

Lab Program - 2

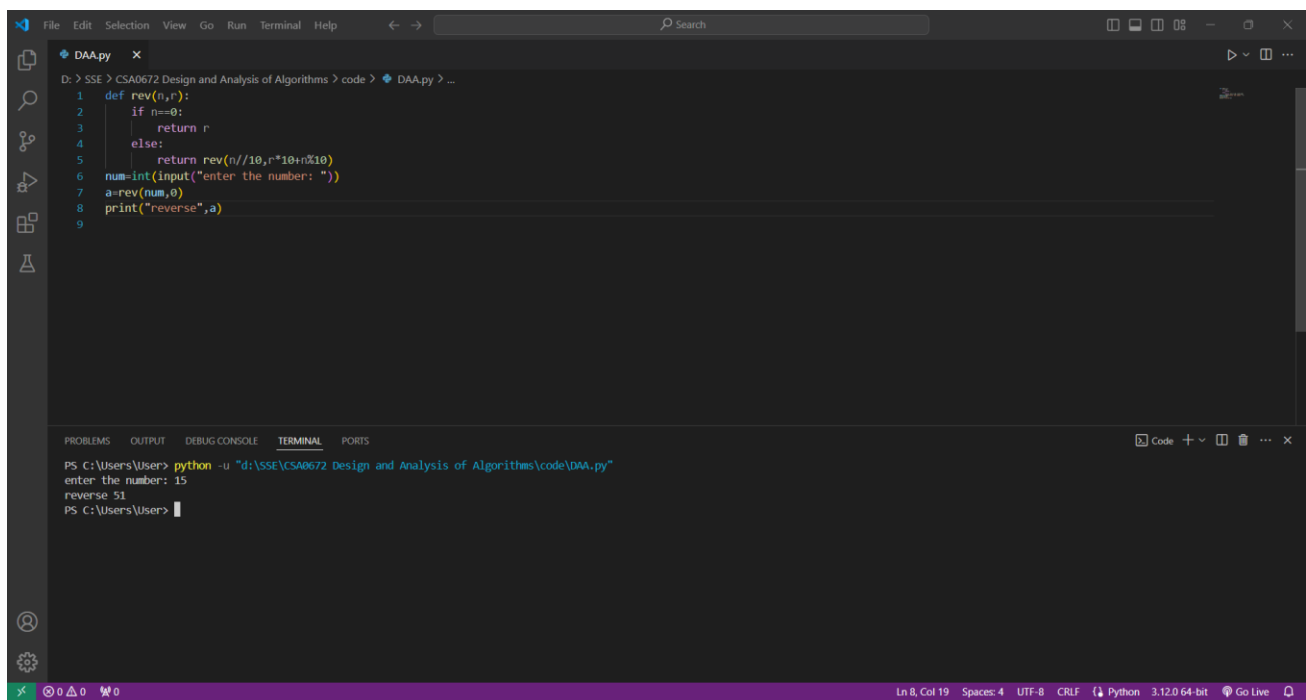
T Godwin Abilash – 192321012

1. Write a program to find the reverse of a given number using recursive.

Code:

```
def rev(n,r):  
    if n==0:  
        return r  
    else:  
        return rev(n//10,r*10+n%10)  
num=int(input("enter the number: "))  
a=rev(num,0)  
print("reverse",a)
```

Screenshot for I/O:



The screenshot displays a code editor with a dark theme. The top section shows the Python code for reversing a number using recursion. The bottom section, labeled 'TERMINAL', shows the execution of the program. The user enters the number 15, and the program outputs 'reverse 51'.

```
DAA.py x  
D:\SSE\CSA0672 Design and Analysis of Algorithms\code> DAA.py > ...  
1 def rev(n,r):  
2     if n==0:  
3         return r  
4     else:  
5         return rev(n//10,r*10+n%10)  
6 num=int(input("enter the number: "))  
7 a=rev(num,0)  
8 print("reverse",a)  
9  
  
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS  
PS C:\Users\User> python -u "d:\SSE\CSA0672 Design and Analysis of Algorithms\code\DAA.py"  
enter the number: 15  
reverse 51  
PS C:\Users\User> |
```

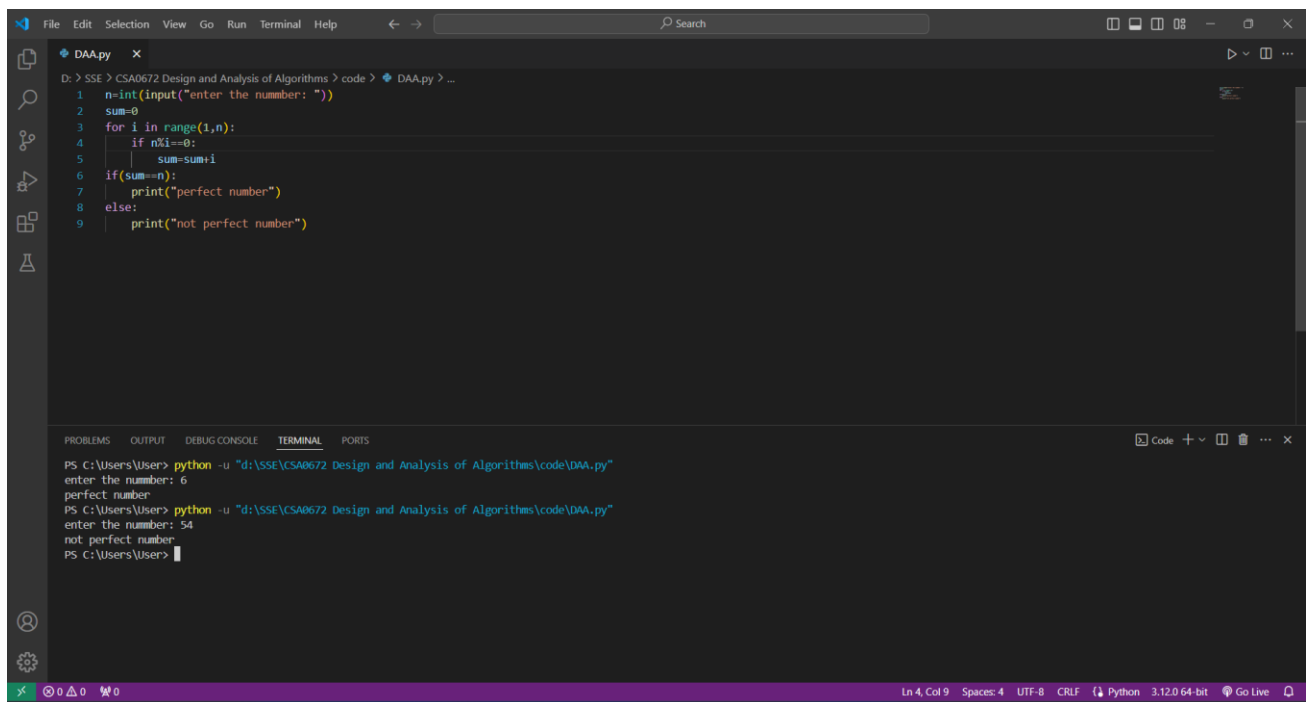
Time Complexity: $O(n)$

2. Write a program to find the perfect number.

Code:

```
n=int(input("enter the nummber: "))
sum=0
for i in range(1,n):
    if n%i==0:
        sum=sum+i
if(sum==n):
    print("perfect number")
else:
    print("not perfect number")
```

Screenshot for I/O:



The screenshot displays a code editor with a Python script named 'DAA.py'. The script implements a function to check if a number is perfect by summing its divisors. Below the code editor, the terminal window shows the execution of the program. It prompts the user to enter a number, and two examples are shown: 6, which is identified as a perfect number, and 54, which is identified as not a perfect number.

```
D:\SSE\CSA0672 Design and Analysis of Algorithms\code> DAA.py > ...
1 n=int(input("enter the number: "))
2 sum=0
3 for i in range(1,n):
4     if n%i==0:
5         sum=sum+i
6 if(sum==n):
7     print("perfect number")
8 else:
9     print("not perfect number")
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\User> python -u "d:\SSE\CSA0672 Design and Analysis of Algorithms\code\DAA.py"
enter the number: 6
perfect number
PS C:\Users\User> python -u "d:\SSE\CSA0672 Design and Analysis of Algorithms\code\DAA.py"
enter the number: 54
not perfect number
PS C:\Users\User>
```

Ln 4, Col 9 Spaces: 4 UTF-8 CRLF Python 3.12.0 64-bit Go Live

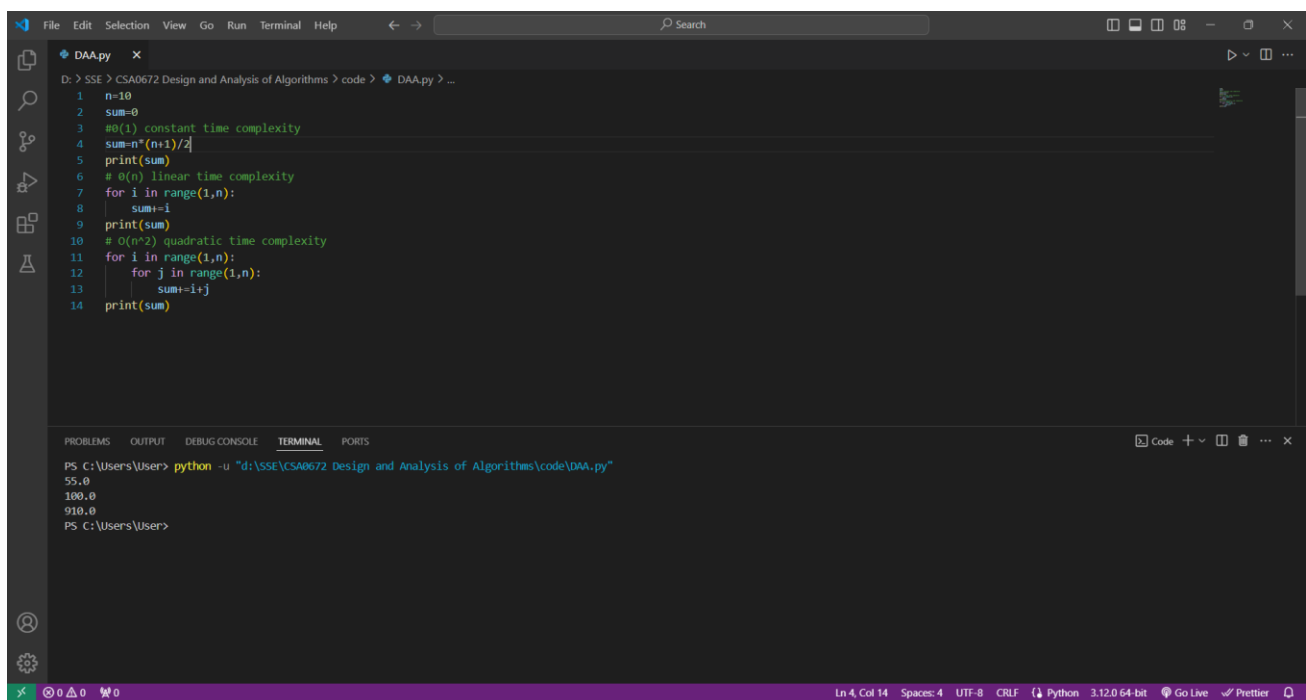
Time Complexity: $O(n)$

3. Write C program that demonstrates the usage of these notations by analyzing the time complexity of some example algorithms.

Code:

```
n=10
sum=0
#O(1) constant time complexity
sum=n*(n+1)/2
print(sum)
# O(n) linear time complexity
for i in range(1,n):
    sum+=i
print(sum)
# O(n^2) quadratic time complexity
for i in range(1,n):
    for j in range(1,n):
        sum+=i+j
print(sum)
```

Screenshot for I/O:



```
File Edit Selection View Go Run Terminal Help
D:\> SSE > CSA0672 Design and Analysis of Algorithms > code > DAA.py > ...
1 n=10
2 sum=0
3 #O(1) constant time complexity
4 sum=n*(n+1)/2
5 print(sum)
6 # O(n) linear time complexity
7 for i in range(1,n):
8     sum+=i
9 print(sum)
10 # O(n^2) quadratic time complexity
11 for i in range(1,n):
12     for j in range(1,n):
13         sum+=i+j
14 print(sum)

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\User> python -u "d:\SSE\CSA0672 Design and Analysis of Algorithms\code\DAA.py"
55.0
100.0
910.0
PS C:\Users\User>
```

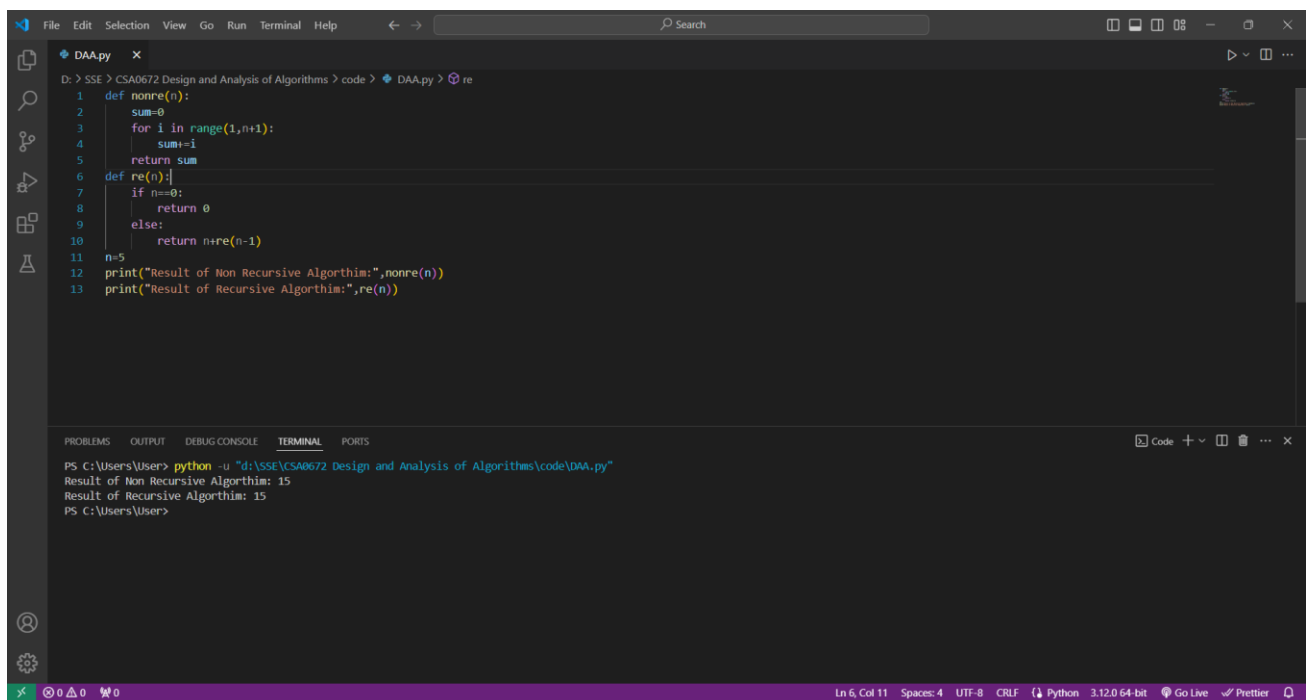
Time Complexity: $O(n^2)$

4. Write C programs that demonstrate the mathematical analysis of non-recursive and recursive algorithms.

Code:

```
def nonre(n):
    sum=0
    for i in range(1,n+1):
        sum+=i
    return sum
def re(n):
    if n==0:
        return 0
    else:
        return n+re(n-1)
n=5
print("Result of Non Recursive Algorithim:",nonre(n))
print("Result of Recursive Algorithim:",re(n))
```

Screenshot for I/O:

A screenshot of a code editor window titled 'DAA.py'. The editor shows the Python code for calculating the sum of numbers from 1 to n using both non-recursive and recursive methods. The code is as follows:

```
1 def nonre(n):
2     sum=0
3     for i in range(1,n+1):
4         sum+=i
5     return sum
6 def re(n):
7     if n==0:
8         return 0
9     else:
10        return n+re(n-1)
11 n=5
12 print("Result of Non Recursive Algorithim:",nonre(n))
13 print("Result of Recursive Algorithim:",re(n))
```

The editor's interface includes a sidebar with icons for Explorer, Search, Source Control, and Run and Debug. The bottom panel shows the 'TERMINAL' output, which displays the execution of the code:

```
PS C:\Users\User> python -u "d:\SSE\CSA0672 Design and Analysis of Algorithms\code\DAA.py"
Result of Non Recursive Algorithm: 15
Result of Recursive Algorithm: 15
PS C:\Users\User>
```

The status bar at the bottom indicates the current line and column (Ln 6, Col 11), the number of spaces (4), the encoding (UTF-8), the line ending (CRLF), the Python version (3.12.0 64-bit), and the active extensions (Go Live, Prettier).

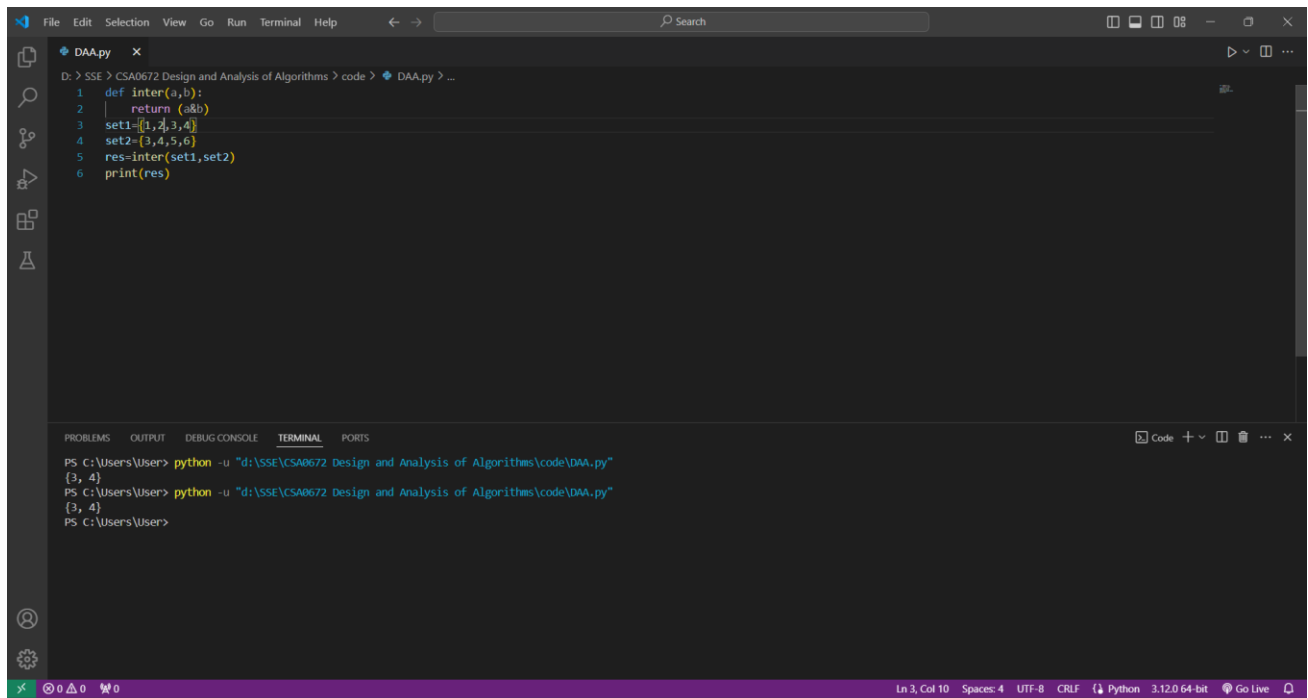
Time Complexity: $O(n)$

6. Given two integer arrays nums1 and nums2, return an array of their Intersection. Each element in the result must be unique and you may return the result in any order.

Code:

```
def inter(a,b):  
    return (a&b)  
set1={1,2,3,4}  
set2={3,4,5,6}  
res=inter(set1,set2)  
print(res)
```

Screenshot for I/O:



The screenshot shows a Visual Studio Code editor window with a file named 'DAA.py'. The code in the editor is as follows:

```
1 def inter(a,b):  
2     return (a&b)  
3 set1={1,2,3,4}  
4 set2={3,4,5,6}  
5 res=inter(set1,set2)  
6 print(res)
```

Below the editor, the 'TERMINAL' panel is open, showing the execution of the script. The commands and output are:

```
PS C:\Users\User> python -u "d:\SSE\CSA0672 Design and Analysis of Algorithms\code\DAA.py"  
{3, 4}  
PS C:\Users\User> python -u "d:\SSE\CSA0672 Design and Analysis of Algorithms\code\DAA.py"  
{3, 4}  
PS C:\Users\User>
```

The status bar at the bottom indicates the current position is 'Ln 3, Col 10', with 'Spaces: 4', 'UTF-8', 'CRLF', and 'Python 3.12.0 64-bit'.

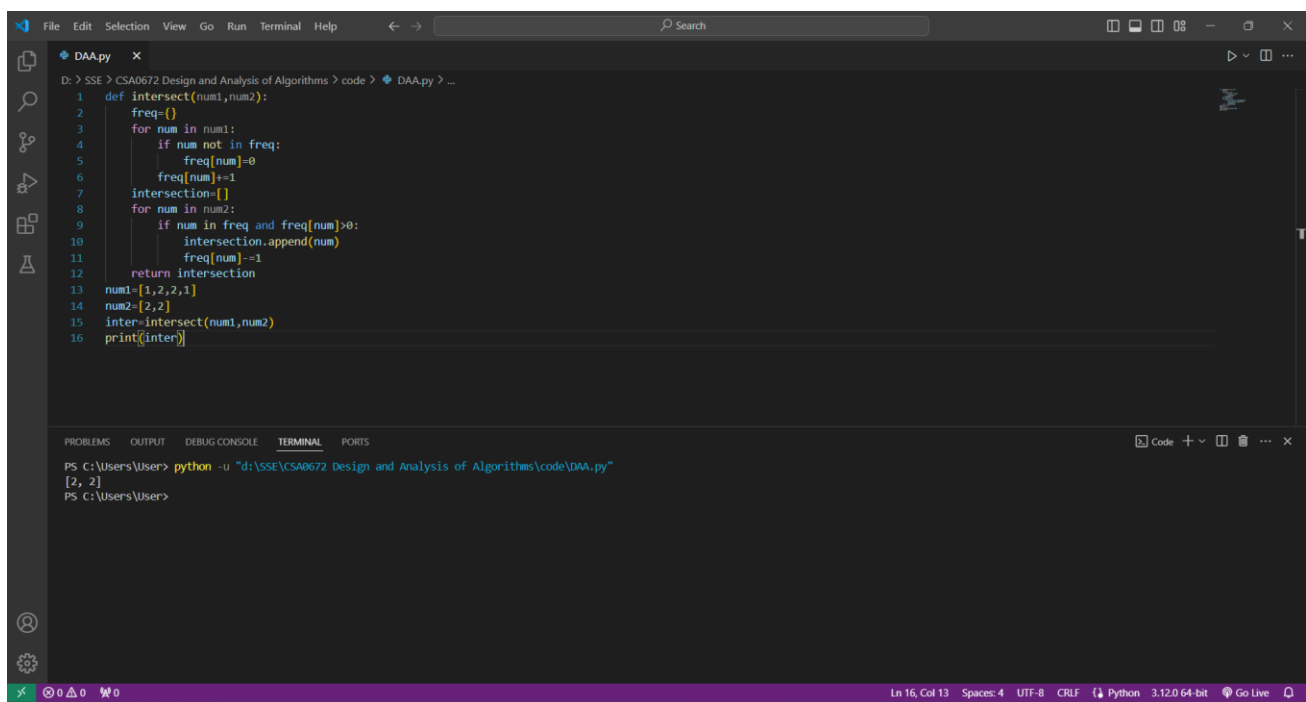
Time Complexity: $O(n)$

7. Given two integer arrays nums1 and nums2, return an array of their intersection. Each element in the result must appear as many times as it shows in both arrays and you may return the result in any order.

Code:

```
def intersect(num1,num2):
    freq={}
    for num in num1:
        if num not in freq:
            freq[num]=0
        freq[num]+=1
    intersection=[]
    for num in num2:
        if num in freq and freq[num]>0:
            intersection.append(num)
            freq[num]-=1
    return intersection
num1=[1,2,2,1]
num2=[2,2]
inter=intersect(num1,num2)
print(inter)
```

Screenshot for I/O:

A screenshot of a code editor window titled 'DAA.py'. The editor shows the following Python code:

```
1 def intersect(num1,num2):
2     freq={}
3     for num in num1:
4         if num not in freq:
5             freq[num]=0
6         freq[num]+=1
7     intersection=[]
8     for num in num2:
9         if num in freq and freq[num]>0:
10            intersection.append(num)
11            freq[num]-=1
12    return intersection
13 num1=[1,2,2,1]
14 num2=[2,2]
15 inter=intersect(num1,num2)
16 print(inter)
```

The bottom panel of the editor shows the terminal output:

```
PS c:\Users\User> python -u "d:\SSE\CSA0672 Design and Analysis of Algorithms\code\DAA.py"
[2, 2]
PS c:\Users\User>
```

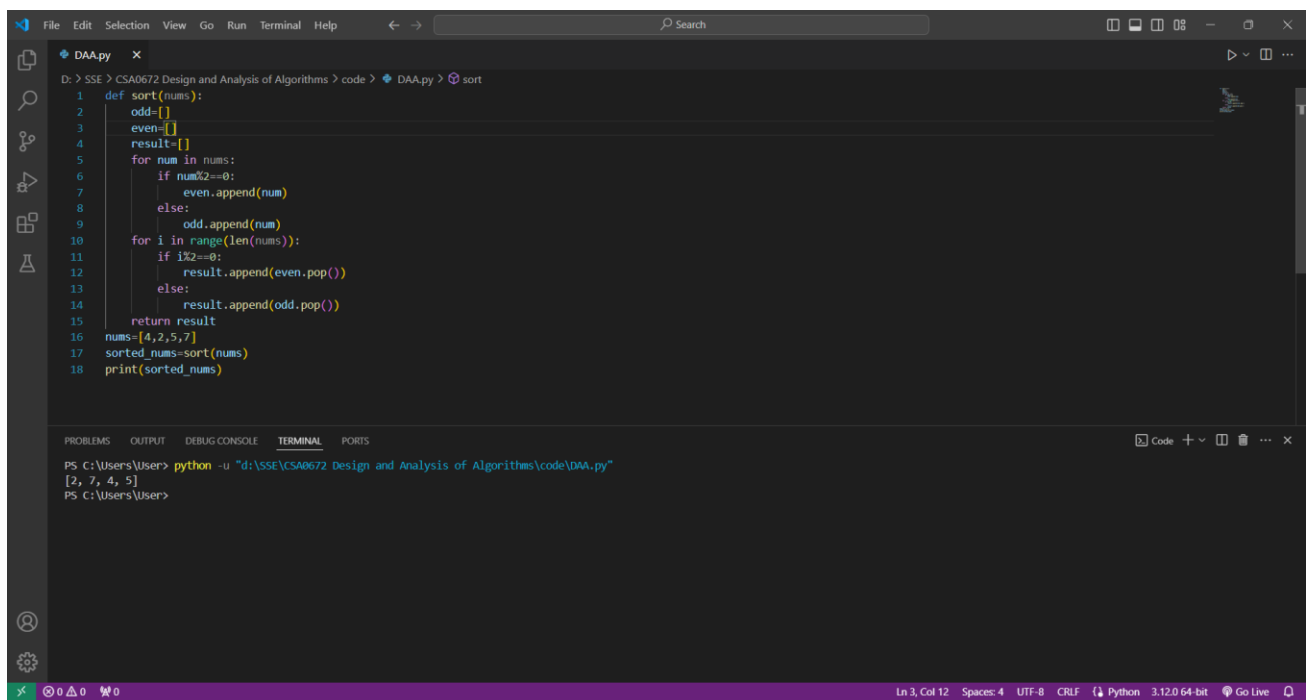
Time Complexity: $O(n*m)$

9. Given an array of integers nums, half of the integers in nums are odd, and the other half are even.

Code:

```
def sort(nums):
    odd=[]
    even=[]
    result=[]
    for num in nums:
        if num%2==0:
            even.append(num)
        else:
            odd.append(num)
    for i in range(len(nums)):
        if i%2==0:
            result.append(even.pop())
        else:
            result.append(odd.pop())
    return result
nums=[4,2,5,7]
sorted_nums=sort(nums)
print(sorted_nums)
```

Screenshot for I/O:

A screenshot of a code editor window titled 'DAA.py'. The editor shows the Python code from the previous block. Below the code editor, there is a terminal window. The terminal shows the command 'python -u "d:\SSE\CSA0672 Design and Analysis of Algorithms\code\DAA.py"' being executed, followed by the output '[2, 7, 4, 5]' and a prompt 'PS c:\Users\User>'. The status bar at the bottom indicates 'Ln 3, Col 12', 'Spaces: 4', 'UTF-8', 'CRLF', 'Python 3.12.0 64-bit', and 'Go Live'.

Time Complexity: $O(n*m)$