SR UNIVERSITY

School of Computer Science and Artificial Intelligence

Lab Assignment # 2

Program	**: B. Tech (CSE)**
Specialization	**:**
Course Title	**: AI Assisted coding**
Course Code	**: 23CS201PC302**
Semester	**: II**
Academic Session	**: 2025-2026**
Name of Student	**: Mohammed Hashir Sayam Hussain**
Enrollment No.	**: 2403A51L27**
Batch No.	**: 51**
Date	**:09-01-2026**

Submission Starts here

OUTPUT :
SCREENSHOTS:

Task1: without using functions

Prompt: `# Fibonacci number series without using functions`

Code Screenshot:

```
# Fibonacci number series without using functions
n = 10  # Number of terms
a, b = 0, 1

print("Fibonacci Series:")
for i in range(n):
    print(a, end=" ")
    a, b = b, a + b
```

0.0s

Output Screenshot:

```
Fibonacci Series:
0 1 1 2 3 5 8 13 21 34
```

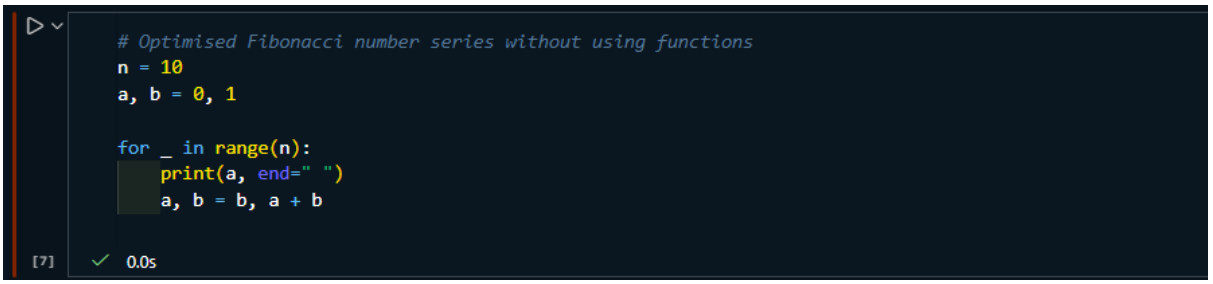
Explanation of code:

- The program prints the Fibonacci series up to 10 numbers.
- It starts with `0` and `1`, which are the first two Fibonacci values.
- In each loop, the current number is printed.
- Then the next number is created by adding the previous two numbers.
- This repeats until all 10 numbers are printed.

Task 2: Optimised Code

Prompt:`# Optimised Fibonacci number series without using functions`

Code Screenshot:



```
# Optimised Fibonacci number series without using functions
n = 10
a, b = 0, 1

for _ in range(n):
    print(a, end=" ")
    a, b = b, a + b
```

[7] ✓ 0.0s

Output Screenshot:

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

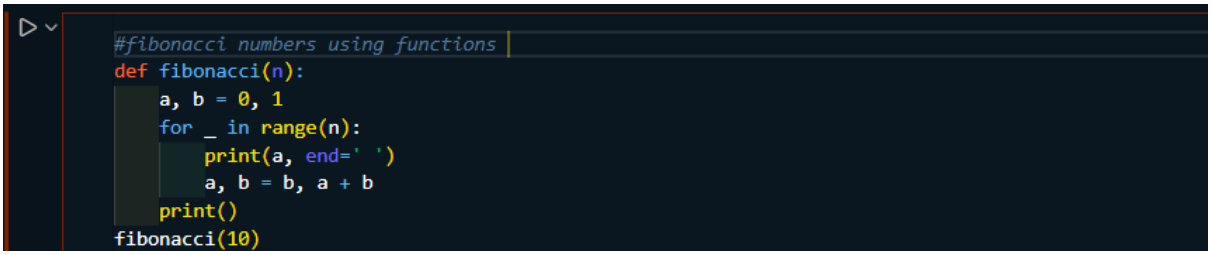
Explanation of code:

- It uses **constant memory (O(1))** — only two variables are used.
- No extra lists or arrays are created.
- Python's **tuple assignment** updates values in a single step, making it fast and clean.

Task 3:with using functions

Prompt:`# Fibonacci number series with using functions`

Code Screenshot:



```
#fibonacci numbers using functions
def fibonacci(n):
    a, b = 0, 1
    for _ in range(n):
        print(a, end=' ')
        a, b = b, a + b
    print()
fibonacci(10)
```

Output Screenshot:

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

Explanation of code:

- The function `fibonacci(n)` prints the first `n` Fibonacci numbers.
- `a` and `b` store the last two numbers of the series.
- The loop runs `n` times to generate `n` values.
- Each iteration prints `a`.
- Then the next Fibonacci number is calculated by `a + b`.
- Tuple assignment updates both values efficiently in one step

SR UNIVERSITY

Task4:with and without using functions

Prompt:*# Fibonacci number series with vs without using functions*

Code Screenshot:

```python
#fibonacci numbers using functions vs without functions
def fibonacci(n):
    a, b = 0, 1
    print("Fibonacci series using functions:")
    for _ in range(n):
        print(a, end=' ')
        a, b = b, a + b
    print()
fibonacci(10)
# Fibonacci number series without using functions
n = 10  # Number of terms
a, b = 0, 1
print("Fibonacci series without functions:")
for i in range(n):
    print(a, end=" ")
    a, b = b, a + b
```

[9] ✓ 0.0s

Output Screenshot:

```
Fibonacci series using functions:
0 1 1 2 3 5 8 13 21 34
Fibonacci series without functions:
0 1 1 2 3 5 8 13 21 34
```

Explanation of code:

Using functions

- The Fibonacci logic is **wrapped inside a function**.
- `n` is passed as a parameter, so the function can be reused for any value.
- Code is **organized, modular, and reusable**.
- Best for **large programs and real projects**.

Without using functions

- The logic is written **directly in the program**.
- Simple and easy to understand for beginners.
- Not reusable; code must be rewritten if needed again.
- Suitable for **small programs or learning basics**.

SR UNIVERSITY

Task5: Using recursion

Prompt:*# #fibonacci numbers using recursion*

Code Screenshot:

```
#fibonacci numbers using recursion
def fibonacci_recursive(n, a=0, b=1):
    if n == 0:
        return
    print(a, end=' ')
    fibonacci_recursive(n - 1, b, a + b)

fibonacci_recursive(10)
```

[10] ✓ 0.0s

Output Screenshot:

```
Fibonacci Series:
0 1 1 2 3 5 8 13 21 34
```

Explanation of code:

- The function prints the current Fibonacci number.
- It calls itself with updated values.
- Stops when `n` becomes `0`.
- No loops are used → **pure recursion**.

SR UNIVERSITY