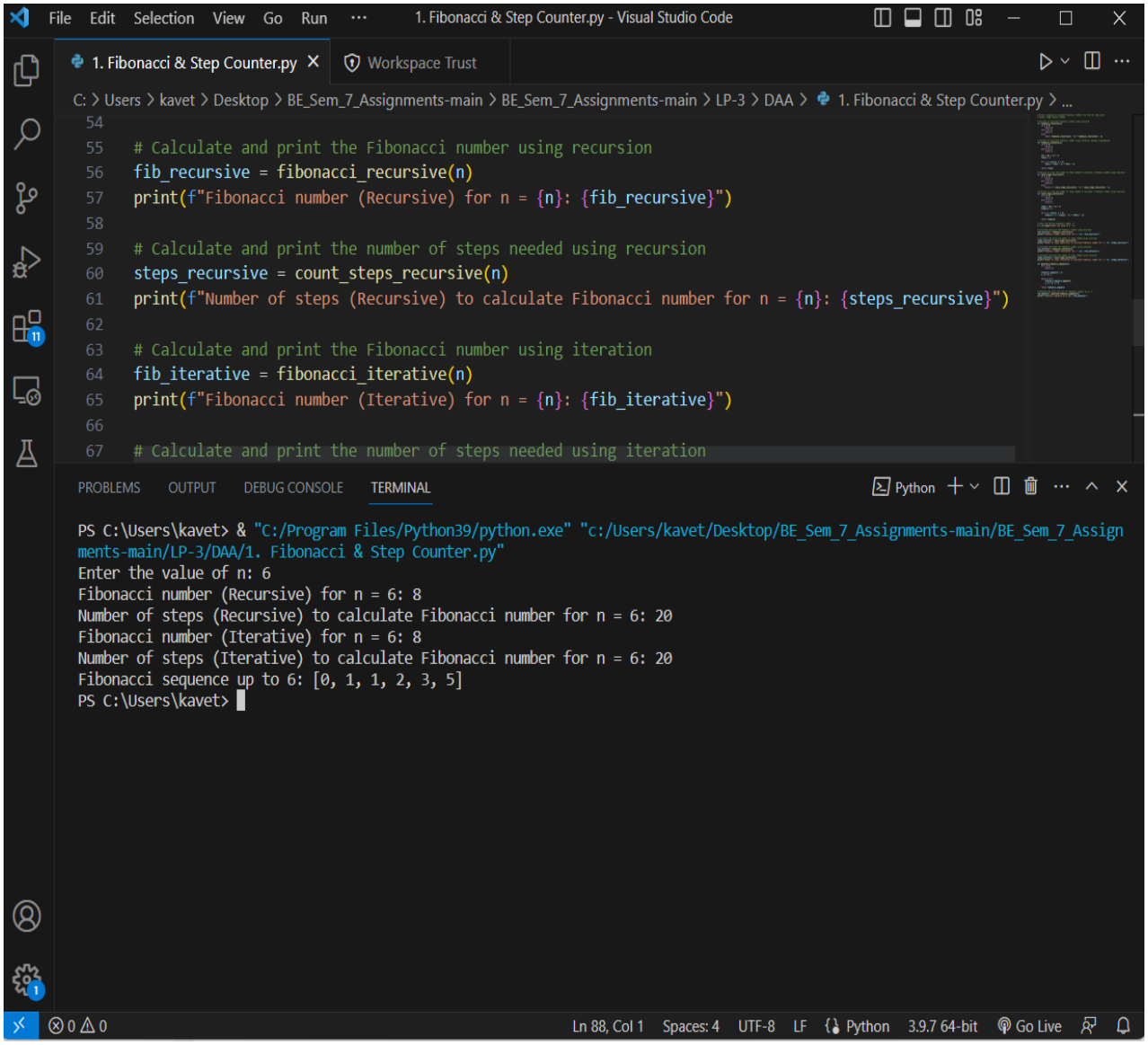
```python
# Write a program to calculate Fibonacci numbers and find its step count
# Function to calculate Fibonacci number using recursion
def fibonacci_recursive(n):
    if n <= 0:
        return 0
    elif n == 1:
        return 1
    else:
        return fibonacci_recursive(n - 1) + fibonacci_recursive(n - 2)

# Function to calculate Fibonacci number using iteration (dynamic programming)
def fibonacci_iterative(n):
    if n <= 0:
        return 0
    elif n == 1:
        return 1

    fib = [0] * (n + 1)
    fib[1] = 1

    for i in range(2, n + 1):
        fib[i] = fib[i - 1] + fib[i - 2]

    return fib[n]

# Function to count the number of steps needed to calculate a Fibonacci number
using recursion
def count_steps_recursive(n):
    if n <= 0:
        return 0
    elif n == 1:
        return 1
    else:
        return 1 + count_steps_recursive(n - 1) + count_steps_recursive(n - 2)

# Function to count the number of steps needed to calculate a Fibonacci number
using iteration
def count_steps_iterative(n):
    if n <= 0:
        return 0
    elif n == 1:
        return 1

    steps = [0] * (n + 1)
    steps[1] = 1

    for i in range(2, n + 1):
        steps[i] = 1 + steps[i - 1] + steps[i - 2]
```

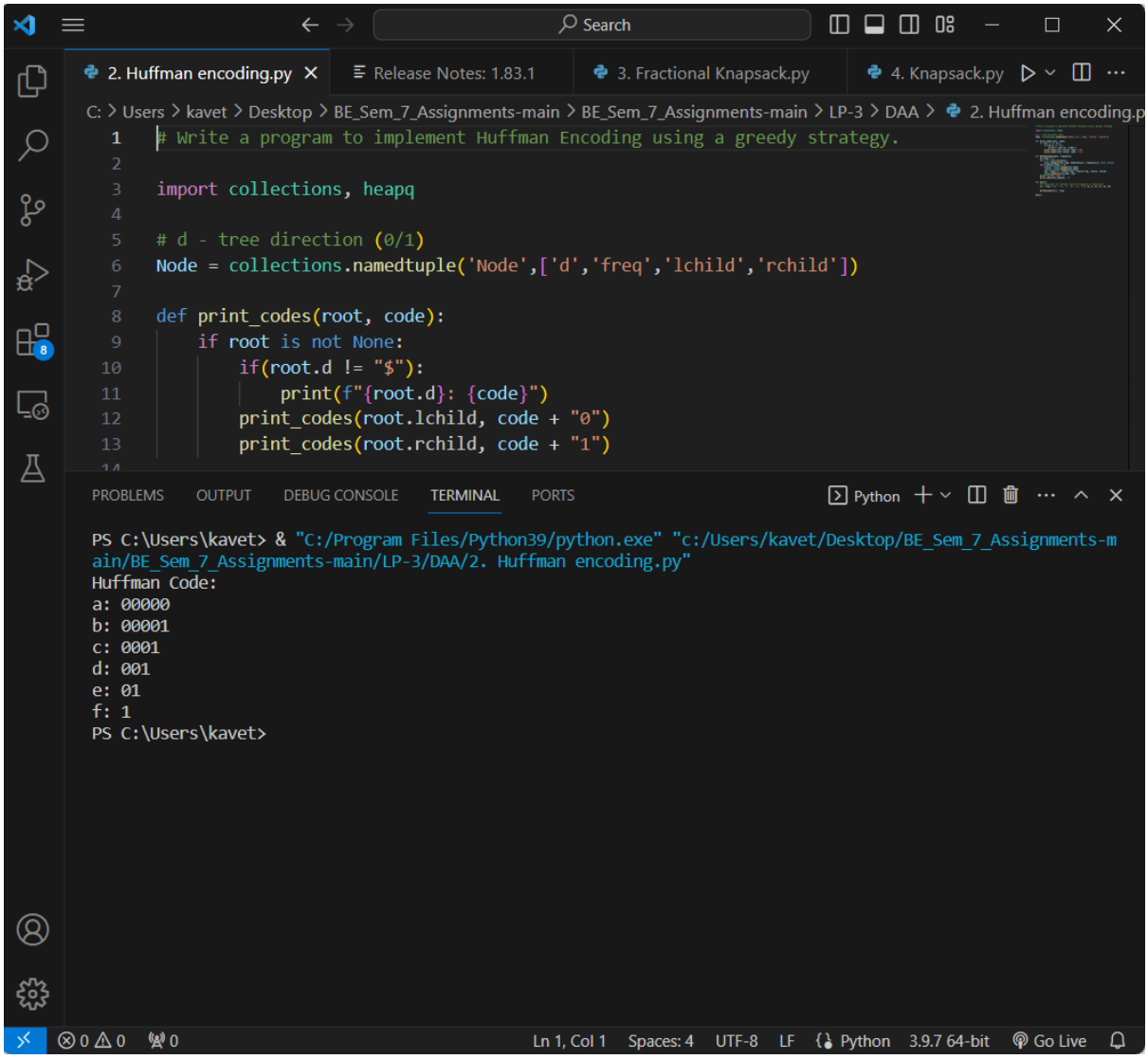
```
    return steps[n]

# Input the desired Fibonacci number 'n'
n = int(input("Enter the value of n: "))

# Calculate and print the Fibonacci number using recursion
fib_recursive = fibonacci_recursive(n)
print(f"Fibonacci number (Recursive) for n = {n}: {fib_recursive}")

# Calculate and print the number of steps needed using recursion
steps_recursive = count_steps_recursive(n)
print(f"Number of steps (Recursive) to calculate Fibonacci number for n = {n}:
{steps_recursive}")

# Calculate and print the Fibonacci number using iteration
fib_iterative = fibonacci_iterative(n)
print(f"Fibonacci number (Iterative) for n = {n}: {fib_iterative}")

# Calculate and print the number of steps needed using iteration
steps_iterative = count_steps_iterative(n)
print(f"Number of steps (Iterative) to calculate Fibonacci number for n = {n}:
{steps_iterative}")

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

    fibonacci_sequence = []
    a, b = 0, 1

    while a <= n:
        fibonacci_sequence.append(a)
        a, b = b, a + b

    return fibonacci_sequence

# Generate and print the list of Fibonacci numbers up to 'n'
fib_sequence = generate_fibonacci_sequence(n)
print(f"Fibonacci sequence up to {n}: {fib_sequence}")
```

OUTPUT:


File Edit Selection View Go Run ··· 1. Fibonacci & Step Counter.py - Visual Studio Code
1. Fibonacci & Step Counter.py
Workspace Trust
C: > Users > kavet > Desktop > BE_Sem_7_Assignments-main > BE_Sem_7_Assignments-main > LP-3 > DAA > 1. Fibonacci & Step Counter.py > ...
54
55 # Calculate and print the Fibonacci number using recursion
56 fib_recursive = fibonacci_recursive(n)
57 print(f"Fibonacci number (Recursive) for n = {n}: {fib_recursive}")
58
59 # Calculate and print the number of steps needed using recursion
60 steps_recursive = count_steps_recursive(n)
61 print(f"Number of steps (Recursive) to calculate Fibonacci number for n = {n}: {steps_recursive}")
62
63 # Calculate and print the Fibonacci number using iteration
64 fib_iterative = fibonacci_iterative(n)
65 print(f"Fibonacci number (Iterative) for n = {n}: {fib_iterative}")
66
67 # Calculate and print the number of steps needed using iteration
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Python
PS C:\Users\kavet> & "C:/Program Files/Python39/python.exe" "c:/Users/kavet/Desktop/BE_Sem_7_Assignments-main/BE_Sem_7_Assignments-main/LP-3/DAA/1. Fibonacci & Step Counter.py"
Enter the value of n: 6
Fibonacci number (Recursive) for n = 6: 8
Number of steps (Recursive) to calculate Fibonacci number for n = 6: 20
Fibonacci number (Iterative) for n = 6: 8
Number of steps (Iterative) to calculate Fibonacci number for n = 6: 20
Fibonacci sequence up to 6: [0, 1, 1, 2, 3, 5]
PS C:\Users\kavet>
Ln 88, Col 1 Spaces: 4 UTF-8 LF Python 3.9.7 64-bit Go Live

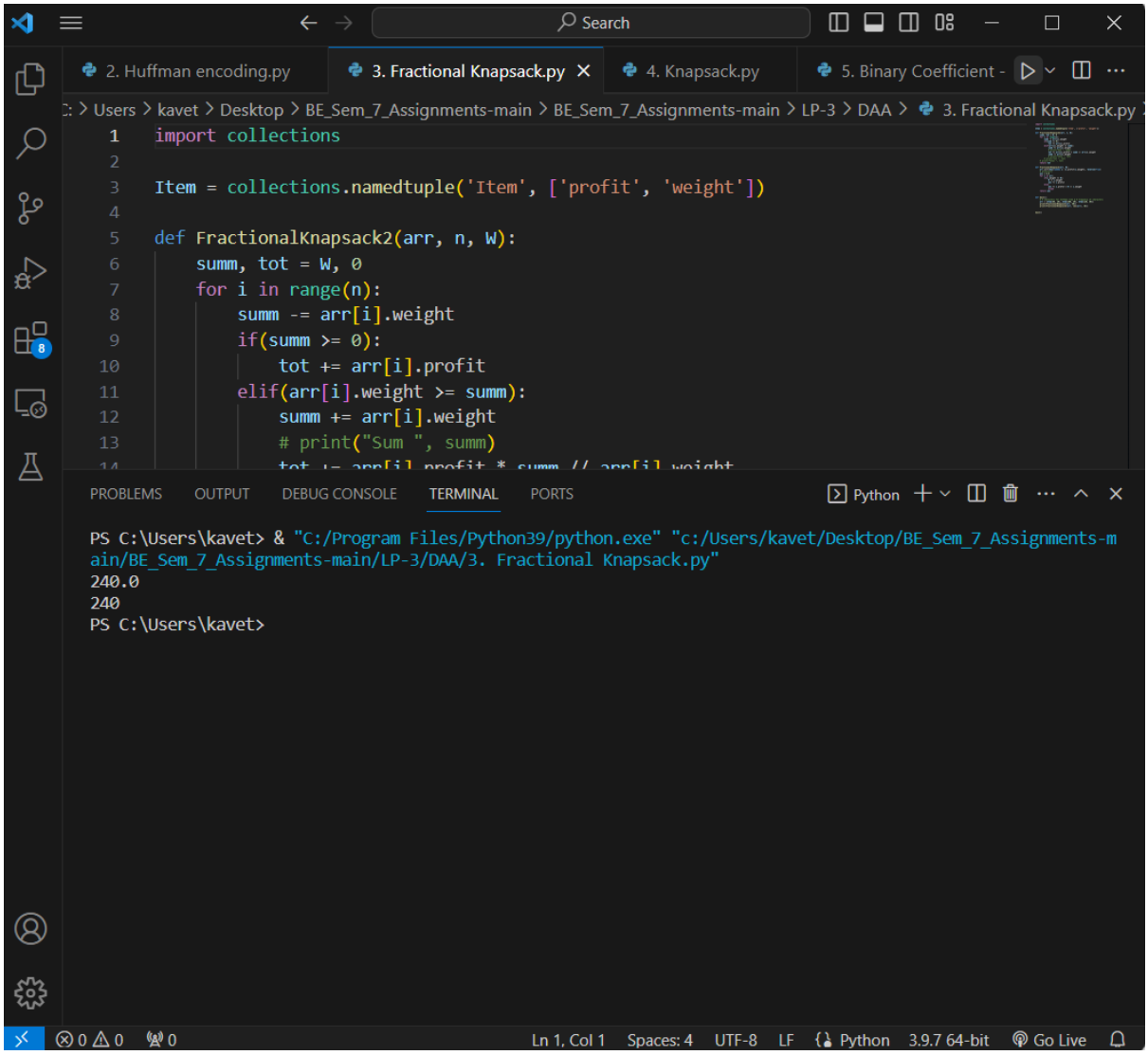
```python
# Write a program to implement Huffman Encoding using a greedy strategy.

import collections, heapq

# d - tree direction (0/1)
Node = collections.namedtuple('Node',['d','freq','lchild','rchild'])

def print_codes(root, code):
    if root is not None:
        if(root.d != "$"):
            print(f"{root.d}: {code}")
        print_codes(root.lchild, code + "0")
        print_codes(root.rchild, code + "1")

def HuffmanCodes(data, frequency):
    min_heap = []
    for i in range(len(data)):
        heapq.heappush(min_heap, Node(data[i], frequency[i], None, None))
    while(len(min_heap) > 1):
        lchild = heapq.heappop(min_heap)
        rchild = heapq.heappop(min_heap)
        top = Node("$", lchild.freq + rchild.freq, lchild, rchild)
        heapq.heappush(min_heap, top)
    print("Huffman Code: ")
    print_codes(min_heap[0], "")

def main():
    # # characters for huffman tree & # frequency of characters
    arr, freq = ['a', 'b', 'c', 'd', 'e', 'f'], [5, 9, 12, 13, 16, 45]

    HuffmanCodes(arr, freq)

main()
```

OUTPUT:


Search
2. Huffman encoding.py
Release Notes: 1.83.1
3. Fractional Knapsack.py
4. Knapsack.py
C: > Users > kavet > Desktop > BE_Sem_7_Assignments-main > BE_Sem_7_Assignments-main > LP-3 > DAA > 2. Huffman encoding.p
1 # Write a program to implement Huffman Encoding using a greedy strategy.
2
3 import collections, heapq
4
5 # d - tree direction (0/1)
6 Node = collections.namedtuple('Node',['d','freq','lchild','rchild'])
7
8 def print_codes(root, code):
9 if root is not None:
10 if(root.d != "$"):
11 print(f"{root.d}: {code}")
12 print_codes(root.lchild, code + "0")
13 print_codes(root.rchild, code + "1")
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
Python
PS C:\Users\kavet> & "C:/Program Files/Python39/python.exe" "c:/Users/kavet/Desktop/BE_Sem_7_Assignments-main/BE_Sem_7_Assignments-main/LP-3/DAA/2. Huffman encoding.py"
Huffman Code:
a: 00000
b: 00001
c: 0001
d: 001
e: 01
f: 1
PS C:\Users\kavet>
Ln 1, Col 1
Spaces: 4
UTF-8
LF
Python
3.9.7 64-bit
Go Live

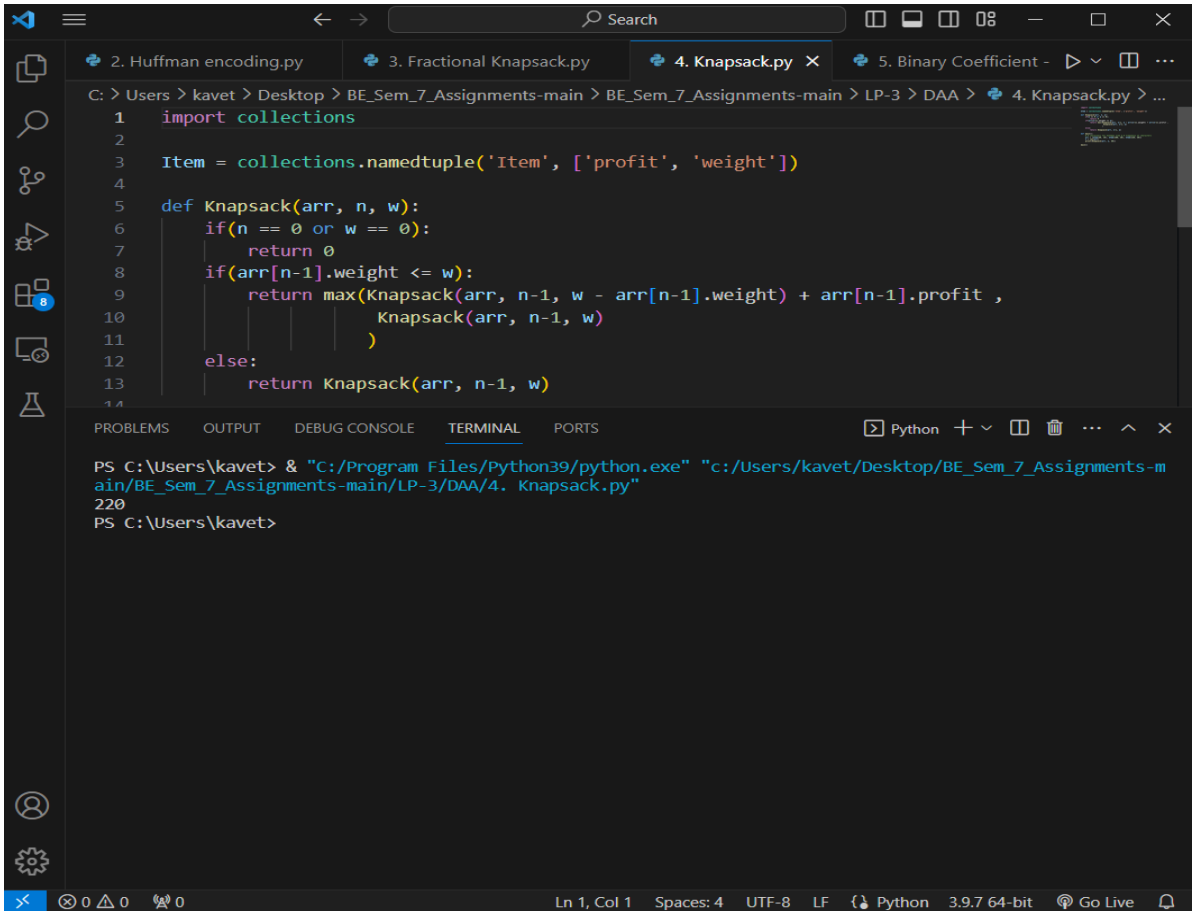
```python
import collections

Item = collections.namedtuple('Item', ['profit', 'weight'])

def FractionalKnapsack2(arr, n, W):
    summ, tot = W, 0
    for i in range(n):
        summ -= arr[i].weight
        if(summ >= 0):
            tot += arr[i].profit
        elif(arr[i].weight >= summ):
            summ += arr[i].weight
            # print("Sum ", summ)
            tot += arr[i].profit * summ // arr[i].weight
            summ -= arr[i].weight
            # print("Tota ", tot)
        # print("tot ", tot)
    # print("Total ", tot)
    return tot

def FractionalKnapsack(arr, W):
    arr.sort(key=lambda x: (x.profit/x.weight), reverse=True)
    # print(arr)
    ans = 0.0
    for i in arr:
        if(i.weight <= W):
            W -= i.weight
            ans += i.profit
        else:
            ans += i.profit * W // i.weight
            break
    return ans


def main():
    # # characters for huffman tree & # frequency of characters
    arr = [Item(60, 10), Item(100, 20), Item(120, 30)]
    print(FractionalKnapsack(arr, 50))
    print(FractionalKnapsack2(arr, len(arr), 50))


main()
```

OUTPUT:


Search
2. Huffman encoding.py
3. Fractional Knapsack.py
4. Knapsack.py
5. Binary Coefficient -
C: > Users > kavet > Desktop > BE_Sem_7_Assignments-main > BE_Sem_7_Assignments-main > LP-3 > DAA > 3. Fractional Knapsack.py
1 import collections
2
3 Item = collections.namedtuple('Item', ['profit', 'weight'])
4
5 def FractionalKnapsack2(arr, n, W):
6 summ, tot = W, 0
7 for i in range(n):
8 summ -= arr[i].weight
9 if(summ >= 0):
10 tot += arr[i].profit
11 elif(arr[i].weight >= summ):
12 summ += arr[i].weight
13 # print("Sum ", summ)
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
Python
PS C:\Users\kavet> & "C:/Program Files/Python39/python.exe" "c:/Users/kavet/Desktop/BE_Sem_7_Assignments-main/BE_Sem_7_Assignments-main/LP-3/DAA/3. Fractional Knapsack.py"
240.0
240
PS C:\Users\kavet>
Ln 1, Col 1
Spaces: 4
UTF-8
LF
Python
3.9.7 64-bit
Go Live

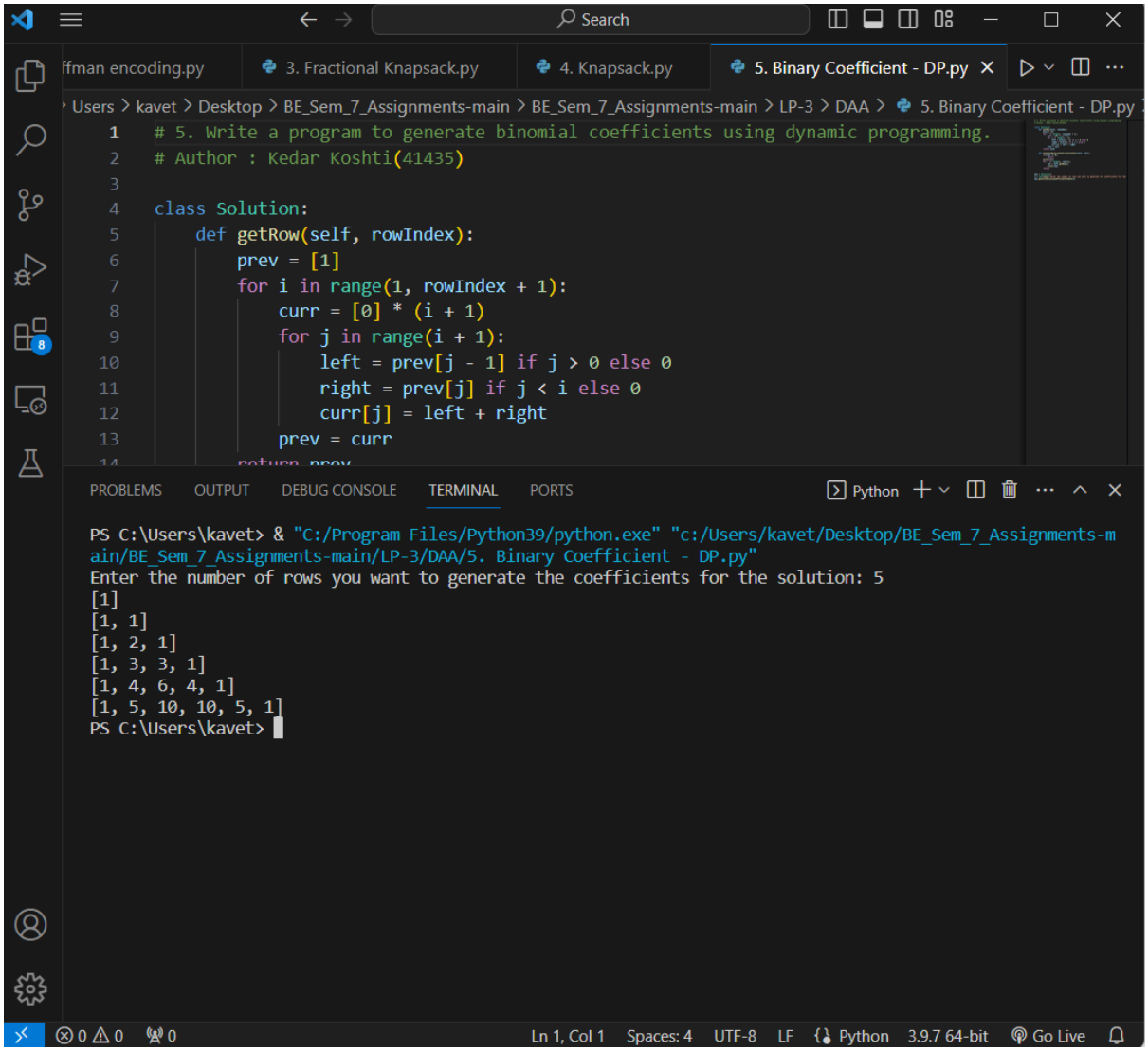
```python
import collections

Item = collections.namedtuple('Item', ['profit', 'weight'])

def Knapsack(arr, n, w):
    if(n == 0 or w == 0):
        return 0
    if(arr[n-1].weight <= w):
        return max(Knapsack(arr, n-1, w - arr[n-1].weight) + arr[n-1].profit ,
                   Knapsack(arr, n-1, w)
                  )
    else:
        return Knapsack(arr, n-1, w)

def main():
    # # characters for huffman tree & # frequency of characters
    arr = [Item(60, 10), Item(100, 20), Item(120, 30)]
    n = len(arr)
    print(Knapsack(arr, n, 50))

main()
```

OUTPUT:



```python
# 5. Write a program to generate binomial coefficients using dynamic
programming.

class Solution:
    def getRow(self, rowIndex):
        prev = [1]
        for i in range(1, rowIndex + 1):
            curr = [0] * (i + 1)
            for j in range(i + 1):
                left = prev[j - 1] if j > 0 else 0
                right = prev[j] if j < i else 0
                curr[j] = left + right
            prev = curr
        return prev

    def generateBinaryCoefficientsTable(self, rows):
        if(rows <= 0):
            return
        print([1])
        for i in range(1, rows+1):
            row = self.getRow(i)
            print(row)
        return


obj = Solution()
n = int(input("Enter the number of rows you want to generate the coefficients
for the solution: "))
obj.generateBinaryCoefficientsTable(n)
```

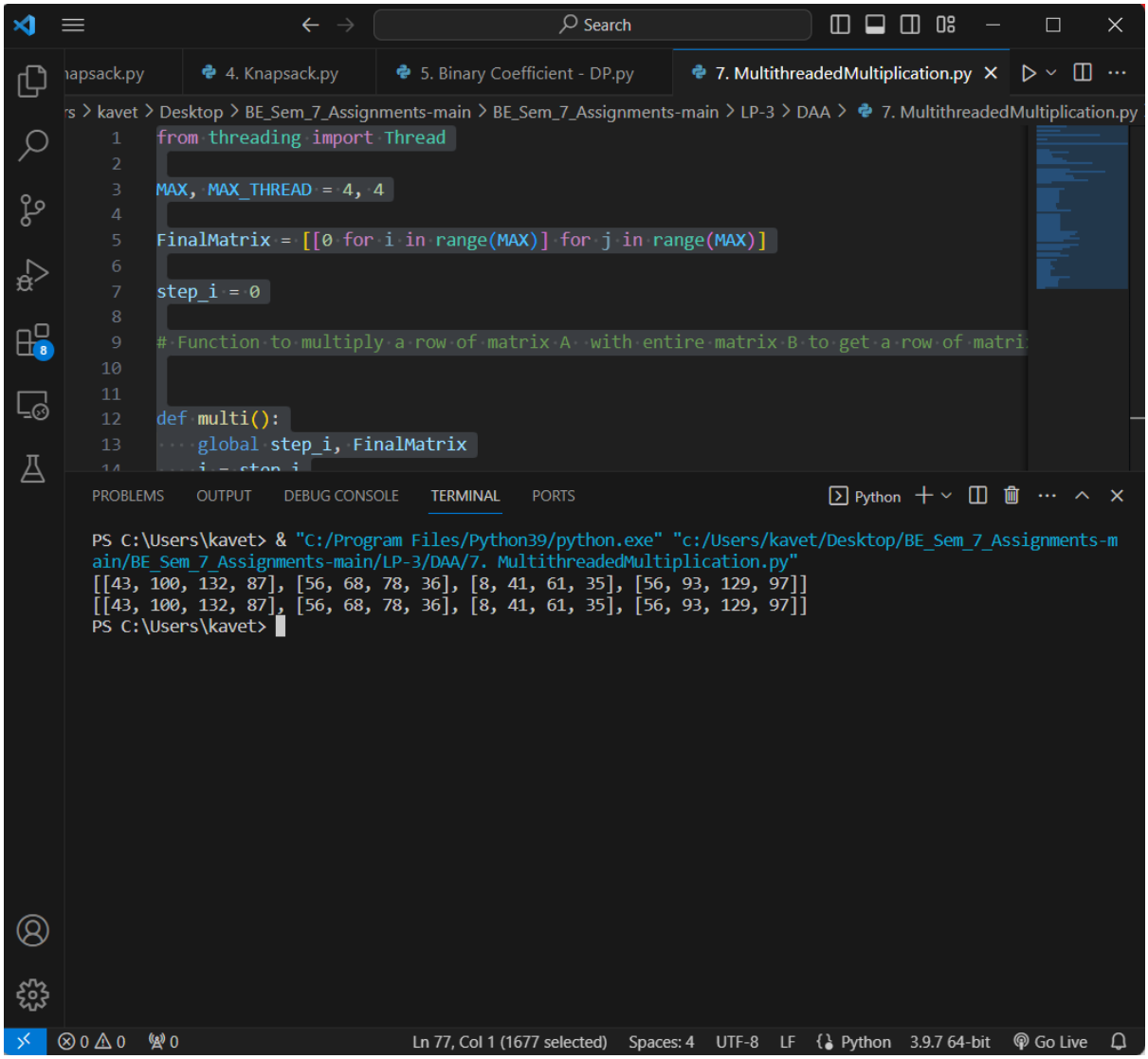
OUTPUT:


Search
ffman encoding.py
3. Fractional Knapsack.py
4. Knapsack.py
5. Binary Coefficient - DP.py
Users > kavet > Desktop > BE_Sem_7_Assignments-main > BE_Sem_7_Assignments-main > LP-3 > DAA > 5. Binary Coefficient - DP.py
# 5. Write a program to generate binomial coefficients using dynamic programming.
# Author : Kedar Koshti(41435)
class Solution:
def getRow(self, rowIndex):
prev = [1]
for i in range(1, rowIndex + 1):
curr = [0] * (i + 1)
for j in range(i + 1):
left = prev[j - 1] if j > 0 else 0
right = prev[j] if j < i else 0
curr[j] = left + right
prev = curr
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
Python
PS C:\Users\kavet> & "C:/Program Files/Python39/python.exe" "c:/Users/kavet/Desktop/BE_Sem_7_Assignments-main/BE_Sem_7_Assignments-main/LP-3/DAA/5. Binary Coefficient - DP.py"
Enter the number of rows you want to generate the coefficients for the solution: 5
[1]
[1, 1]
[1, 2, 1]
[1, 3, 3, 1]
[1, 4, 6, 4, 1]
[1, 5, 10, 10, 5, 1]
PS C:\Users\kavet>
Ln 1, Col 1
Spaces: 4
UTF-8
LF
Python
3.9.7 64-bit
Go Live

```python
from threading import Thread

MAX, MAX_THREAD = 4, 4

FinalMatrix = [[0 for i in range(MAX)] for j in range(MAX)]

step_i = 0

# Function to multiply a row of matrix A  with entire matrix B to get a row of
matrix C


def multi():
    global step_i, FinalMatrix
    i = step_i
    step_i += 1
    for j in range(MAX):
        for k in range(MAX):
            FinalMatrix[i][j] += (A[i][k] * B[k][j])


def normalMultiplication(a, b):
    ans = [[0 for i in range(len(b[0]))] for j in range(len(a))]
    for m in range(len(a)):
        for n in range(len(b[0])):
            for o in range(len(b)):
                ans[m][n] += (a[m][o] * b[o][n])
    print(ans)


if __name__ == "__main__":
    # A = [[5, 4, 3],
    #      [2, 4, 6],
    #      [4, 7, 9]]
    # B = [[3, 2, 4],
    #      [4, 3, 6],
    #      [2, 7, 5]]
    # [37, 43, 59]
    # [34, 58, 62]
    # [58, 92, 103]
    # normalMultiplication(A, B)

    A = [[3, 7, 3, 6],
         [9, 2, 0, 3],
         [0, 2, 1, 7],
         [2, 2, 7, 9]]
    B = [[6, 5, 5, 2],
         [1, 7, 9, 6],
         [6, 6, 8, 9],
```

```
         [0, 3, 5, 2]]
    # creating list of size MAX_THREAD
    thread = list(range(MAX_THREAD))
    for i in range(MAX_THREAD):
        thread[i] = Thread(target=multi)
        thread[i].start()

    # Waiting for all threads to finish
    for i in range(MAX_THREAD):
        thread[i].join()

    print(FinalMatrix)
    normalMultiplication(A, B)

    # A = [[7 2 6 8
    # 7 0 6 6
    # 6 1 7 5
    # 3 7 7 2]]
    # B = 3 5 3 6
    # 5 7 5 8
    # 8 9 4 9
    # 6 5 7 2
    # Multiplication of A and B
    # 127 143 111 128
    # 105 119 87 108
    # 109 125 86 117
    # 112 137 86 141
# main()
```

OUTPUT:


Search
4. Knapsack.py
5. Binary Coefficient - DP.py
7. MultithreadedMultiplication.py
kavet > Desktop > BE_Sem_7_Assignments-main > BE_Sem_7_Assignments-main > LP-3 > DAA > 7. MultithreadedMultiplication.py
from threading import Thread
MAX, MAX_THREAD = 4, 4
FinalMatrix = [[0 for i in range(MAX)] for j in range(MAX)]
step_i = 0
# Function to multiply a row of matrix A with entire matrix B to get a row of matri
def multi():
global step_i, FinalMatrix
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
Python
PS C:\Users\kavet> & "C:/Program Files/Python39/python.exe" "c:/Users/kavet/Desktop/BE_Sem_7_Assignments-main/BE_Sem_7_Assignments-main/LP-3/DAA/7. MultithreadedMultiplication.py"
[[43, 100, 132, 87], [56, 68, 78, 36], [8, 41, 61, 35], [56, 93, 129, 97]]
[[43, 100, 132, 87], [56, 68, 78, 36], [8, 41, 61, 35], [56, 93, 129, 97]]
PS C:\Users\kavet>
Ln 77, Col 1 (1677 selected)
Spaces: 4
UTF-8
LF
Python
3.9.7 64-bit
Go Live