

Lab Assignment – 01

Name:G.Harika

HT.NO:2303A51612

Batch:05

Task-01:

Prompt:

Find Fibanocci series upto n terms without using user defined functions

Code:

```
n = int(input("Enter the number of terms: "))

a, b = 0, 1
count = 0

if n <= 0:
    print("Please enter a positive integer")

elif n == 1:
    print("Fibonacci sequence upto", n, "term:")
    print(a)

else:
    print("Fibonacci sequence upto", n, "terms:")
    while count < n:
        print(a, end=' ')
```

$c = a + b$

$a = b$

$b = c$

$\text{count} += 1$

Output:

Enter the number of terms: 5

Fibonacci sequence upto 5 terms:

0 1 1 2 3

Implementation:

1. The program asks for n and starts the Fibonacci series with $a = 0$ and $b = 1$.
2. If n is 0 or negative, it shows an error; if $n == 1$, it prints only 0.
3. Otherwise, it runs a while loop, printing a each time and updating the values using $c = a + b$ then $a = b$, $b = c$ and count increases by 1.
4. The loop stops after printing n numbers.

The screenshot shows a code editor window with a dark theme. At the top, there's a tab labeled 'Fibonacci.py X'. Below the tabs, there's a toolbar with icons for file operations. The main area contains Python code for generating a Fibonacci sequence. The code uses variables 'a' and 'b' to store the current and next numbers in the sequence, respectively. It starts by checking if the input 'n' is less than or equal to 0, printing an error message if so. If 'n' is 1, it prints the sequence up to that term. For n > 1, it enters a loop where it prints 'a', adds 'a' and 'b' to get 'c', then updates 'a' to 'b' and 'b' to 'c', and increments the count. The code ends with a closing brace for the else block.

```
❶ Fibonacci.py X
❷ Fibonacci.py > -
1 #Find Fibonacci series upto n terms without using user defined functions
2 n = int(input("Enter the number of terms: "))
3 a, b = 0, 1
4 count = 0
5 if n <= 0:
6     print("Please enter a positive integer")
7 elif n == 1:
8     print("Fibonacci sequence upto", n, "term:")
9     print(a)
10 else:
11     print("Fibonacci sequence upto", n, "terms:")
12     while count < n:
13         print(a, end=' ')
14         c = a + b
15         a = b
16         b = c
17         count += 1
18     |
```

Below the code editor, there's a terminal window showing the execution of the script. It starts with the command 'PS C:\Users\GAJE HARIKA\Desktop\AI Coding> & "C:/Users/GAJE HARIKA/AppData/Local/Python/pythoncore-3.14-64/python.exe" "c:/Users/GAJE HARIKA/Desktop/AI Coding/Fibonacci.py"'. The user then enters '5' when prompted for the number of terms. The script outputs 'Fibonacci sequence upto 5 terms:' followed by the sequence '0 1 1 2 3'. Finally, the terminal shows the prompt 'PS C:\Users\GAJE HARIKA\Desktop\AI Coding>' again.

Task-02:

Prompt:

Find Fibonacci series upto n terms optimized code without using functions

Code:

```
n = int(input("Enter the number of terms: "))
```

```
a, b = 0, 1
```

```
if n <= 0:
```

```
    print("Please enter a positive integer")
```

```
elif n == 1:
```

```
    print("Fibonacci sequence upto", n, "term:")
```

```
print(a)
else:
    print("Fibonacci sequence upto", n, "terms:")
    for _ in range(n):
        print(a, end=' ')
        a, b = b, a + b
```

Output:

Enter the number of terms: 7

Fibonacci sequence upto 7 terms:

0 1 1 2 3 5 8

Implementation:

1. The program takes n from the user and starts with $a = 0$ and $b = 1$.
2. If n is not positive, it shows an error message.
3. Otherwise, a for loop runs n times, printing a each time.
4. Inside the loop, the next Fibonacci number is generated using $a, b = b, a + b$.

The screenshot shows a code editor interface with a dark theme. At the top, there are two tabs: 'Fibonacci.py' and 'Fibonacci.py ...'. The code in the editor is as follows:

```
20  #Find Fibonacci series upto n terms optimized code without using functions
21  n = int(input("Enter the number of terms: "))
22  a, b = 0, 1
23  if n <= 0:
24      print("Please enter a positive integer")
25  elif n == 1:
26      print("Fibonacci sequence upto", n, "term:")
27      print(a)
28  else:
29      print("Fibonacci sequence upto", n, "terms:")
30      for _ in range(n):
31          print(a, end=' ')
32          a, b = b, a + b
33
34
35
```

Below the code editor is a terminal window showing the execution of the script:

```
PS C:\Users\GAJE HARIKA\OneDrive\Desktop\AI Coding> & "C:/Users/GAJE HARIKA/AppData/Local/Python/pythoncore-3.14-64/python.exe" "C:/Users/GAJE HARIKA/OneDrive/Desktop/AI Coding/Fibonacci.py"
Enter the number of terms: 7
Fibonacci sequence upto 7 terms:
0 1 1 2 3 5 8
PS C:\Users\GAJE HARIKA\OneDrive\Desktop\AI Coding>
```

At the bottom of the terminal window, it says 'Ln 35, Col 1 Spaces: 4 UTF-8 CRLF [] Python'.

Task-03:

Prompt:

Find Fibonacci series using Functions returns or prints the sequence upto n terms

Code:

```
def fibonacci_series(n):
```

```
    a, b = 0, 1
```

```
    series = []
```

```
    for _ in range(n):
```

```
series.append(a)
a, b = b, a + b
return series

n = int(input("Enter the number of terms: "))
if n <= 0:
    print("Please enter a positive integer")
elif n == 1:
    print("Fibonacci sequence upto", n, "term:")
    print(fibonacci_series(n))
else:
    print("Fibonacci sequence upto", n, "terms:")
    print(fibonacci_series(n))
```

Output:

Enter the number of terms: 7

Fibonacci sequence upto 7 terms:

[0, 1, 1, 2, 3, 5, 8]

Implementation:

1. The function starts with 0 and 1 as the first Fibonacci numbers.
2. It repeats n times to create the next numbers.
3. Each number is added to a list.
4. The list is returned and printed as the Fibonacci sequence.

```
❸ Fibanocci.py X
❹ Fibanocci.py > ...

34  #use user defined function to generate Fibonacci series up to n terms
35  #print the sequence with comments
36  def fibonacci_series(n):
37      a, b = 0, 1
38      series = []
39      for _ in range(n):
40          series.append(a)
41          a, b = b, a + b
42      return series
43  n = int(input("Enter the number of terms: "))
44  if n <= 0:
45      print("Please enter a positive integer")
46  elif n == 1:
47      print("Fibonacci sequence upto", n, "term:")
48      print(fibonacci_series(n))
49  else:
50      print("Fibonacci sequence upto", n, "terms:")
51      print(fibonacci_series(n))
52
53

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Python + ~ ⌂ ⌂ ... | ⌂ ×
PS C:\Users\GAJE HARIBA\OneDrive\Desktop\AI Coding> & "c:/users/gaje harika/appdata/local/python/pythoncore-3.14-64/python.exe" "c:/users/gaje harika/onedrive/desktop/ai coding/Fibanocci.py"
Enter the number of terms: 7
Fibonacci sequence upto 7 terms:
[0, 1, 1, 2, 3, 5, 8]
PS C:\Users\GAJE HARIBA\OneDrive\Desktop\AI Coding> []

Create Commit Message
In 32, Col 24  Spaces: 4  UTF-8  CR/LF  { } Python  ⌂
```

Task-04:

Prompt:

Fibonacci series with Procedural vs Modular Fibonacci Code
AI code with and without

Procedural approach

Code:

```
def fibonacci_series(n):
```

```
    a, b = 0, 1
```

```
    series = []
```

```
for _ in range(n):
    series.append(a)
    a, b = b, a + b
return series

n = int(input("Enter the number of terms: "))

if n <= 0:
    print("Please enter a positive integer")
elif n == 1:
    print("Fibonacci sequence upto", n, "term:")
    print(fibonacci_series(n))
else:
    print("Fibonacci sequence upto", n, "terms:")
    print(fibonacci_series(n))
```

Output:

Enter the number of terms: 5

Fibonacci sequence upto 5 terms:

[0, 1, 1, 2, 3]

Prompt:

Modular approach

Code:

```
def get_fibonacci_series(n):
```

```
a, b = 0, 1
series = []
for _ in range(n):
    series.append(a)
    a, b = b, a + b
return series

def main():
    n = int(input("Enter the number of terms: "))
    if n <= 0:
        print("Please enter a positive integer")
    elif n == 1:
        print("Fibonacci sequence upto", n, "term:")
        print(get_fibonacci_series(n))
    else:
        print("Fibonacci sequence upto", n, "terms:")
        print(get_fibonacci_series(n))

if __name__ == "__main__":
    main()
```

Output:

Enter the number of terms: 7

Fibonacci sequence upto 7 terms:

[0, 1, 1, 2, 3, 5, 8]

Implementation:

The screenshot shows a code editor with a dark theme and a terminal window below it.

Code Editor Content (Fibonacci.py):

```
◆ Fibonacci.py ×
❶ # Modular approach
❷ def get_fibonacci_series(n):
❸     a, b = 0, 1
❹     series = []
❺     for _ in range(n):
❻         series.append(a)
❼         a, b = b, a+b
⫽     return series
⫾ def main():
⫿     n = int(input("Enter the number of terms: "))
⫿     if n <= 0:
⫿         print("Please enter a positive integer")
⫿     elif n == 1:
⫿         print("Fibonacci sequence upto", n, "term:")
⫿         print(get_fibonacci_series(n))
⫿     else:
⫿         print("Fibonacci sequence upto", n, "terms:")
⫿         print(get_fibonacci_series(n))
⫾ if __name__ == "__main__":
⫾     main()
```

Terminal Output:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\GAJE HARIKA\OneDrive\Desktop\AI Coding> & "c:/Users/GAJE HARIKA/AppData/Local/Python/pythoncore-3.14-64/python.exe" "c:/Users/GAJE HARIKA/OneDrive/Desktop/AI Coding/Fibonacci.py"
Enter the number of terms: 7
Fibonacci sequence upto 7 terms:
[0, 1, 1, 2, 3, 5, 8]
PS C:\Users\GAJE HARIKA\OneDrive\Desktop\AI Coding>
```

Task-05:

Prompt:

#AI-Generated Iterative vs Recursive Fibonacci Approaches
(Different Algorithmic Approaches for Fibonacci Series)

Iterative approach

Code:

```
def fibonacci_iterative(n):
```

```
    a, b = 0, 1
```

```
series = []
for _ in range(n):
    series.append(a)
    a, b = b, a + b
return series

n = int(input("Enter the number of terms: "))
if n <= 0:
    print("Please enter a positive integer")
elif n == 1:
    print("Fibonacci sequence upto", n, "term:")
    print(fibonacci_iterative(n))
else:
    print("Fibonacci sequence upto", n, "terms:")
    print(fibonacci_iterative(n))
```

Code:

Enter the number of terms: 8

Fibonacci sequence upto 8 terms:

[0, 1, 1, 2, 3, 5, 8, 13]

Implementation:

- 1.The function is defined to generate Fibonacci numbers.
2. a and b are initialized to 0 and 1.
3. A loop runs n times.

4. Each Fibonacci number is added to a list.

5. The list is returned and printed.

```
 93 #AI-Generated Iterative vs Recursive Fibonacci Approaches (Different
 94 #Algorithmic Approaches for Fibonacci Series)
 95 # Iterative approach
 96 def fibonacci_iterative(n):
 97     a, b = 0, 1
 98     series = []
 99     for _ in range(n):
100         series.append(a)
101         a, b = b, a + b
102     return series
103 n = int(input("Enter the number of terms: "))
104 if n <= 0:
105     print("Please enter a positive integer")
106 elif n == 1:
107     print("Fibonacci sequence upto", n, "term:")
108     print(fibonacci_iterative(n))
109 else:
110     print("Fibonacci sequence upto", n, "terms:")
111     print(fibonacci_iterative(n))
112
113
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\GAJE HARIKA\OneDrive\Desktop\AI Coding> & "C:/Users/GAJE HARIKA/AppData/Local/Python/pythoncore-3.14-64/python.exe" "c:/Users/GAJE HARIKA/OneDrive/Desktop/AI Coding/Fibonacci.py"
Enter the number of terms: 8
Fibonacci sequence upto 8 terms:
[0, 1, 1, 2, 3, 5, 8, 13]
PS C:\Users\GAJE HARIKA\OneDrive\Desktop\AI Coding>

Prompt:

Recursive approach

Code:

```
def fibonacci_recursive(n):
    if n <= 0:
        return []

```

```
elif n == 1:  
    return [0]  
  
elif n == 2:  
    return [0, 1]  
  
else:  
    series = fibonacci_recursive(n - 1)  
    series.append(series[-1] + series[-2])  
    return series  
  
n = int(input("Enter the number of terms: "))  
  
if n <= 0:  
    print("Please enter a positive integer")  
  
elif n == 1:  
    print("Fibonacci sequence upto", n, "term:")  
    print(fibonacci_recursive(n))  
  
else:  
    print("Fibonacci sequence upto", n, "terms:")  
    print(fibonacci_recursive(n))
```

Output:

Enter the number of terms: 6

Fibonacci sequence upto 6 terms:

[0, 1, 1, 2, 3, 5]

Implementation:

1. The function `fibonacci_recursive(n)` is called.
2. If n is 1 or 2, it returns the base Fibonacci values.
3. If n is greater than 2, the function calls itself with $n-1$.
4. The last two numbers are added to get the next Fibonacci number.
5. The updated list is returned and printed.

The screenshot shows a code editor window with a dark theme. The file is named `Fibonacci.py`. The code implements a recursive approach to generate a Fibonacci sequence. It starts by defining a function `fibonacci_recursive(n)` which handles the base cases for $n=0$, $n=1$, and $n=2$. For $n > 2$, it calls itself with $n-1$ and adds the last two elements of the resulting list. The main part of the script prompts the user for the number of terms, checks if it's a positive integer, and then prints the sequence up to that term count. The terminal below shows the execution of the script and its output for 6 terms.

```
#Recursive approach
def fibonacci_recursive(n):
    if n <= 0:
        return []
    elif n == 1:
        return [0]
    elif n == 2:
        return [0, 1]
    else:
        series = fibonacci_recursive(n - 1)
        series.append(series[-1] + series[-2])
        return series

n = int(input("Enter the number of terms: "))
if n <= 0:
    print("Please enter a positive integer")
elif n == 1:
    print("Fibonacci sequence upto", n, "term:")
    print(fibonacci_recursive(n))
else:
    print("Fibonacci sequence upto", n, "terms:")
    print(fibonacci_recursive(n))

PS C:\Users\GAJE HARIBA\OneDrive\Desktop\AI Coding> & "C:/Users/GAJE HARIBA/AppData/Local/Python/pythoncore-3.14-64/python.exe" "c:/Users/GAJE HARIBA/OneDrive/Desktop/AI Coding/Fibonacci.py"
Enter the number of terms: 6
Fibonacci sequence upto 6 terms:
[0, 1, 1, 2, 3, 5]
PS C:\Users\GAJE HARIBA\OneDrive\Desktop\AI Coding>
```

