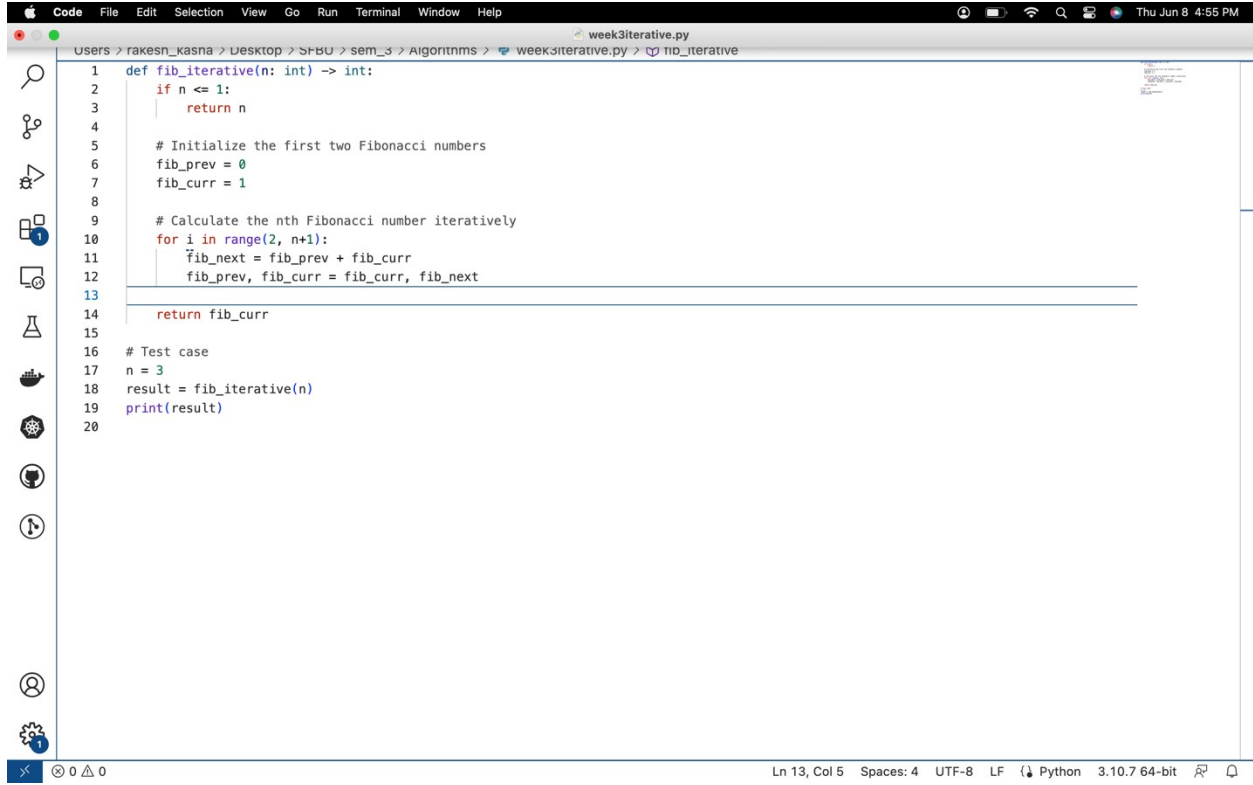


Homework3_q1

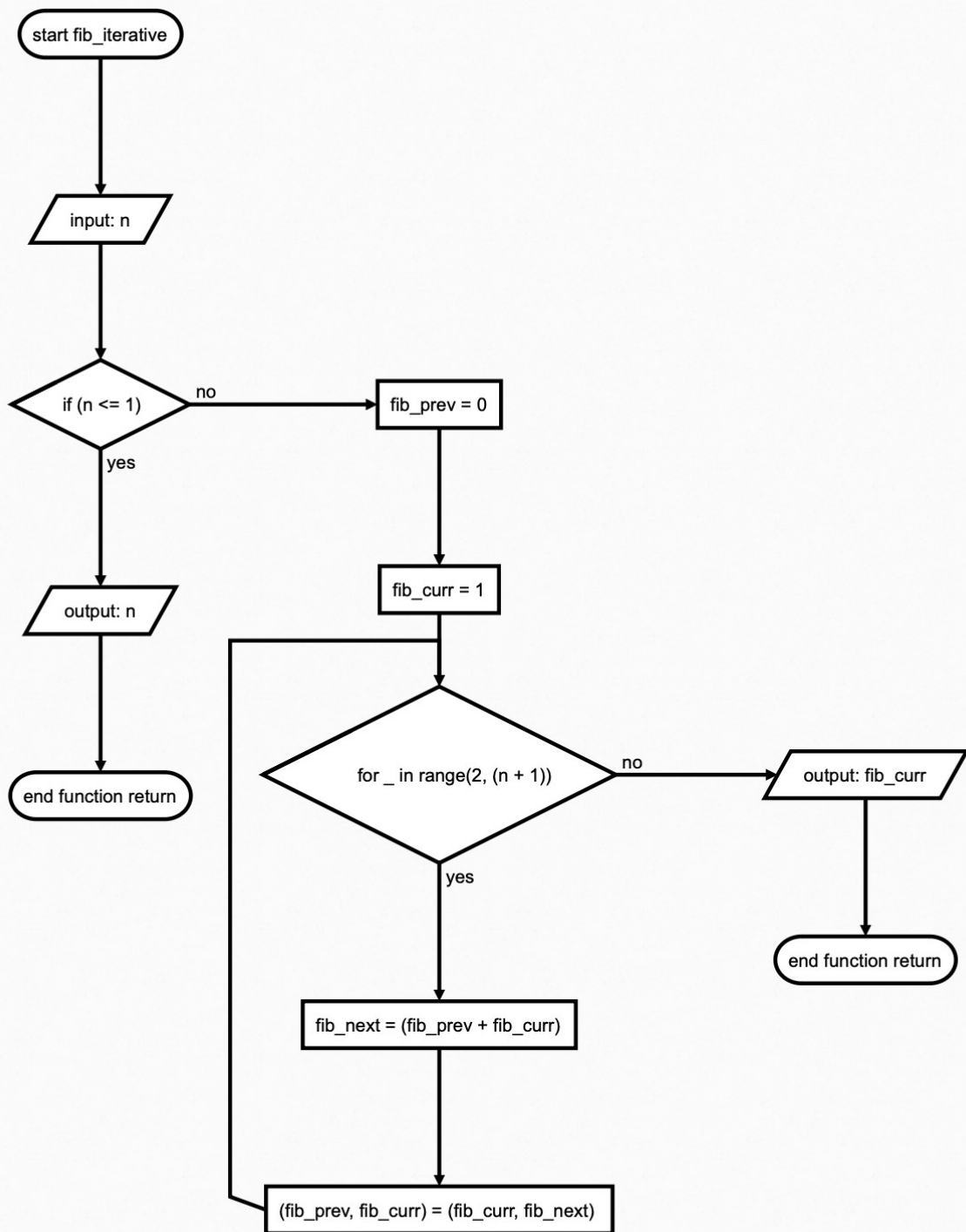
Iterative:



The screenshot shows a code editor window with a menu bar (Code, File, Edit, Selection, View, Go, Run, Terminal, Window, Help) and a status bar at the bottom. The status bar indicates the current position is Line 13, Column 5, with 4 spaces, UTF-8 encoding, LF line endings, Python 3.10.7 64-bit, and a file icon. The code is as follows:

```
1 def fib_iterative(n: int) -> int:
2     if n <= 1:
3         return n
4
5     # Initialize the first two Fibonacci numbers
6     fib_prev = 0
7     fib_curr = 1
8
9     # Calculate the nth Fibonacci number iteratively
10    for i in range(2, n+1):
11        fib_next = fib_prev + fib_curr
12        fib_prev, fib_curr = fib_curr, fib_next
13
14    return fib_curr
15
16 # Test case
17 n = 3
18 result = fib_iterative(n)
19 print(result)
20
```

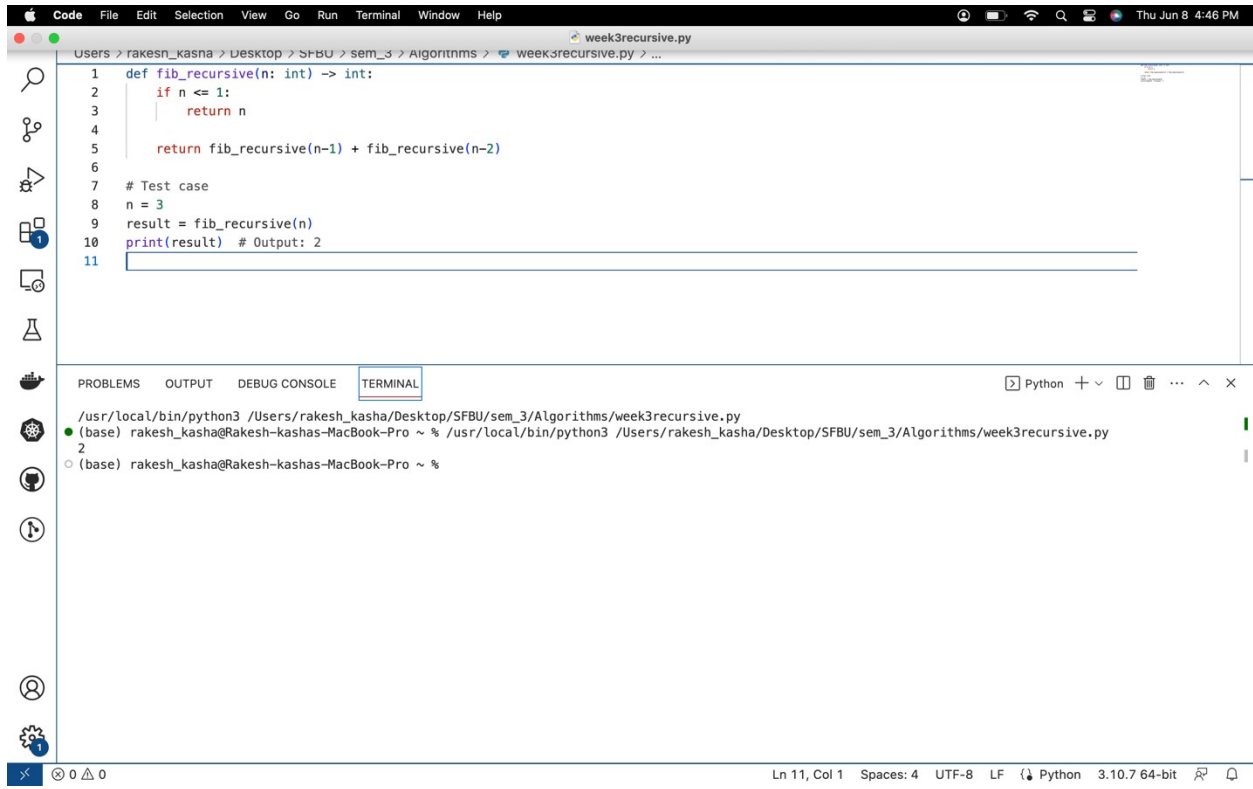
Flowchart:



Trace table:

Step	n	$(n \leq 1)$	fib_prev	fib_curr	i	fib_next	return
1	3						
2		False					
3			0				
4				1			
5					2		
6						0+1	
7			1	1			
8					3		
9						1+1=2	
10			1	2			
11							2

Recursive:



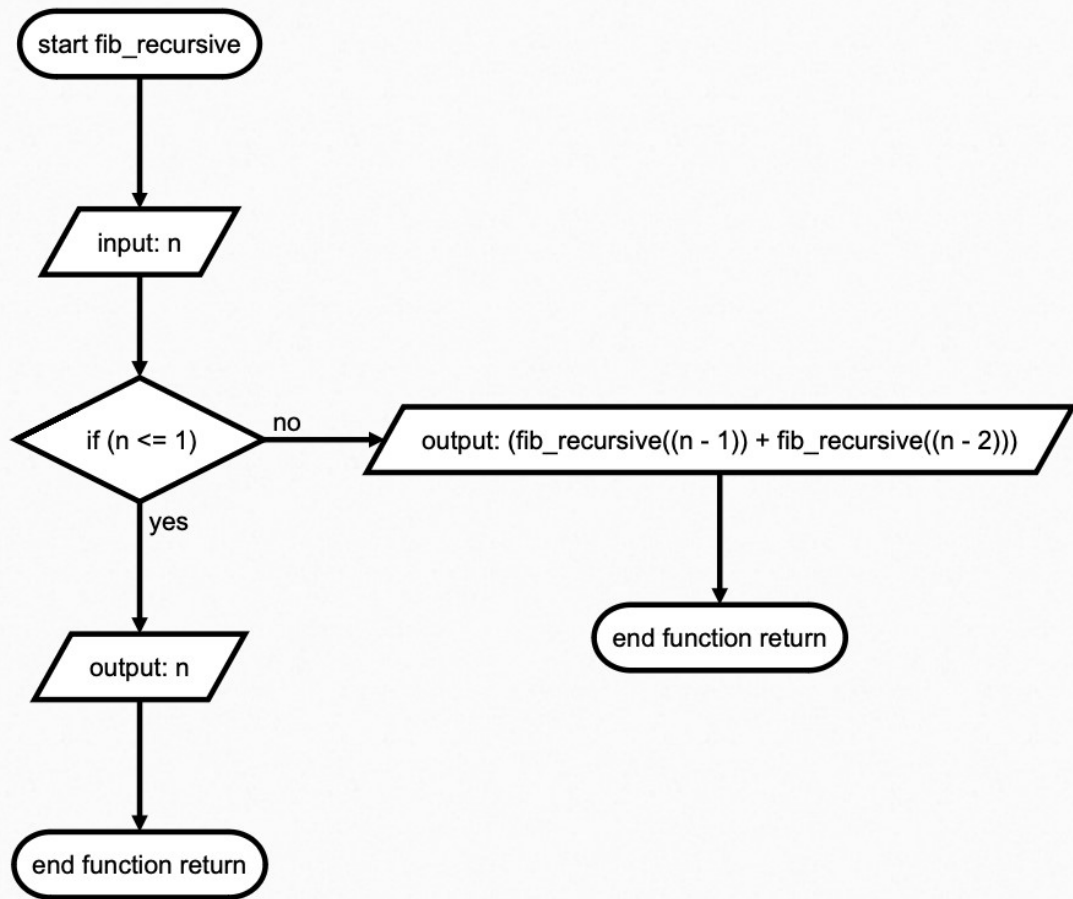
The image shows a Visual Studio Code editor window with a Python file named `week3recursive.py`. The code defines a recursive function `fib_recursive` to calculate the nth Fibonacci number. A test case is provided where `n = 3`, and the output is printed as `2`. The terminal at the bottom shows the command `python3 /Users/rakesh_kasha/Desktop/SFBU/sem_3/Algorithms/week3recursive.py` being executed, which outputs `2`.

```
1 def fib_recursive(n: int) -> int:
2     if n <= 1:
3         return n
4
5     return fib_recursive(n-1) + fib_recursive(n-2)
6
7 # Test case
8 n = 3
9 result = fib_recursive(n)
10 print(result) # Output: 2
11
```

Terminal Output:

```
/usr/local/bin/python3 /Users/rakesh_kasha/Desktop/SFBU/sem_3/Algorithms/week3recursive.py
(base) rakesh_kasha@Rakesh-kashas-MacBook-Pro ~ % /usr/local/bin/python3 /Users/rakesh_kasha/Desktop/SFBU/sem_3/Algorithms/week3recursive.py
2
(base) rakesh_kasha@Rakesh-kashas-MacBook-Pro ~ %
```

Flowchart:



Trace table:

Recursive

Step	n	$n \leq 1$	fib_recursive return
1	3		
2		false	
3			$\text{fib_recursive}(n-1) + \text{fib_recursive}(n-2)$ $(3-1) \quad (3-2)$
4	2		
5		False	
6			$\text{fib_recursive}(n-1) + \text{fib_recursive}(n-2)$ $(2-1) + (2-2)$
7	1		
8		True	
9	3		
10	4		1 + $\text{fib_recursive}(0)$
11	0		
12		True	
13			1
14			$1 + 1 = 2$

Comparison code:

```
import time
import sys

# Increase the recursion limit to handle larger
values of n
sys.setrecursionlimit(10**5)

# Iterative solution
def fib_iterative(n: int) -> int:
    if n <= 1:
        return n

    fib_prev = 0
    fib_curr = 1

    for _ in range(2, n+1):
        fib_next = fib_prev + fib_curr
        fib_prev, fib_curr = fib_curr, fib_next

    return fib_curr

# Recursive solution
def fib_recursive(n: int) -> int:
    if n <= 1:
        return n

    return fib_recursive(n-1) + fib_recursive(n-2)

# Test with option 1: n = 20 cycles
n_option1 = 20

# Measure the execution time of the iterative
solution
start_time = time.time()
fib_iterative(n_option1)
iterative_time = time.time() - start_time

# Measure the execution time of the recursive
solution
start_time = time.time()
fib_recursive(n_option1)
recursive_time = time.time() - start_time

print(f"Iterative Time (n={n_option1}):
{iterative_time} seconds")
```

```
print(f"Recursive Time (n={n_option1}):  
{recursive_time} seconds")  
print()  
  
# Test with option 2: n = 100000 cycles  
n_option2 = 100000  
  
# Measure the execution time of the iterative  
solution  
start_time = time.time()  
fib_iterative(n_option2)  
iterative_time = time.time() - start_time  
  
# Measure the execution time of the recursive  
solution  
start_time = time.time()  
fib_recursive(n_option2)  
recursive_time = time.time() - start_time  
  
print(f"Iterative Time (n={n_option2}):  
{iterative_time} seconds")  
print(f"Recursive Time (n={n_option2}):  
{recursive_time} seconds")
```



```

37 start_time = time.time()
38 fib_recursive(n_option1)
39 recursive_time = time.time() - start_time
40
41 print(f"Iterative Time (n={n_option1}): {iterative_time} seconds")
42 print(f"Recursive Time (n={n_option1}): {recursive_time} seconds")
43 print()
44
45 # Test with option 2: n = 100000 cycles
46 n_option2 = 100000
47
48 # Measure the execution time of the iterative solution
49 start_time = time.time()
50 fib_iterative(n_option2)
51 iterative_time = time.time() - start_time
52

```

PROBLEMS OUTPUT DEBUG CONSOLE **TERMINAL**

```

/usr/local/bin/python3 /Users/rakesh_kasha/Desktop/SFBU/sem_3/Algorithms/comparsion.py
(base) rakesh_kasha@Rakesh-kashas-MacBook-Pro ~ % /usr/local/bin/python3 /Users/rakesh_kasha/Desktop/SFBU/sem_3/Algorithms/comparsion.py
Iterative Time (n=20): 2.86102294921875e-06 seconds
Recursive Time (n=20): 0.0011641979217529297 seconds

zsh: segmentation fault /usr/local/bin/python3
(base) rakesh_kasha@Rakesh-kashas-MacBook-Pro ~ %

```

Table Results:

Option	Iterative	Recursive
n = 20 cycles	The program's execution time is 2.861	The program's execution time is 0.001
n = 100000 cycles	The program got crashed	The program got crashed
Big-O	$O(n)$	$O(2^n)$