17/9/25

Task 8: Implement python generator and decorators.

8a: Fibonacci sequence generator.

Aim: To implement a generator function to yield fibonacci numbers up to a given limit and display the sequence

Algorithm:

- Start the program
- Define a generator function fibonacci(n)
  - initializes a=0, b=1
  - yields fibonacci numbers while a<=n
- Accept the number n from the user
- Call the generator function and print all fibonacci numbers using the loop
- End the program

Program:-

```
def fibonacci(n):
a,b=0,1
while a<=n:
  yield a
  a,b=b,a+b

n=int(input("Enter the limit for Fibonacci sequence:"))
print(f"\n fibonacci sequence up to {n}:")
for num in fibonacci(n)
  print(num, end=" ")
  print()
```

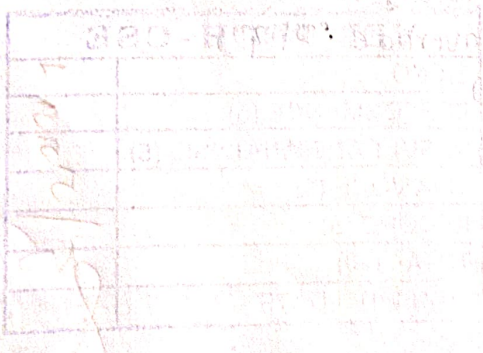
Result: Thus the fibonacci sequence generator of the program Executed Successfully.

Output

Enter the limit for fibonacci sequence: 50

fibonacci sequence upto 50:

0 1 1 2 3 5 8 13 21 34.



8b: Function Execution Time decorator.

17/9/25

Aim: The implements to a decorator to calculate the Execution time of any function and apply it to a function that sort a list of random numbers.

Algorithm:

1. Start the program
2. Import the time and random modules
3. Records the start time before calling the function
   - Executes the function
   - Displays the time taken for execution
4. Define a function Sort_numbers() that
   - generates a list of random numbers
5. call the sort_numbers() function
6. End the program

Program:

```
import time
import random
def execution_time(func):
def wrapper(*args,**kwargs):
start=time.time()
result=func(*args,**kwargs)
end=time.time()
print(f"\nExecution Time: {end-start:.6f} seconds")
return result
return wrapper.
```

VEL TECH	
EX NO.	8
PERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	5
VIVA VOCE (5)	5
RECORD (5)	5
TOTAL (20)	10
SIGN WITH DATE	

Result:-

Thus the function Execution time decorator program Executed Successfully

Output :-

Sorting random numbers
Sorting completed
Execution time : 0.003211 seconds.