Task 8.1 (a)

Aim :- To produce a sequence of numbers when provided with start, end & step values.

Algorithm :-

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

2. Initialize Current value :-
   - Set current to value of start
3. Generate Sequence
   - while current is less than or equal to end.
     - Yield the current value of current.
     - Increment corrent by step.
4. Get user Input :-
   - Read the string number(start) user input.
   - Read the ending number (end) user input.
   - Read the valule (step) from input.
5. Create generator object :-
   - create generator obj by calling number_ Sequence (star, end, stop)
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
    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, stop)
# print the generated sequence of number
for number in sequence_generator
    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

8.1 (b) Produce a default sequence of number Start from 0, ending at 10, with a step 1 if no value.

produce

Aim :- To produce a default sequence of number start from 0 ending 10.

Algorithm :-

1. Start Function :-
   - Define the function my-generator(n) then takes a parameter
2. Initialize Counter :-
   - set value to 0
3. Generate values :-
   - while value is less than n :
     - yeild the current value
     - increment value by 1.
4. Create a Generator Object :-
   - Call my generator(11) to create a generator obj.
5. Iterate & print values :-
   - For each value produced by generator object
     - print values.

program :-

```
def my-generator(n);
# initialize counter
  value=0
# loop with until counter is less than n
  while value<n:
# produce the current value of counter
  yield value
# increment the counter
  value += 1
# iterater over the generator object produced by
  my-generator for value in my-generator(5):
# print each value produced by generator:
  print(value).
```

output:

Task : 8.2

Image you are working on a messaging application that needs to format message differently based

Aim: To work a messaging application that needs to format message.

Algorithm:

1. Create Decorders:
   - Define uppercase_decaiator to convert the result of a function to uppercase.
   - Define lower-case decarator to convert result to lowercase.
2. Define functions:
   - Define shout function to return the input. Apply @ uppercase-decarator to this function.
   - Define whis per function to return the input
3. Define Greet Function:
   - Define great function that
     - Accepts a function as input
     - Calls this function with the test "H.I on created by a function.

program:

```
def uppercase-decarator (func);
  def wrapper(text);
    return func(text) upper()
  return wrapper
def lowercase- decarator (func);
  def wrapper(text);
    return func (text):lower()
return wrapper
@ uppercase_decarator
  def Shout (text)
    return text
@ lowercase_decaiator
  def wispher (text);
    return text
  def greet (tune):
  greeting = func ("Hi, Iam created by a function
                                passed or argument")
```

Output:

HI, I AM CREATED BY FUNCTION PASSED AS AN ARGUMENT.

hi, iam created by a function passed as an argument.

greet (shout)
greet (whisper)

VEL TECH - CSE	
EX NO.	9
PERFORMANCE (5)	5
RESULT AND ANALYSIS (3)	3
VIVA VOCE (3)	3
RECORD (4)	4
TOTAL (15)	
SIGN WITH DATE	15

Result: Thus the python program to implement python generator and decarators was successfully executed & output was verified.