Lab Program - 1

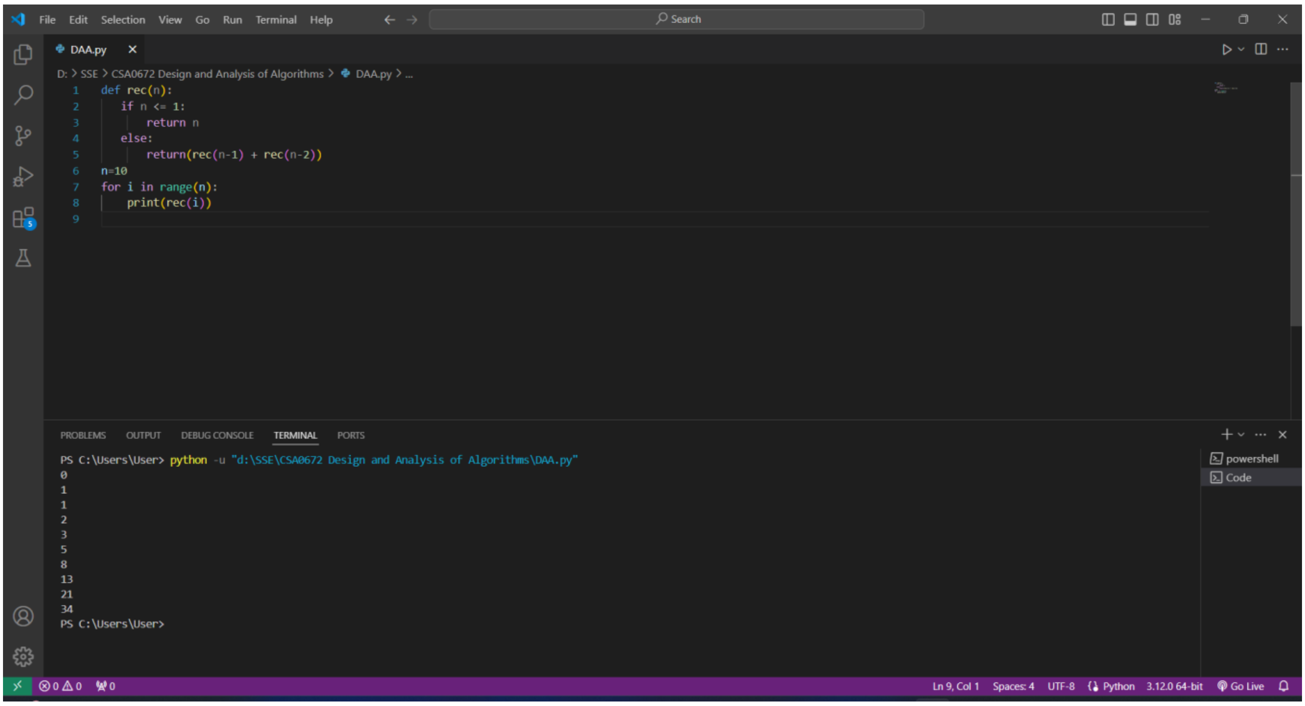
SOORYA – 192311060

1. Write a program to Print Fibonacci Series using recursion.

Code:

```
def rec(n):
  if n <= 1:
      return n
  else:
      return(rec(n-1) + rec(n-2))
n=10
for i in range(n):
   print(rec(i))
```

Screenshot for I/O:



```
def rec(n):
    if n <= 1:
        return n
    else:
        return(rec(n-1) + rec(n-2))
n=10
for i in range(n):
    print(rec(i))
```

```
PS C:\Users\User> python -u "d:\SSE\CSA0672 Design and Analysis of Algorithms\DAA.py"
0
1
1
2
3
5
8
13
21
34
PS C:\Users\User>
```

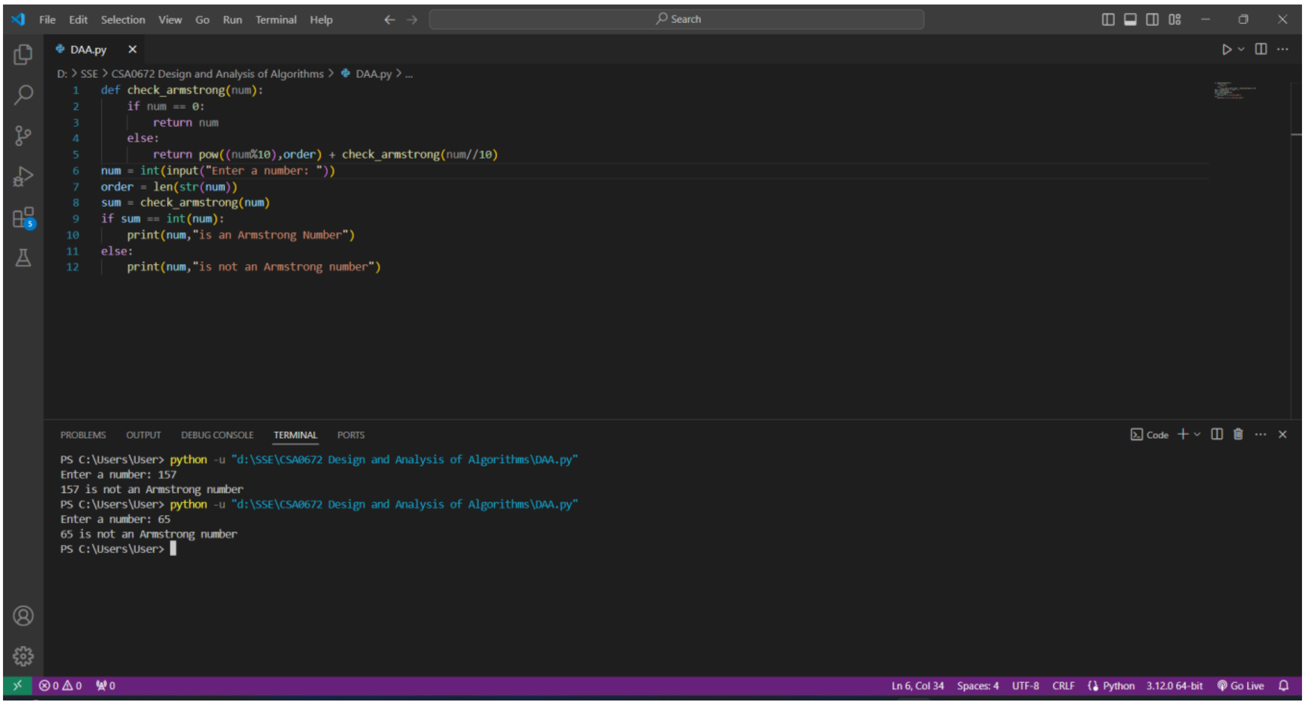
Time Complexity:O(n)

2. Write a program to check the given no is Armstrong or not using recursive function.

Code:

```
def check_armstrong(num):
    if num == 0:
        return num
    else:
        return pow((num%10),order) + check_armstrong(num//10)
num = int(input("Enter a number: "))
order = len(str(num))
sum = check_armstrong(num)
if sum == int(num):
    print(num,"is an Armstrong Number")
else:
    print(num,"is not an Armstrong number")
```

Screenshot for I/O:



```
def check_armstrong(num):
    if num == 0:
        return num
    else:
        return pow((num%10),order) + check_armstrong(num//10)
num = int(input("Enter a number: "))
order = len(str(num))
sum = check_armstrong(num)
if sum == int(num):
    print(num,"is an Armstrong Number")
else:
    print(num,"is not an Armstrong number")
```

```
PS C:\Users\User> python -u "d:\SSE\CSA0672 Design and Analysis of Algorithms\DAA.py"
Enter a number: 157
157 is not an Armstrong number
PS C:\Users\User> python -u "d:\SSE\CSA0672 Design and Analysis of Algorithms\DAA.py"
Enter a number: 65
65 is not an Armstrong number
PS C:\Users\User>
```

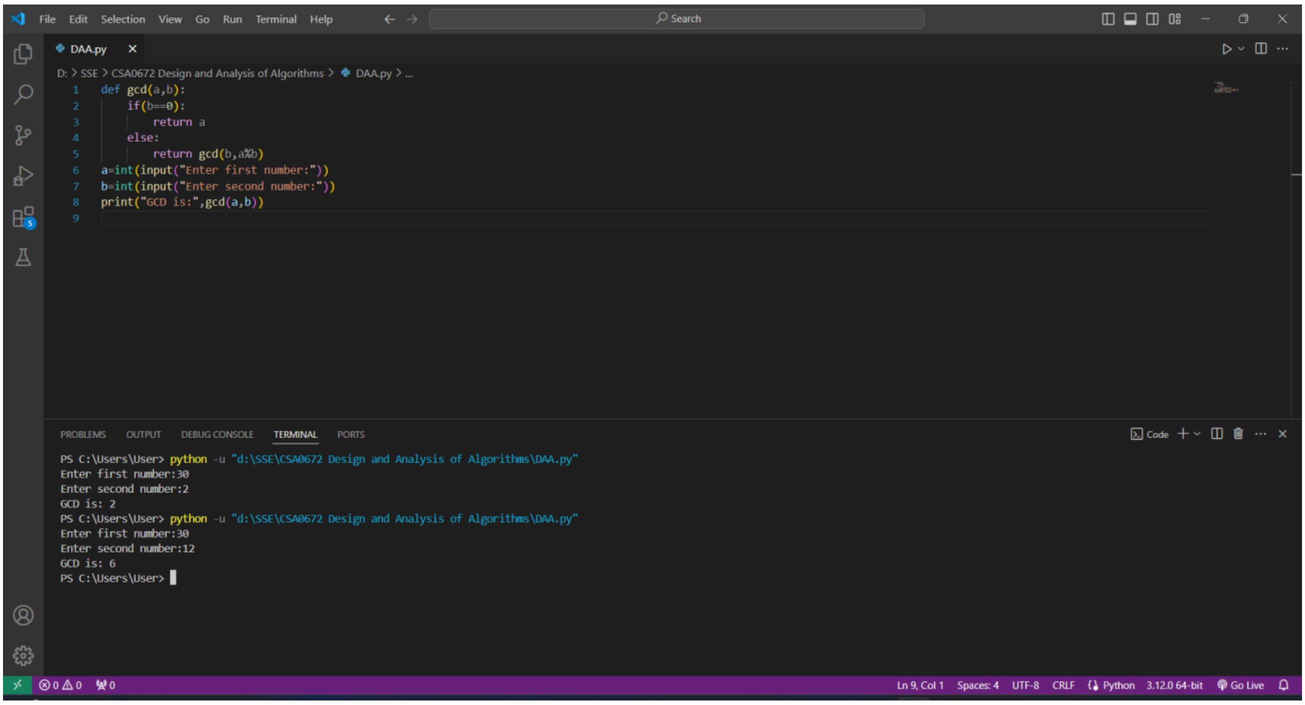
Time Complexity:O(n)

3. Write a program to find the GCD of two numbers using recursive factorization

Code:

```
def gcd(a,b):
    if(b==0):
        return a
    else:
        return gcd(b,a%b)
a=int(input("Enter first number:"))
b=int(input("Enter second number:"))
print("GCD is:",gcd(a,b))
```

Screenshot for I/O:



File Edit Selection View Go Run Terminal Help
Search
DAA.py
D: > SSE > CSA0672 Design and Analysis of Algorithms > DAA.py > ...
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Code
Ln 9, Col 1 Spaces: 4 UTF-8 CRLF Python 3.12.0 64-bit Go Live

```
def gcd(a,b):
    if(b==0):
        return a
    else:
        return gcd(b,a%b)
a=int(input("Enter first number:"))
b=int(input("Enter second number:"))
print("GCD is:",gcd(a,b))
```

```
PS C:\Users\User> python -u "d:\SSE\CSA0672 Design and Analysis of Algorithms\DAA.py"
Enter first number:30
Enter second number:2
GCD is: 2
PS C:\Users\User> python -u "d:\SSE\CSA0672 Design and Analysis of Algorithms\DAA.py"
Enter first number:30
Enter second number:12
GCD is: 6
PS C:\Users\User>
```

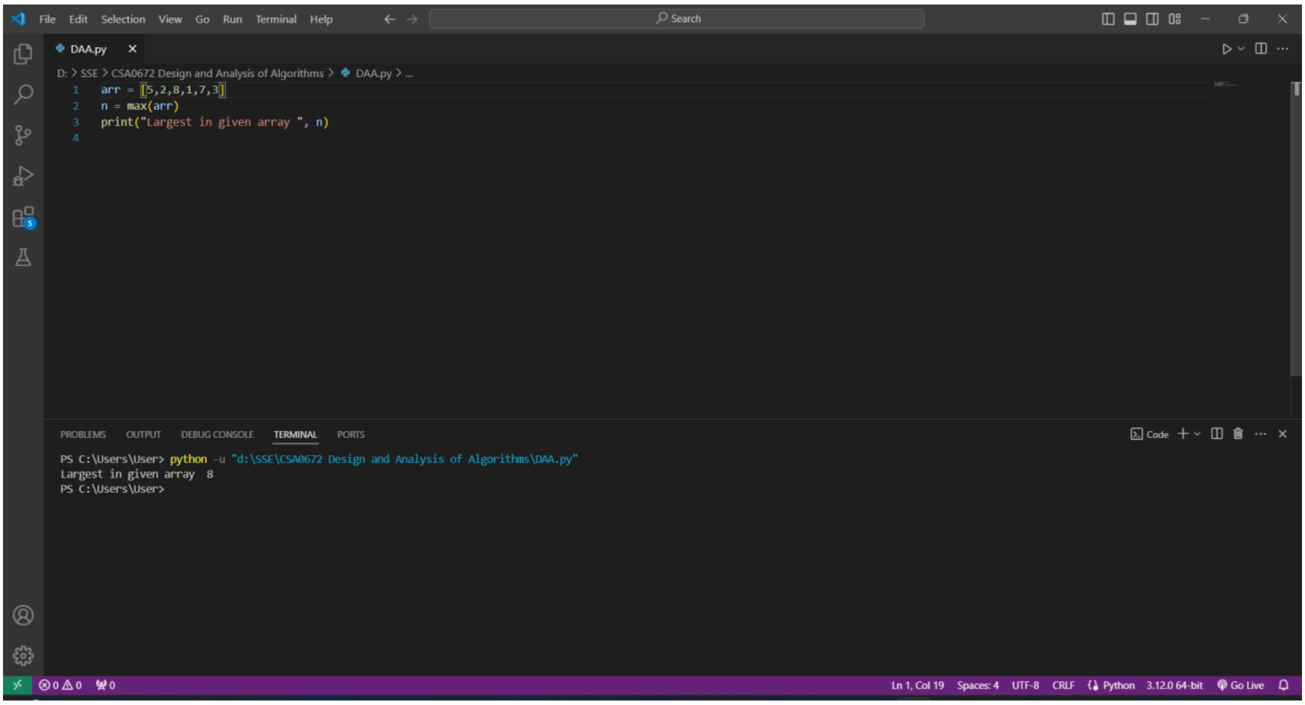
Time Complexity: O(n)

4. Write a program to get the largest element of an array.

Code:

```
arr = [5,2,8,1,7,3]
n = max(arr)
print("Largest in given array ", n)
```

Screenshot for I/O:



File Edit Selection View Go Run Terminal Help
Search
DAA.py
D: > SSE > CSA0672 Design and Analysis of Algorithms > DAA.py > ...
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Code
Ln 1, Col 19 Spaces: 4 UTF-8 CRLF Python 3.12.0 64-bit Go Live

```
arr = [5,2,8,1,7,3]
n = max(arr)
print("Largest in given array ", n)
```

```
PS C:\Users\User> python -u "d:\SSE\CSA0672 Design and Analysis of Algorithms\DAA.py"
Largest in given array  8
PS C:\Users\User>
```

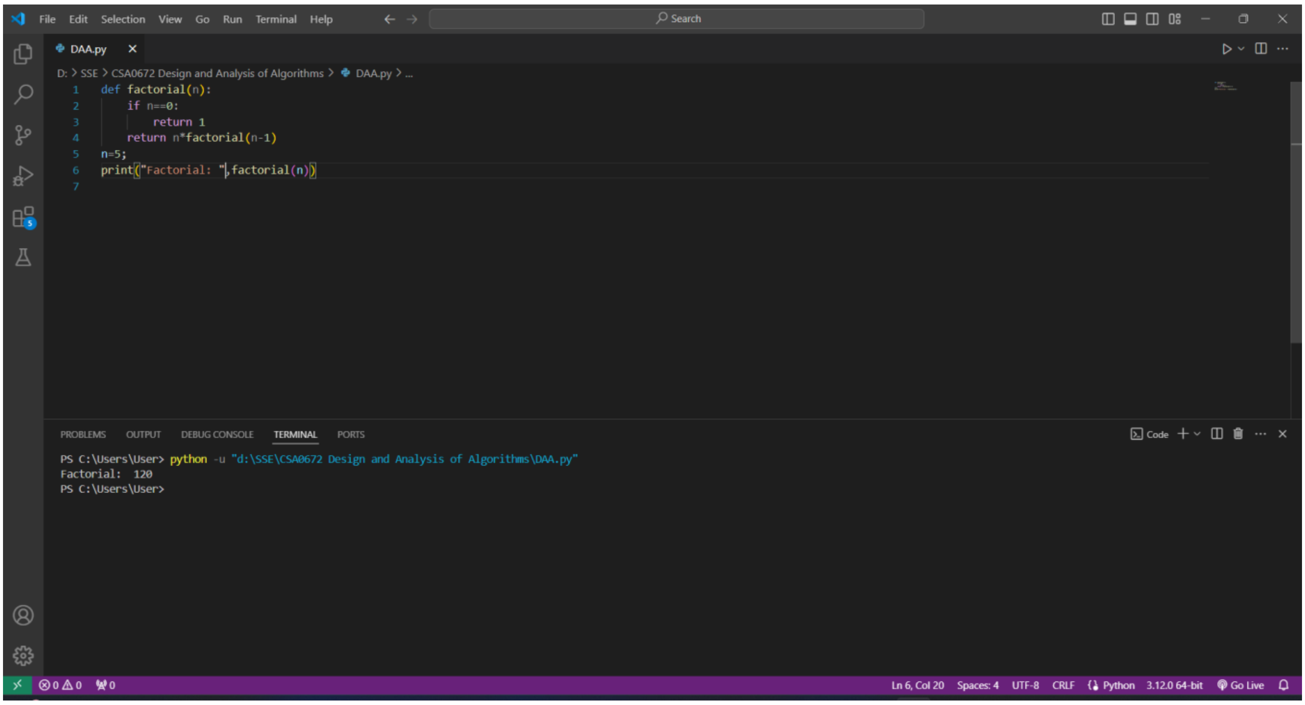
Time Complexity: O(n)

5. Write a program to find the Factorial of a number using recursion.

Code:

```
def factorial(n):
    if n==0:
        return 1
    return n*factorial(n-1)
n=5;
print("Factorial: ",factorial(n))
```

Screenshot for I/O:



```
def factorial(n):
    if n==0:
        return 1
    return n*factorial(n-1)
n=5;
print("Factorial: ",factorial(n))
```

```
PS C:\Users\User> python -u "d:\SSE\CSA0672 Design and Analysis of Algorithms\DAA.py"
Factorial:  120
PS C:\Users\User>
```

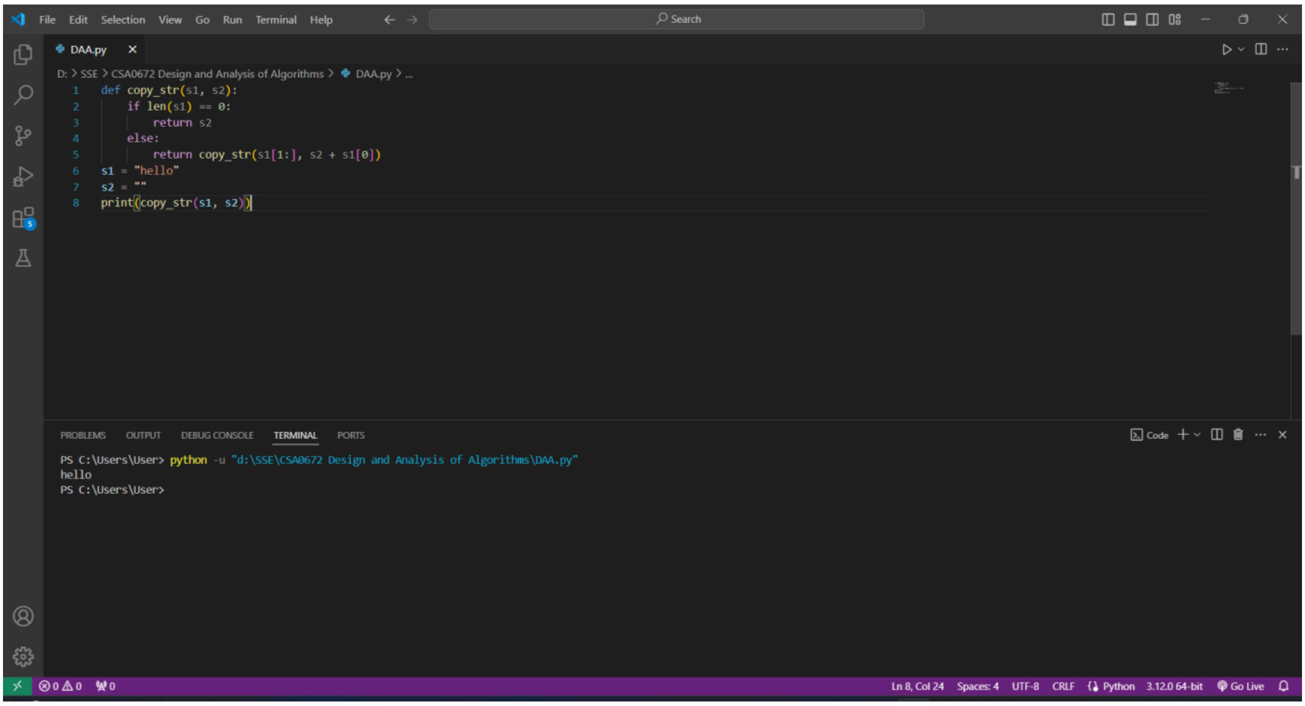
Time Complexity: O(n)

6. Write a program for to copy one string to another using recursion

Code:

```
def copy_str(s1, s2):
    if len(s1) == 0:
        return s2
    else:
        return copy_str(s1[1:], s2 + s1[0])
s1 = "hello"
s2 = ""
print(copy_str(s1, s2))
```

Screenshot for I/O:



```
def copy_str(s1, s2):
    if len(s1) == 0:
        return s2
    else:
        return copy_str(s1[1:], s2 + s1[0])
s1 = "hello"
s2 = ""
print(copy_str(s1, s2))
```

```
PS C:\Users\User> python -u "d:\SSE\CSA0672 Design and Analysis of Algorithms\DAA.py"
hello
PS C:\Users\User>
```

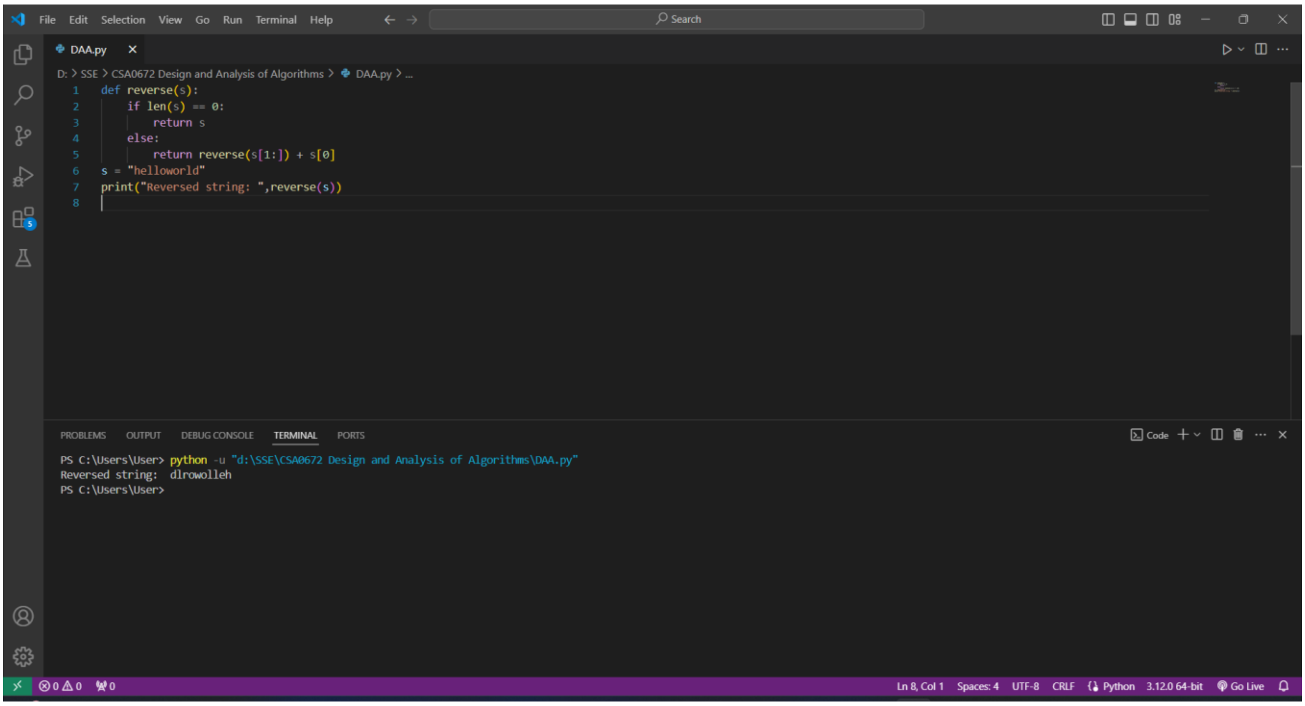
Time Complexity: O(n)

7. Write a program to print the reverse of a string using recursion

Code:

```
def reverse(s):
        if len(s) == 0:
                return s
        else:
                return reverse(s[1:]) + s[0]
s = "helloworld"
print("Reversed string: ",reverse(s))
```

Screenshot for I/O:



```
def reverse(s):
    if len(s) == 0:
        return s
    else:
        return reverse(s[1:]) + s[0]
s = "helloworld"
print("Reversed string: ",reverse(s))
```

```
PS C:\Users\User> python -u "d:\SSE\CSA0672 Design and Analysis of Algorithms\DAA.py"
Reversed string:  dlrowolleh
PS C:\Users\User>
```

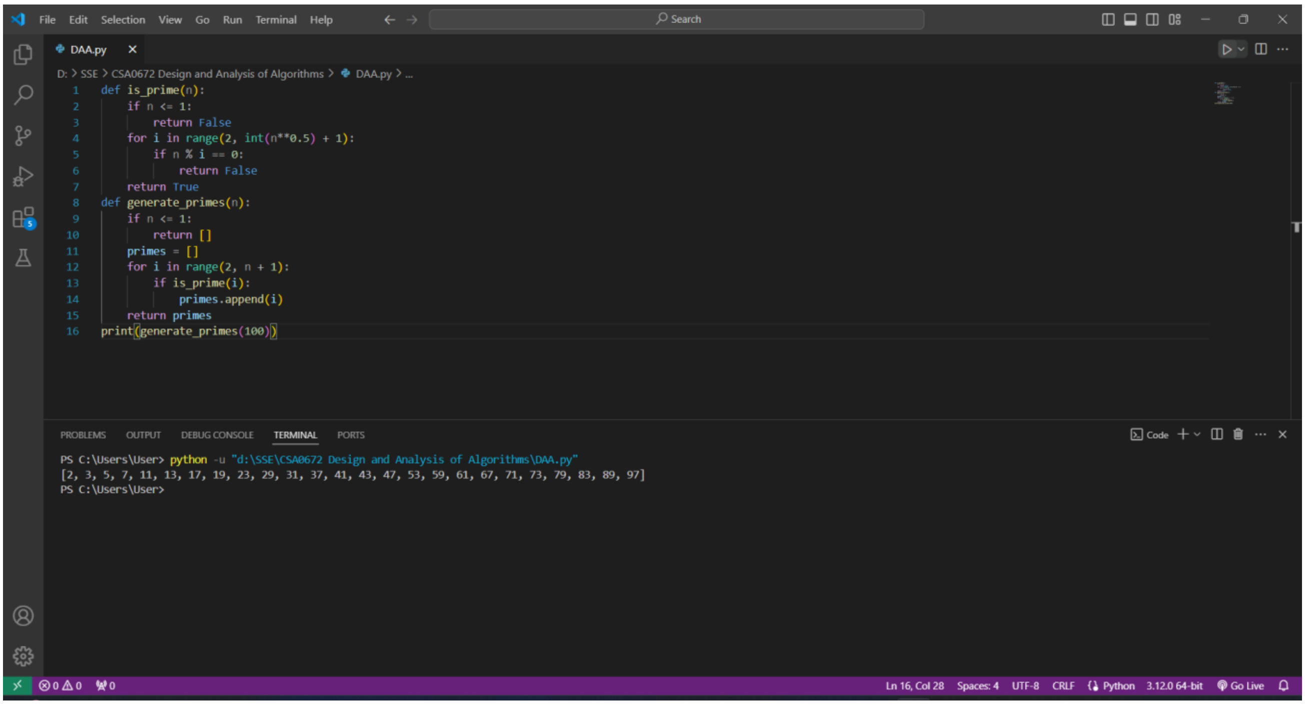
Time Complexity: O(n)

8. Write a program   to generate all the prime numbers using recursion

Code:

```
def is_prime(n):
    if n <= 1:
        return False
    for i in range(2, int(n**0.5) + 1):
        if n % i == 0:
            return False
    return True
def generate_primes(n):
    if n <= 1:
        return []
    primes = []
    for i in range(2, n + 1):
        if is_prime(i):
            primes.append(i)
    return primes
print(generate_primes(100))
```

Screenshot for I/O:


File Edit Selection View Go Run Terminal Help
Search
DAA.py
D: > SSE > CSA0672 Design and Analysis of Algorithms > DAA.py > ...
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Code
Ln 16, Col 28 Spaces: 4 UTF-8 CRLF Python 3.12.0 64-bit Go Live

```
def is_prime(n):
    if n <= 1:
        return False
    for i in range(2, int(n**0.5) + 1):
        if n % i == 0:
            return False
    return True
def generate_primes(n):
    if n <= 1:
        return []
    primes = []
    for i in range(2, n + 1):
        if is_prime(i):
            primes.append(i)
    return primes
print(generate_primes(100))
```

```
PS C:\Users\User> python -u "d:\SSE\CSA0672 Design and Analysis of Algorithms\DAA.py"
[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97]
PS C:\Users\User>
```

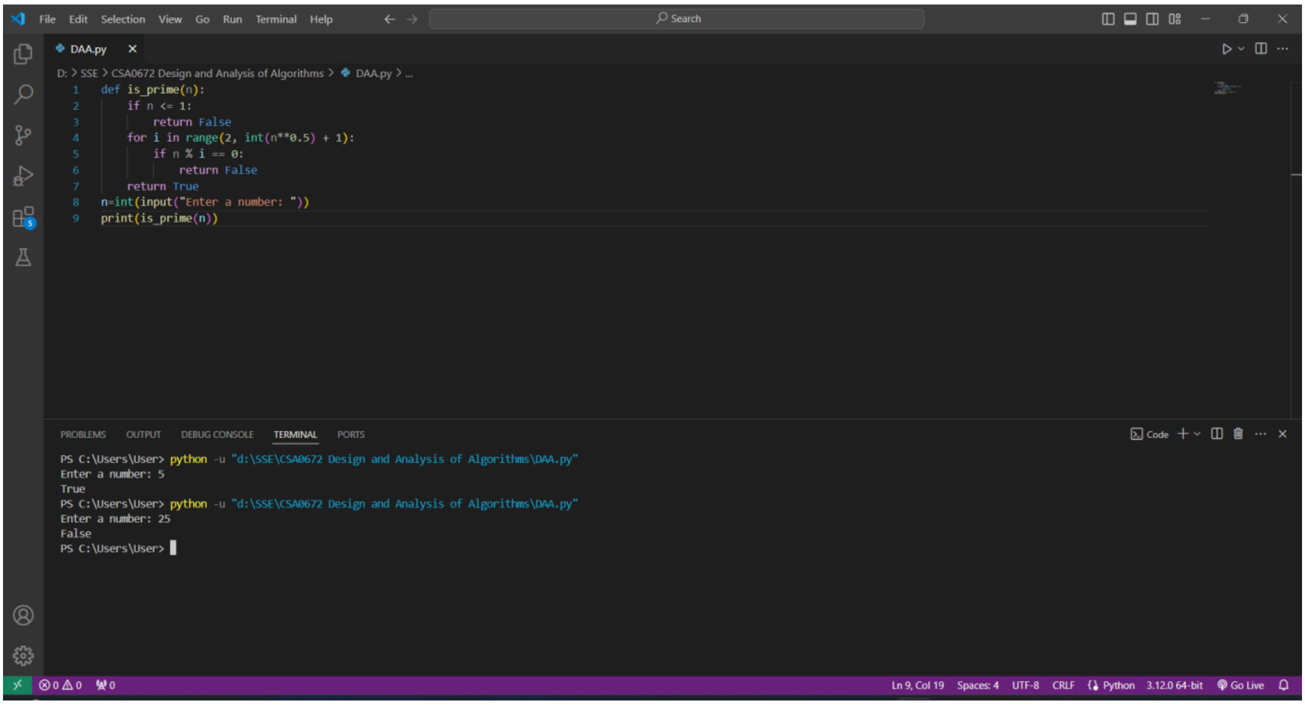
Time Complexity: O(n*m)

9. Write a program to check a number is a prime number or not using recursion.

Code:

```
def is_prime(n):
    if n <= 1:
        return False
    for i in range(2, int(n**0.5) + 1):
        if n % i == 0:
            return False
    return True
n=int(input("Enter a number: "))
print(is_prime(n))
```

Screenshot for I/O:



```
def is_prime(n):
    if n <= 1:
        return False
    for i in range(2, int(n**0.5) + 1):
        if n % i == 0:
            return False
    return True
n=int(input("Enter a number: "))
print(is_prime(n))
```

```
PS C:\Users\User> python -u "d:\SSE\CSA0672 Design and Analysis of Algorithms\DAA.py"
Enter a number: 5
True
PS C:\Users\User> python -u "d:\SSE\CSA0672 Design and Analysis of Algorithms\DAA.py"
Enter a number: 25
False
PS C:\Users\User>
```

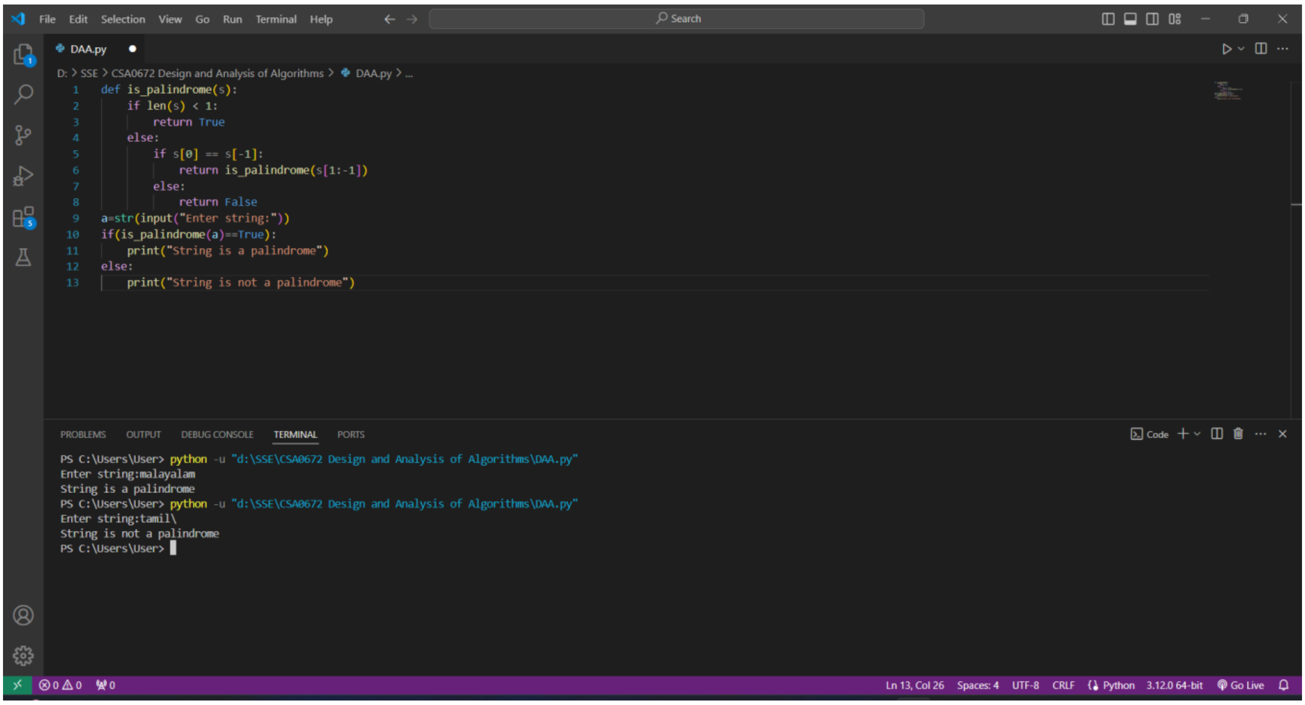
Time Complexity: O(n)

10. Write a program for to check whether a given String is Palindrome or not using recursion.

Code:

```
def is_palindrome(s):
    if len(s) < 1:
        return True
    else:
        if s[0] == s[-1]:
            return is_palindrome(s[1:-1])
        else:
            return False
a=str(input("Enter string:"))
if(is_palindrome(a)==True):
    print("String is a palindrome")
else:
    print("String is not a palindrome")
```

Screenshot for I/O:



```
def is_palindrome(s):
    if len(s) < 1:
        return True
    else:
        if s[0] == s[-1]:
            return is_palindrome(s[1:-1])
        else:
            return False
a=str(input("Enter string:"))
if(is_palindrome(a)==True):
    print("String is a palindrome")
else:
    print("String is not a palindrome")
```

```
PS C:\Users\User> python -u "d:\SSE\CSA0672 Design and Analysis of Algorithms\DAA.py"
Enter string:malayalam
String is a palindrome
PS C:\Users\User> python -u "d:\SSE\CSA0672 Design and Analysis of Algorithms\DAA.py"
Enter string:tamil\
String is not a palindrome
PS C:\Users\User>
```

Time Complexity: O(n)