on the user's preferences. Users can choose to have their messages automatically converted to uppercase (for emphasis) or to lowercase (for a softer tone). You are provided with two decorators uppercase_decorator and lowercase_decorator. These decorators modify the behaviour of the functions they decorate by converting text to uppercase or lowercase respectively. Write a program to implement it.

Algorithm:-

1) Create decorators:
   - Define uppercase_decorator to convert the result of a function to uppercase
   - Define lowercase_decorator to convert the result of a function to lower case

2) Define functions:
   - Define shout function to return the input text. Apply @ uppercase_decorator to this function
   - Define whisper function to return the input text apply @ lowercase_decorator to this function

3. Define Greet function:

- Define greet function that:-
  - Accepts a function [func] as input
  - Calls this function with the text "Hi, I am created by a function passed as an argument"
  - 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_decorate(func):
    def wrapper(text):
        return func(text).upper()
    return wrapper

def lowercase_decorate(func):
    def wrapper(text):
        return func(text).lower()
    return wrapper

@uppercase_decorator
def Shout(text):
    return text

@lowercase_decorator
def whisper(text):
    return text
```

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

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

```
def greet(fun):
    greeting = fun("Hi, I am created, by a function passed 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
RECORD (5)	5
TOTAL (20)	15
SIGN WITH DATE	10/9/24

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