**Task 8 a:**

**Program**

def square\_generator(n):

for i in range(1, n+1):

yield i \* i # yield returns values one by one

# Using the generator

n = int(input("Enter a number: "))

print(f"Squares from 1 to {n} are:")

for val in square\_generator(n):

print(val)

**Sample I/O**

Enter a number: 5

Squares from 1 to 5 are:

1

4

9

16

25

**Task 8b:**

**Program**

mport time

# 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

Execution Time: 2.50012 seconds

**Task 8 c:**

**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=" ")

**Output**

0 1 1 2 3 5 8 13 21 34