TASK 8 : Implement Python generators and decorators

24/9/25

AIM : To write a python program that implements a generator to produce the squares of numbers upto a given limit

Algorithm

1. Start the program
2. Define a generator function using the def keyword
3. Inside the function, use a loop from 1 to n.
4. use the yield statement to return the square of each number one by one.
5. in the main Program, accept a number n from user.
6. call the generator function and iterate through it using a for loop.
7. Print squares generated
8. End the Program.

Sample I/O

```
Enter a number : 5
Squares from 1 to 5 are:

1
4
9
16
25
```

Program

```
def square_generator(n):
    for i in range(1, n+1):
        yield i*i #yield returns values one by one
 n=int(input("Enter a number:"))
Print(f"squares from 1 to {n} are:")
for val in square_generator(n):
    Print(val)
```

TASK 8.2

AIM : To write a Python Program that implements a decorator to Calculate and display the execution time of function

Algorithm :

1. Start the Program
2. import time module
3. Define a decorator function that accepts another function as an argument.
4. Inside the decorator, define a wrapper function:
   - Record the start time.
   - Call the original function
   - Record the end time
   - Print the execution time
5. Return the wrapper function from decorator.
6. use the @decorator_name syntax to apply the decorator to a function.
7. Define a function (e.g. display_numbers) that decorator to a function.
8. Call the decorator function
9. End program

Program

```
#Decorator function
def timer_decorator(func):
    def wrapper():
        Start = time.time()
      func()
      end = time.time()
    Print(f"execution Time:{end-start:5f} seconds")
  return wrapper
# Function to be decorated
@timer_decorator
def display_numbers():
    for i in range(1,6):
        Print(i)
    time.sleep(0.5)  # just to Simulate delay
# Calling the decorated function
display_numbers()
```

Sample I/O

```
1
2
3
4
5
```

Program :

```
def  fibonacci(n):
     a, b = 0, 1
     for _ in range(n):
         yield a
        a, b = b, a+b
   for num in fibonacci(10):
       Print(num, end = " ")
```

TASK 8.3 : Fibonacci sequence

AIM : To design a Python Program that implements a generator function fibonacci(n) which yields the first n fibonacci numbers.

Algorithm

1. start
2. Define a generator function fibonacci(n) that takes an integer n as input
3. initialize two variables:

   a = 0
   b = 1

4. Repeat the following steps for n iterations
5. outside the function, call generator using for num in fibonacci(n):
6. print each fibonacci number as it is generated
7. End

Output:

```
0 1 1 2 3 5 8 13 21 34
```

VELTECH	
EX No.	8
PERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	5
VIVA VOCE (5)	5
RECORD (5)	5
TOTAL (20)	20
SIGN WITH DATE	

RESULT: Thus, the Python decorators has been Executed Successfully.