

Lab 1.3 – AI Assisted Coding (Fibonacci Tasks)

Name: Mohammad Muneer Ahmed

Roll No: 2303A51475

Task 1 – Fibonacci Without Functions (Procedural)

Algorithm:

1. Read integer n
2. Initialize a=0, b=1
3. Handle base cases
4. Loop and compute next = a+b
5. Update variables

Pseudocode:

```
INPUT n
a←0,b←1
FOR i from 0 to n
PRINT a
a,b←b,a+b
```

The screenshot shows a code editor window with a dark theme. On the left is a vertical scroll bar. The main area contains the following Python code:

```
day5.py > ...
1  # Fibonacci without using user-defined functions
2
3  n = int(input("Enter number of terms: "))
4
5  a = 0
6  b = 1
7
8  if n <= 0:
9      print("Invalid input")
10 elif n == 1:
11     print(a)
12 else:
13     print(a, b, end=" ")
14     for _ in range(2, n):
15         c = a + b
16         print(c, end=" ")
17         a = b
18         b = c
19
```

Below the code, there are several tabs: PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, and PORTS. The TERMINAL tab is selected. It displays the following terminal session:

```
python -u "/Users/mohammadmuneerahmed/Documents/training2.py/day5.py"
● mohammadmuneerahmed@Muneers-MacBook-Air training2.py % python -u "/Users/mohammadmuneerahmed/Documents/training2.py/day5.py"
Enter number of terms: 0
Invalid input
● mohammadmuneerahmed@Muneers-MacBook-Air training2.py % python -u "/Users/mohammadmuneerahmed/Documents/training2.py/day5.py"
Enter number of terms: 34
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765 10946 17711 28657 46368 75025 121393 196418 317811 514229 832040 1346269 2178309 3524578 5702887 9227465 14938352 24157817
39088169 63245986 102334155 165580141 267914296 433494437 701408733 1134903170 1836311903 2971215073 4807526976 7778742049 12586269025 20365011074 32951280099 53316291173
◇ mohammadmuneerahmed@Muneers-MacBook-Air training2.py %
```

Task 2 – Optimized Fibonacci

Algorithm:

1. Read n
2. Use tuple swap update
3. Print each term

Pseudocode:

```
INPUT n
a,b←0,1
REPEAT n times
PRINT a
a,b←b,a+b
```

```
day5.py > ...
1 # Optimized Fibonacci without redundant variables
2
3 n = int(input("Enter number of terms: "))
4
5 a, b = 0, 1
6
7 for _ in range(n):
8     print(a, end=" ")
9     a, b = b, a + b
10

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS
python -u "/Users/mohammadmuneerahmed/Documents/training2.py/day5.py"
● mohammadmuneerahmed@Muneers-MacBook-Air training2.py % python -u "/Users/mohammadmuneerahmed/Documents/training2.py/day5.py"
Enter number of terms: 4
0 1 1 2 %
↳ mohammadmuneerahmed@Muneers-MacBook-Air training2.py %
```

Task 3 – Modular Fibonacci Using Function

Algorithm:

1. Define function
2. Loop and append
3. Return sequence

Pseudocode:

FUNCTION fib(n)

a,b←0,1

list←[]

LOOP n

append a

a,b←b,a+b

RETURN list

The screenshot shows a code editor window with a dark theme. At the top, there's a file icon followed by "day5.py > ...". Below the code area, there are tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL (which is underlined), and PORTS.

```
day5.py > ...
1  # Fibonacci using modular design
2
3  def fibonacci(n):
4      seq = []
5      a, b = 0, 1
6      for _ in range(n):
7          seq.append(a)
8          a, b = b, a + b
9      return seq
10
11
12 n = int(input("Enter number of terms: "))
13 print(*fibonacci(n))
14
```

Below the code, the terminal output is displayed:

```
python -u "/Users/mohammadmuneerahmed/Documents/training2.py/day5.py"
mohammadmuneerahmed@Muneers-MacBook-Air training2.py % python -u "/Users/mohammadmuneerahmed/Documents/training2.py/day5.py"
Enter number of terms: 6
0 1 1 2 3 5
mohammadmuneerahmed@Muneers-MacBook-Air training2.py %
```

Task 4 – Procedural vs Modular Comparison

Algorithm:

1. Evaluate reuse
2. Evaluate clarity
3. Evaluate debugging

Pseudocode:

COMPARE procedural vs modular

REPORT differences

Procedural Fibonacci code is acceptable for quick scripts and demonstrations but fails in reuse, testing, and maintainability. Modular Fibonacci code separates concerns, supports reuse, simplifies debugging, and scales to larger applications.

Task 5 – Iterative vs Recursive Fibonacci

Algorithm:

Iterative loop method

Recursive base+call method

Pseudocode:

ITERATIVE loop update

RECURSIVE fib(n-1)+fib(n-2)

```
day5.py > ...
1  def fib_recursive(n):
2      if n <= 1:
3          return n
4      return fib_recursive(n-1) + fib_recursive(n-2)
5
6
7  terms = int(input("Enter number of terms: "))
8  for i in range(terms):
9      print(fib_recursive(i), end=" ")
10 
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
python -u "/Users/mohammadmuneerahmed/Documents/training2.py/day5.py"
● mohammadmuneerahmed@Muneers-MacBook-Air training2.py % python -u "/Users/mohammadmuneerahmed/Documents/training2.py/day5.py"
Enter number of terms: 5
0 1 1 2 3 %
◇ mohammadmuneerahmed@Muneers-MacBook-Air training2.py %
```