## Practical 9 `

**Aim:** Write a program to implement multi-threaded computation concepts in the generation of Fibonacci numbers.

**Theory:**

1. A thread is a path of execution within a process, where a process can contains ‘n’ numbers of threads
2. Hence, the idea of running multiple threads in parallel to complete a process is called multi-Threading

**Algorithm:**

Procedure Iterative\_Fibonacci(n):

int f0, f1, fib

f0 = 0

f1 = 1

display f0, f1

for int i := 1 to n:

fib := f0 + f1

f0 := f1

f1 := fib

display fib

END for loop

END Iterative\_Fibonacci

**Code:**

# Function for nth Fibonacci number

def Fibonacci(n):

# Check if input is 0 then it will

# print incorrect input

if n < 0:

print("Incorrect input")

# Check if n is 0

# then it will return 0

elif n == 0:

return 0

# Check if n is 1,2

# it will return 1

elif n == 1 or n == 2:

return 1

else:

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

# Driver Program

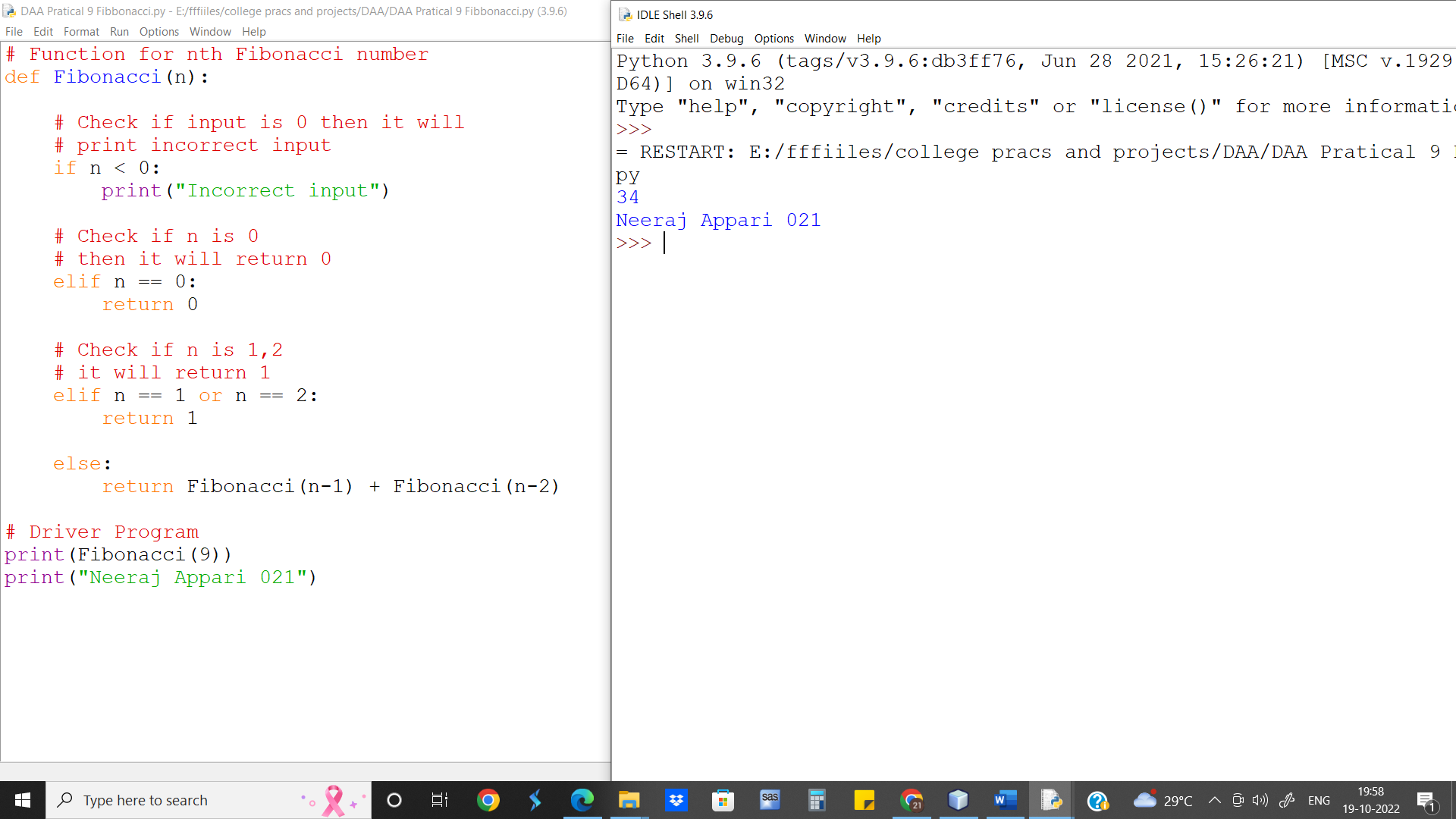
print(Fibonacci(9))

print("Neeraj Appari 021")

**Output**:

34

Neeraj Appari 021



**Complexity Analysis:-** Normal iterative Fibonacci code takes O(n) time.

To compute x numbers of elements it will take x\*n or O(x\*n)

Where as using multithread the time need to calculate is reduced greatly.

Theoretically the time complexity will still be O(n) for x elements.

**Conclusion:-** We notice that for comparing multithreaded with non-thread versions, the time difference is not much when either the numbers of elements are less or the number themselves are small.