TASK: 8
DATE: 24/9/25

IMPLEMENT PYTHON GENERATORS AND DECORATORS.

8.1 : Produce the squares of numbers up to a limit.

AIM:-

To write a Python that implements a generator to produce the squares of numbers up to a given limit.

ALGORITHM :-

1) Start the program
2) Define a generator function, use a loop from 1 to n
3) Use the yield statement to return the square of each number one by one
4) In the main program, accept a number n from the user.
5) Call the generator function & iterate through it using a for loop.
6) Print the squares generated
7) End the program.

Sample I/O :-

Enter a number : 5
Squares from 1 to 5 are
1
4
9
16
25

PROGRAMS:-

```
def square_generator(n):
    for i in range(1, n+1):
        yield i*i

n = int(input("Enter a number:"))
print(f"Squares from 1 to {n} are:")
for val in square_generator(n):
    print(val)
```

8.2 :- To Show execution time of a function

Aim:-

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

Algorithm :-

1) Start
2) Import the time table
3) Define a decorator function that accepts another function as an argument.
4) Inside the decorator, define a wrapper function :-
   * Record start time.
   * Call the original function.
   * 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 that prints numbers with a small delay
8) Call the decorated function
9) End

Sample I/O:-

1
2
3
4
5

Execution time : 2.50012 seconds.

PROGRAM:-

```
import time
def timer_decorator(func):
    def wrapper():
        start = time.time()
        func()
        end = time.time()
        print(f"Execution Time: {end - start:.5f}
        seconds")
    return wrapper

@timer_decorator
def display_numbers():
    for i in range(1, 6):
        print(i)
        time.sleep(0.5)
display_numbers()
```

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 :-

Start
Define a generator function fibonacci(n) that takes an integer n as input.
Initialize two variables:

a = 0
b = 1

Repeat the following steps for n iteration
yield the current value of a
Update values:

a = b
b = a + b

Outside the function, call the generator using for num in fibonacci(n):.
Print each Fibonacci number as it is generated
End

I/O

0 1 1 2 3 5 8 13 21 34

VERIFICATION	
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 Program to Implement Python generator & Decorators are verified & executed successfully.

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 = " ")
```