task: 8

Implement python generator and decodetors

Aim: write a python program to implement python generator and decorators.

8.1 Write a python program that includes a genarator function to produce a sequence of numbers.

a) Produce a sequence of numbers where provided with start, end and step values.

Algorithm:

1. Define generator function
   - ~~se~~ Define the function number -sequence.
2. initialize current value:
   - set current to the value of start
3. Generate sequence:
   - while current is less than or equal to end:
4. Get user input
   - read the starting number
   - read the ending number
   - read the step value
5. Create Generator object:
   - Create a Generator object by calling number_ sequence
6. Print Generated object
   - iterate over the values
   - Print each value.

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
```

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 :"))
sequence_generator = number_sequence (start, end, step)
for number in sequence_generator:
   print (number)
```

8-2) b

Algorithm

1. start function:
   - Define the function my_generator
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:
   - Print value.

Program

```
def my_generator(n):
    value = 0
    while value < n:
        yield value
        value += 1
for value in my_generator(3):
    Print (value)
```

8.2 Imagine you are working on a messaging application that needs to formal msg differently based on the users preference. user can choose to have their messages automatically converted to upper case to a lower case. You are provided with two decorators uppercase_decorator and lowercase_decorator.

Algorithm:

1. Create Decorators:
   - Define uppercase_decorator to convert the result of a function
   - Define lowercase_decorator to convert the result of a function
2. Define Greet function:
   - Accepts a function as input
   - Prints the result.
3. Execute the program
   - call greet (shout) to print greeting in upper case
   - call greet (whispter) to print in lower case.

output

```
0
1
2
```

Output

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

hi, i am created by a function Passed as an
argument
```

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()
    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.

VEL TECH

EX No.	8
PERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	5
VIVA VOCE (5)	5
RECORD (5)	5
TOTAL (20)	20
SIGN WITH DATE	[illegible] 17/9