***Assignment – 1***

**Name : Hem Gabhawala**

**Id : 23000858**

Q1

(a) Accept file name from the user and read text from the given file. Find the

frequency of each letter.

(b) Given the above information and a cipher text, which has been encrypted by

Caesar Cipher encryption algorithm, decrypt the given cipher text to get back the

plain text

Code : def letter\_frequency(file\_name):

try:

with open(file\_name, 'r') as file:

text = file.read()

except FileNotFoundError:

print("File not founf. please provide a value")

return

frequency = {}

for char in text:

char = char.lower()

if char in frequency:

frequency[char] += 1

else:

frequency[char] = 1

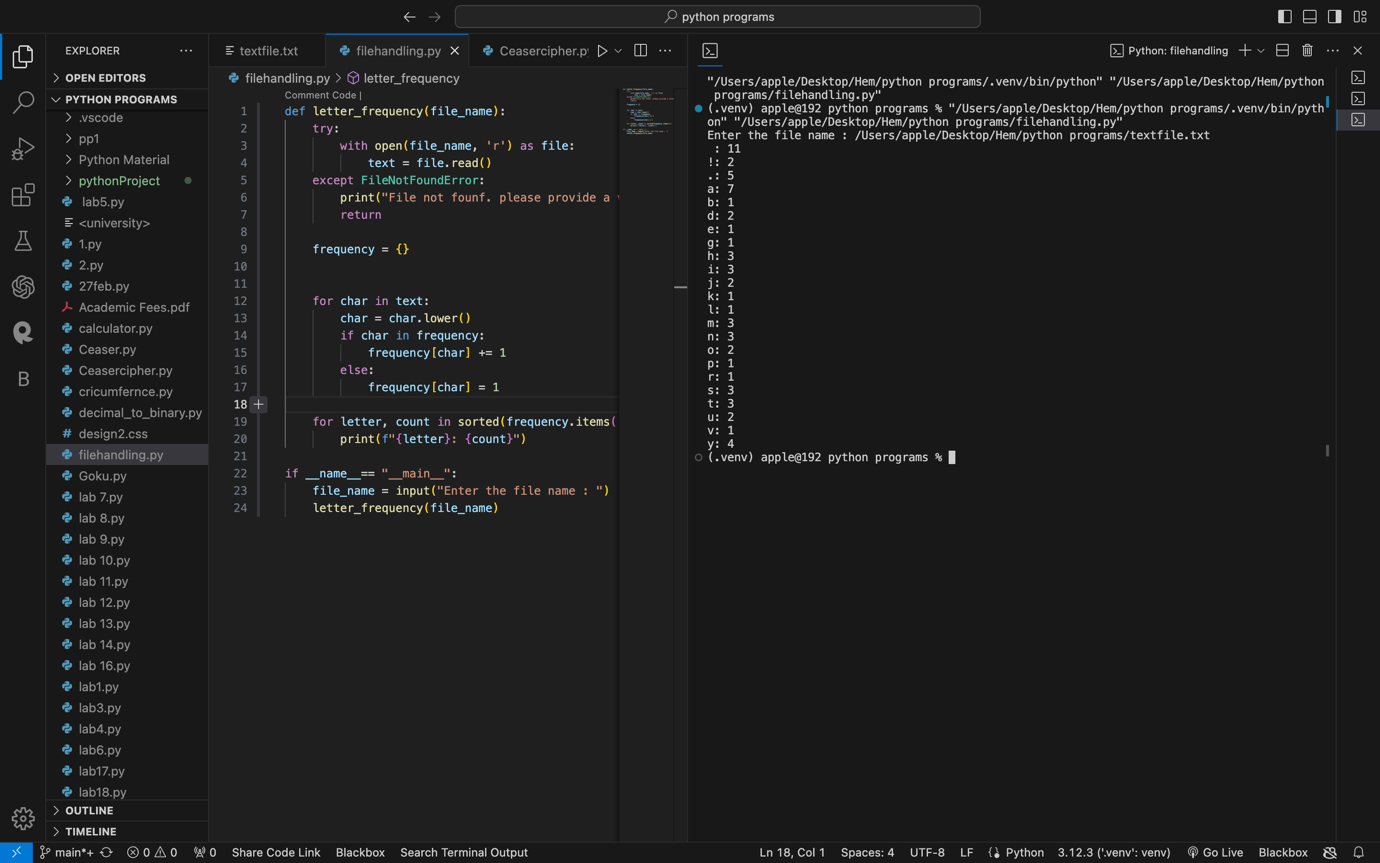
for letter, count in sorted(frequency.items()):

print(f"{letter}: {count}")

if \_\_name\_\_== "\_\_main\_\_":

file\_name = input("Enter the file name : ")

letter\_frequency(file\_name)

Output: 

Code :

Method:1

from typing import Counter

string\_input = input("Enter a string: ").lower()

list = list(string\_input)

print(list)

shifted\_chars = []

for char in list:

if char == ' ':

shifted\_chars.append(' ')

else:

unicode\_value = ord(char)

shifted\_unicode = ((unicode\_value + 3) - 26) if unicode\_value + 3 > 122 else unicode\_value + 3

shifted\_char = chr(shifted\_unicode)

shifted\_chars.append(shifted\_char)

print(shifted\_chars)

shifted\_string = ''.join(shifted\_chars)

print(shifted\_string)

frequency = Counter(shifted\_string)

print(frequency)

frequency\_dict = dict(frequency)

print(frequency\_dict)

sorted\_dict = dict(sorted(frequency\_dict.items(), key=lambda item: item[1], reverse=True))

print(sorted\_dict)

keys\_list = []

for key in sorted\_dict.keys():

keys\_list.append(key)

print(keys\_list)

Letter\_frequencies = ['e','t','a','o','i','n','s','r','h','d','l','u','c','m','f','y','w','g','p','b','v','k','x','q','j','z']

keys\_list = [key for key in sorted\_dict.keys() if key != ' ']

print(keys\_list)

first\_key = keys\_list[0]

first\_frequency = Letter\_frequencies[0]

for i in range(1, len(keys\_list)):

current\_key = keys\_list[i]

current\_frequency = Letter\_frequencies[i]

difference = ord(current\_key) - ord(current\_frequency)

print(abs(difference))

unshifted\_chars = []

for char in shifted\_chars:

if char == ' ':

unshifted\_chars.append(' ')

else:

unicode\_value = ord(char)

reversed\_unicode = ((unicode\_value - abs(difference)) + 26) if unicode\_value - abs(difference) < 97 else unicode\_value - abs(difference);

reversed\_char = chr(reversed\_unicode)

unshifted\_chars.append(reversed\_char)

reversed\_string = ''.join(unshifted\_chars)

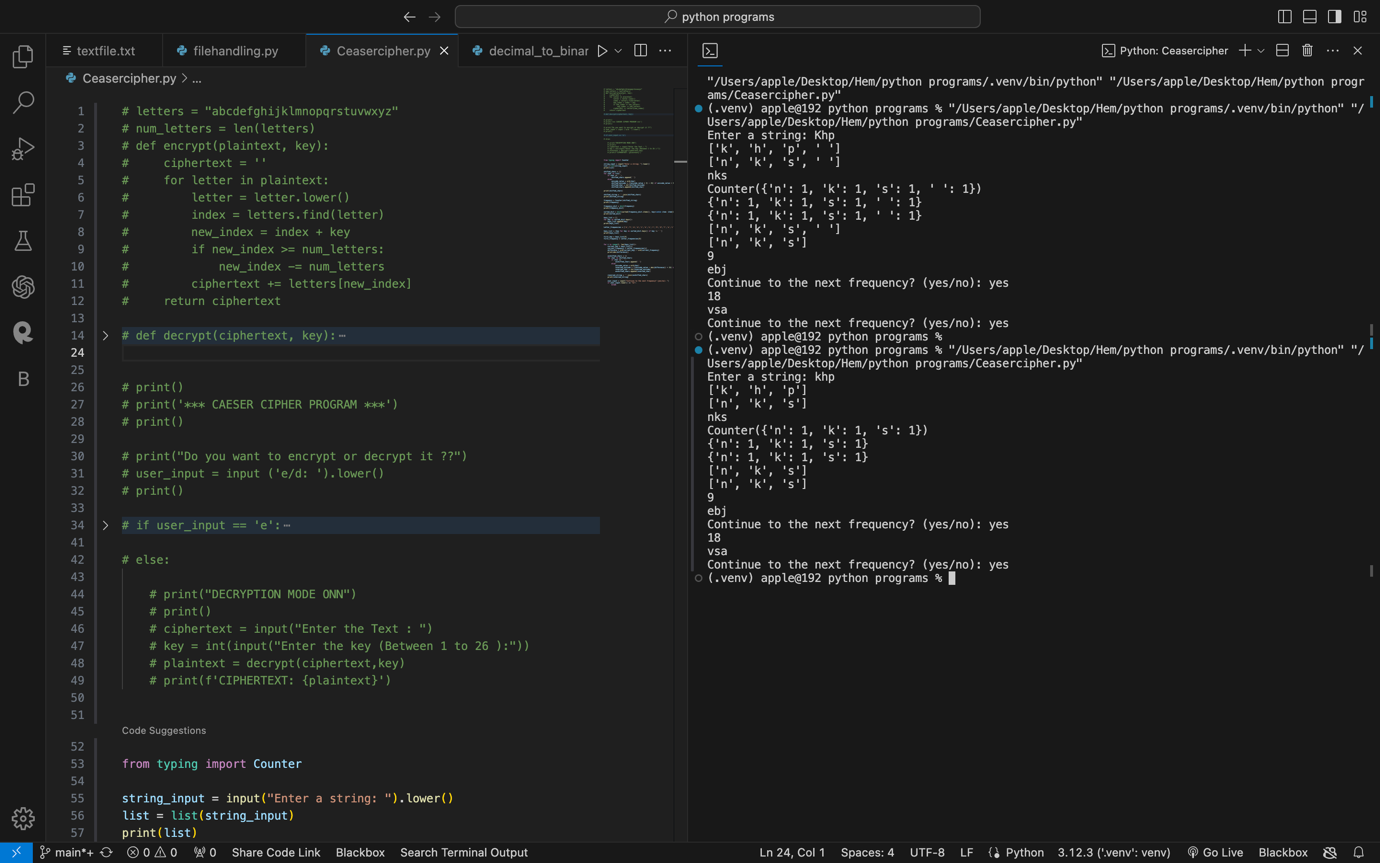
print(reversed\_string)

user\_input = input("Continue to the next frequency? (yes/no): ")

if user\_input.lower() == "no":

break

Output:



Method 2:

Code:

letters = "abcdefghijklmnopqrstuvwxyz"

num\_letters = len(letters)

def encrypt(plaintext, key):

ciphertext = ''

for letter in plaintext:

letter = letter.lower()

index = letters.find(letter)

new\_index = index + key

if new\_index >= num\_letters:

new\_index -= num\_letters

ciphertext += letters[new\_index]

return ciphertext

def decrypt(ciphertext, key):

plaintext = ''

for letter in ciphertext:

letter = letter.lower()

index = letters.find(letter)

new\_index = index - key

if new\_index < 0 :

new\_index += num\_letters

plaintext += letters[new\_index]

return plaintext

print()

print('\*\*\* CAESER CIPHER PROGRAM \*\*\*')

print()

print("Do you want to encrypt or decrypt it ??")

user\_input = input ('e/d: ').lower()

print()

if user\_input == 'e':

print("ENCRYPTION MODE ONN")

print()

plaintext = input("Enter the Text : ")

key = int(input("Enter the key (Between 1 to 26 ):"))

ciphertext = encrypt(plaintext,key)

print(f'CIPHERTEXT: {ciphertext}')

else:

print("DECRYPTION MODE ONN")

print()

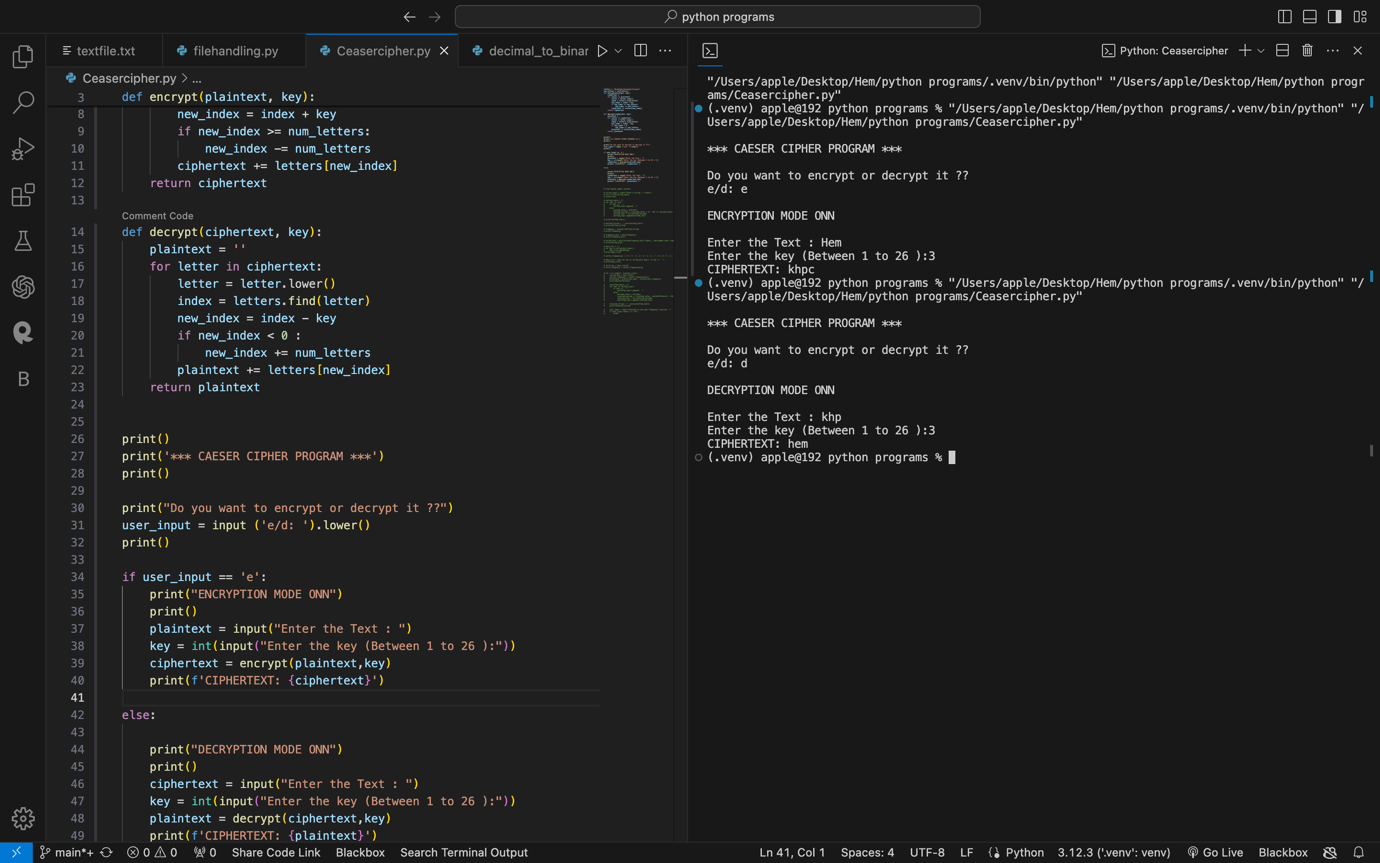
ciphertext = input("Enter the Text : ")

key = int(input("Enter the key (Between 1 to 26 ):"))

plaintext = decrypt(ciphertext,key)

print(f'CIPHERTEXT: {plaintext}')

Output :



Q2:

(a) Accept a Decimal number from the user, the number may be positive or

negative and it may have 0 or more decimal digits. Convert the given number into the

corresponding Binary number.

(b) Accept a positive Binary number from the user. The number may have integral as

well as fractional parts. Convert the given Binary number into the corresponding

Octal number.

Code:

def binarytodecimal(binary):

return int(binary, 2)

if \_\_name\_\_ == '\_\_main\_\_':

# print(binarytodecimal('100'))

# print(binarytodecimal('101'))

# print(binarytodecimal('1001'))

binary = input("Enter your binary Number : ")

decimal = binarytodecimal(binary)

print(f'DECIMALNUMBER : {decimal}')

def decimaltobinary(decimal):

return bin(decimal).replace("0b","")

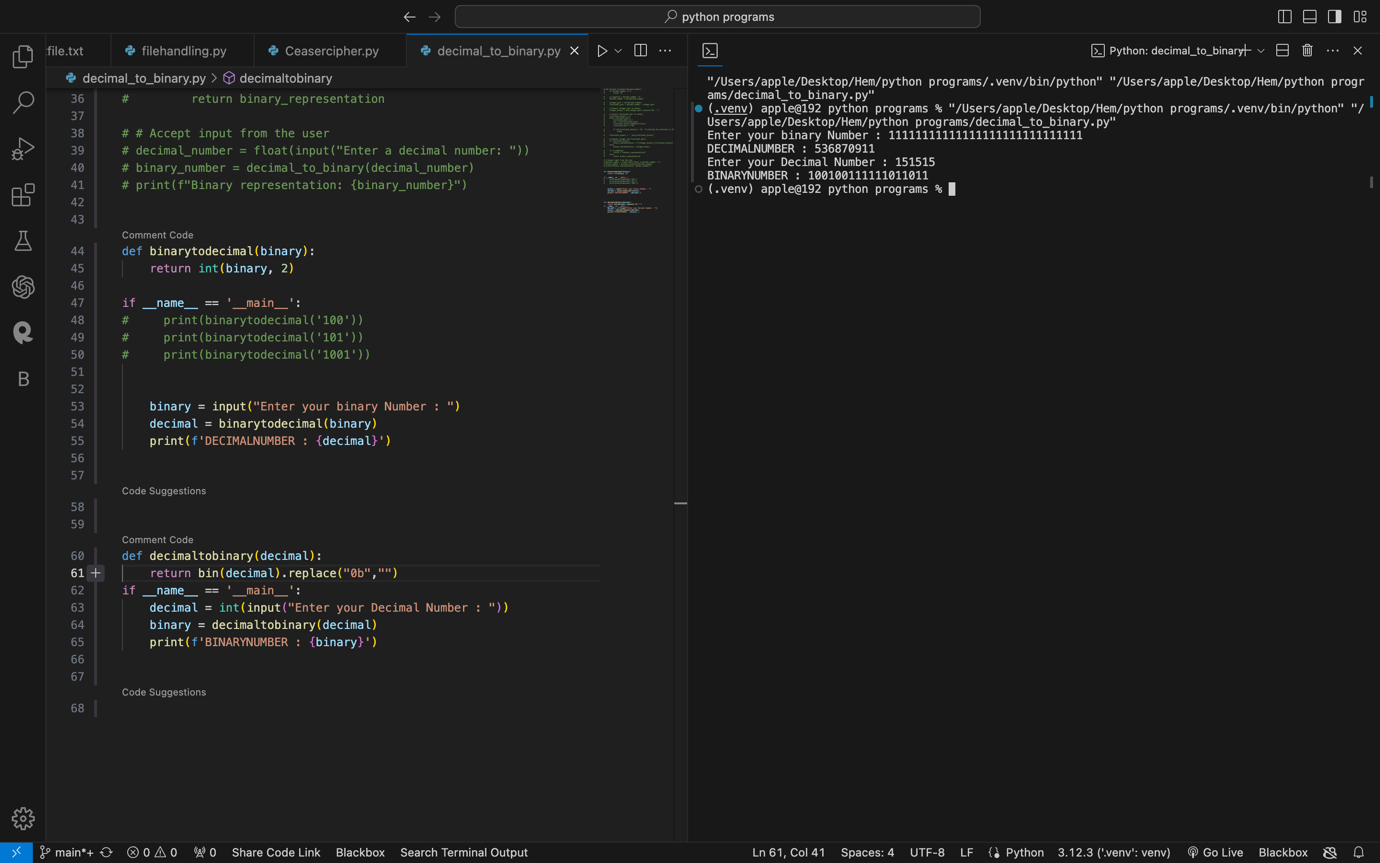
if \_\_name\_\_ == '\_\_main\_\_':

decimal = int(input("Enter your Decimal Number : "))

binary = decimaltobinary(decimal)

print(f'BINARYNUMBER : {binary}')

Output :



Code : def binary\_to\_octal():

binary\_num = input("Enter a positive binary number: ")

# Check if the input is a valid binary number

if not set(binary\_num).issubset({'0', '1', '.'}):

print("Invalid binary number. Please enter a number consisting of 0s, 1s, and an optional decimal point.")

return

# Split the binary number into integral and fractional parts

parts = binary\_num.split('.')

# Convert the integral part to decimal

integral\_part = int(parts[0], 2)

# Convert the fractional part to decimal

fractional\_part = 0

if len(parts) > 1:

fractional\_part = sum([int(bit) \* 2\*\*(-i) for i, bit in enumerate(parts[1], start=1)])

# Combine the integral and fractional parts to get the decimal number

decimal\_num = integral\_part + fractional\_part

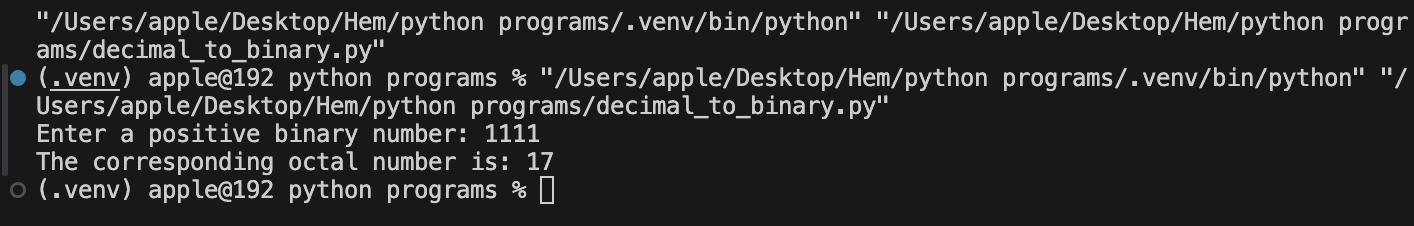
# Convert the decimal number to octal

octal\_num = oct(int(decimal\_num)).replace("0o", "")

print("The corresponding octal number is:", octal\_num)

binary\_to\_octal()

Output:



Q3 –

Accept an amount (from the user) having at least one integral digit and zero or two

decimal digits. Convert the given amount in words: (a) Each digit is converted to

corresponding word, (b) Entire integral part of the number is converted to words and

entire decimal part is also converted to words.

Code : def number\_to\_words(n):

ones = ["", "one", "two", "three", "four", "five", "six", "seven", "eight", "nine"]

tens = ["", "ten", "twenty", "thirty", "forty", "fifty", "sixty", "seventy", "eighty", "ninety"]

teens = ["eleven", "twelve", "thirteen", "fourteen", "fifteen", "sixteen", "seventeen", "eighteen", "nineteen"]

def one(num):

return ones[num]

def two\_less\_20(num):

if num < 10:

return one(num)

elif 10 < num < 20:

return teens[num - 11]

return tens[num // 10] + ('' if num % 10 == 0 else ' ' + one(num % 10))

def two(num):

if num < 10:

return one(num)

elif num < 20:

return two\_less\_20(num)

else:

return tens[num // 10] + ('' if num % 10 == 0 else ' ' + one(num % 10))

def three(num):

hundred = num // 100

rest = num % 100

if hundred == 0:

return two(rest)

else:

if rest == 0:

return one(hundred) + ' hundred'

else:

return one(hundred) + ' hundred and ' + two(rest)

def convert\_number(num):

if num == 0:

return 'zero'

elif num < 100:

return two(num)

elif num < 1000:

return three(num)

return convert\_number(n)

def amount\_to\_words(amount):

integral, decimal = str(amount).split('.')

integral = int(integral)

decimal = int(decimal) if decimal else 0

integral\_words = number\_to\_words(integral)

decimal\_words = ' '.join([number\_to\_words(int(d)) for d in decimal])

return integral\_words + ' point ' + decimal\_words if decimal else integral\_words

def digit\_to\_word(amount):

digits = {'0': 'zero', '1': 'one', '2': 'two', '3': 'three', '4': 'four', '5': 'five', '6': 'six', '7': 'seven', '8': 'eight', '9': 'nine'}

amount\_str = str(amount)

return ' '.join(digits[d] for d in amount\_str)

def main():

amount = input("Enter an amount with at least one integral digit and zero or two decimal digits: ")

try:

amount = float(amount)

if not (amount % 1 == 0 or (len(str(amount).split('.')[1]) in [0, 2])):

raise ValueError("Amount must have zero or two decimal digits.")

print("Each digit in words:", digit\_to\_word(amount))

print("Amount in words:", amount\_to\_words(amount))

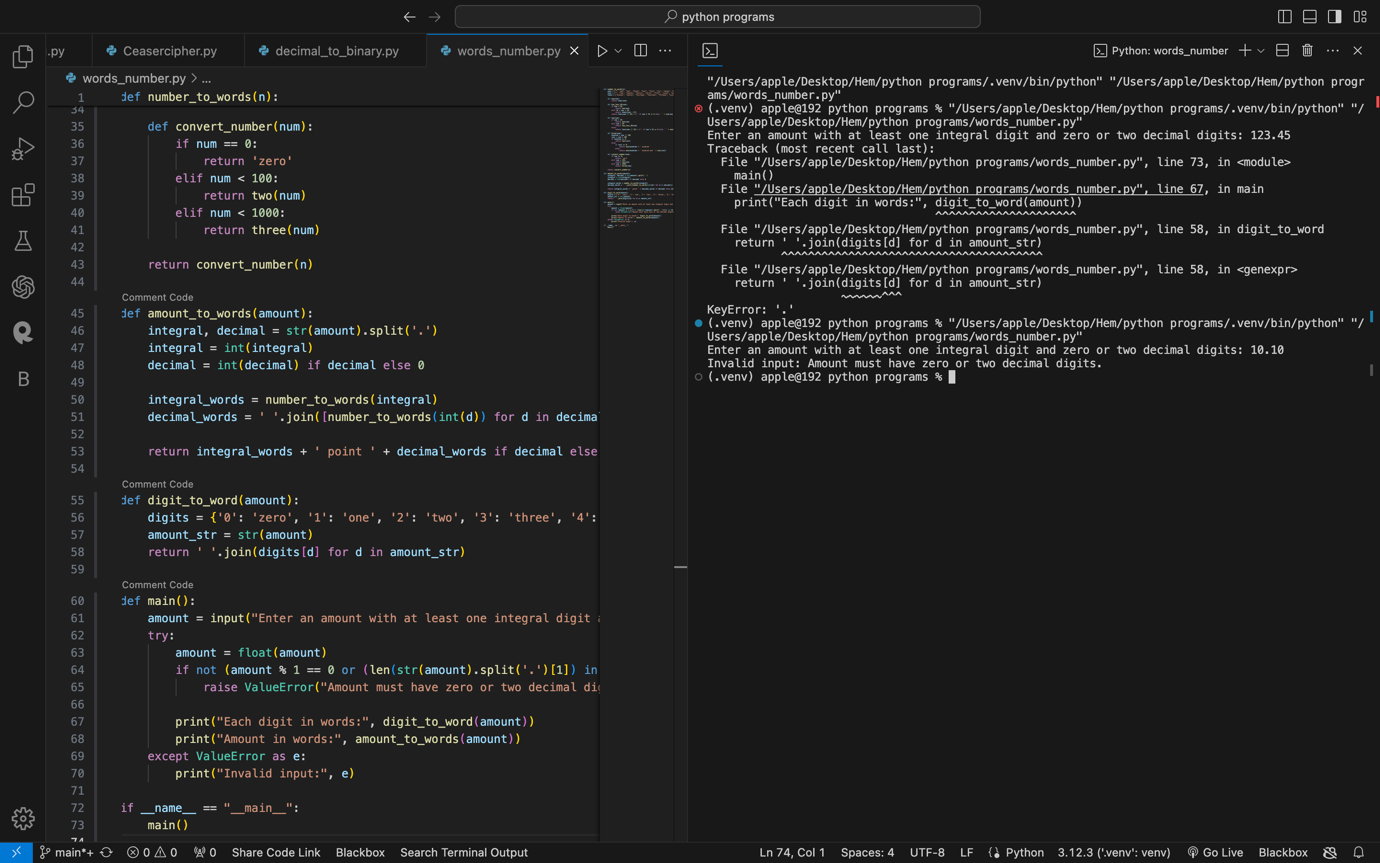
except ValueError as e:

print("Invalid input:", e)

if \_\_name\_\_ == "\_\_main\_\_":

main()

Output:



Q4 –

For a given positive integer, (a) find its divisors, (b) find its prime factors, and (c)

find the 5 co-prime numbers of the given integer.

Code : def find\_divisors(n):

divisors = [i for i in range(1, n + 1) if n % i == 0]

return divisors

def find\_prime\_factors(n):

i = 2

prime\_factors = []

while i \* i <= n:

if n % i:

i += 1

else:

n //= i

prime\_factors.append(i)

if n > 1:

prime\_factors.append(n)

return prime\_factors

def find\_co\_primes(n):

co\_primes = []

for i in range(1, n \* 2):

if gcd(i, n) == 1:

co\_primes.append(i)

if len(co\_primes) == 5:

break

return co\_primes

def gcd(a, b):

while b:

a, b = b, a % b

return a

n = int(input("Enter the Number : "))

Divisors = find\_divisors(n)

PrimeFactors = find\_prime\_factors(n)

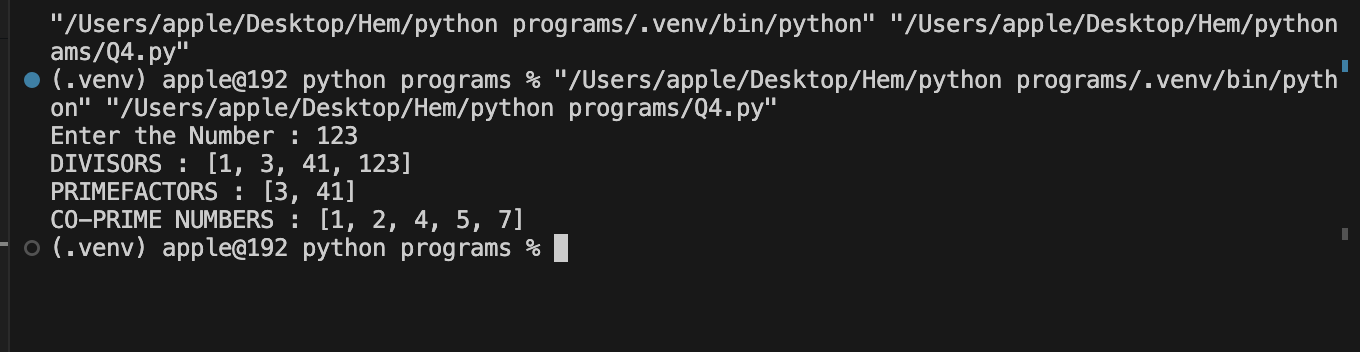
CoPrimeNumbers = find\_co\_primes(n)

print(f'DIVISORS : {Divisors}')

print(f'PRIMEFACTORS : {PrimeFactors}')

print(f'CO-PRIME NUMBERS : {CoPrimeNumbers}')

Output :



Q5 –

(a)In a 3 x 3 matrix arrange numbers 1, 2, 3 such that each row and each column

has different numbers.

(b) Make it general for n x n list for the given n unique numbers.

(c) First write a solution for 9 x 9 matrix with above-mentioned conditions. Each set

of 3 rows will have three 3 x 3 matrices, resulting in a total of nine 3 x 3 matrices.

Check whether each of nine 3 x 3 matrices have different numbers in each row and

each column.

Code :

# matrix = [[1, 2, 3], [3, 1, 2], [2, 3, 1]]

# for row in matrix:

# print(row)

#Qa

def matrix\_find(i,j,k):

matrix = [[i,j,k],[k,i,j],[j,k,i]]

for row in matrix:

print(row)

i = int(input("Enter the Frist Number : "))

j = int(input("Enter the second Number : "))

k = int(input("Enter the third Number : "))

result = matrix\_find(i,j,k)

print(f'FinalMAtrix : {result}')

#Qb

import numpy as np

def arrange\_numbers(n, numbers):

# Create an empty matrix

matrix = np.zeros((n, n), dtype=int)

# Fill the matrix in a cyclic manner

for i in range(n):

for j in range(n):

matrix[i, j] = numbers[(i + j) % n]

return matrix

# Example usage

n = 5

numbers = list(range(1, n+1)) # [1, 2, 3, 4, 5]

matrix = arrange\_numbers(n, numbers)

print(matrix)

#Qc

import numpy as np

def create\_9x9\_matrix():

# Define the 9 unique numbers

numbers = list(range(1, 10))

# Create the 9x9 matrix

matrix = np.zeros((9, 9), dtype=int)

# Fill the matrix in a cyclic manner

for i in range(9):

for j in range(9):

matrix[i, j] = numbers[(i + j) % 9]

return matrix

def check\_3x3\_matrices(matrix):

# Split the 9x9 matrix into 9 3x3 matrices

matrices = []

for i in range(0, 9, 3):

for j in range(0, 9, 3):

matrices.append(matrix[i:i+3, j:j+3])

# Check each 3x3 matrix

for mat in matrices:

if not (len(set(mat[:, 0])) == 3 and len(set(mat[:, 1])) == 3 and len(set(mat[:, 2])) == 3):

return False

if not (len(set(mat[0, :])) == 3 and len(set(mat[1, :])) == 3 and len(set(mat[2, :])) == 3):

return False

return True

# Create the 9x9 matrix

matrix = create\_9x9\_matrix()

# Check the 3x3 matrices

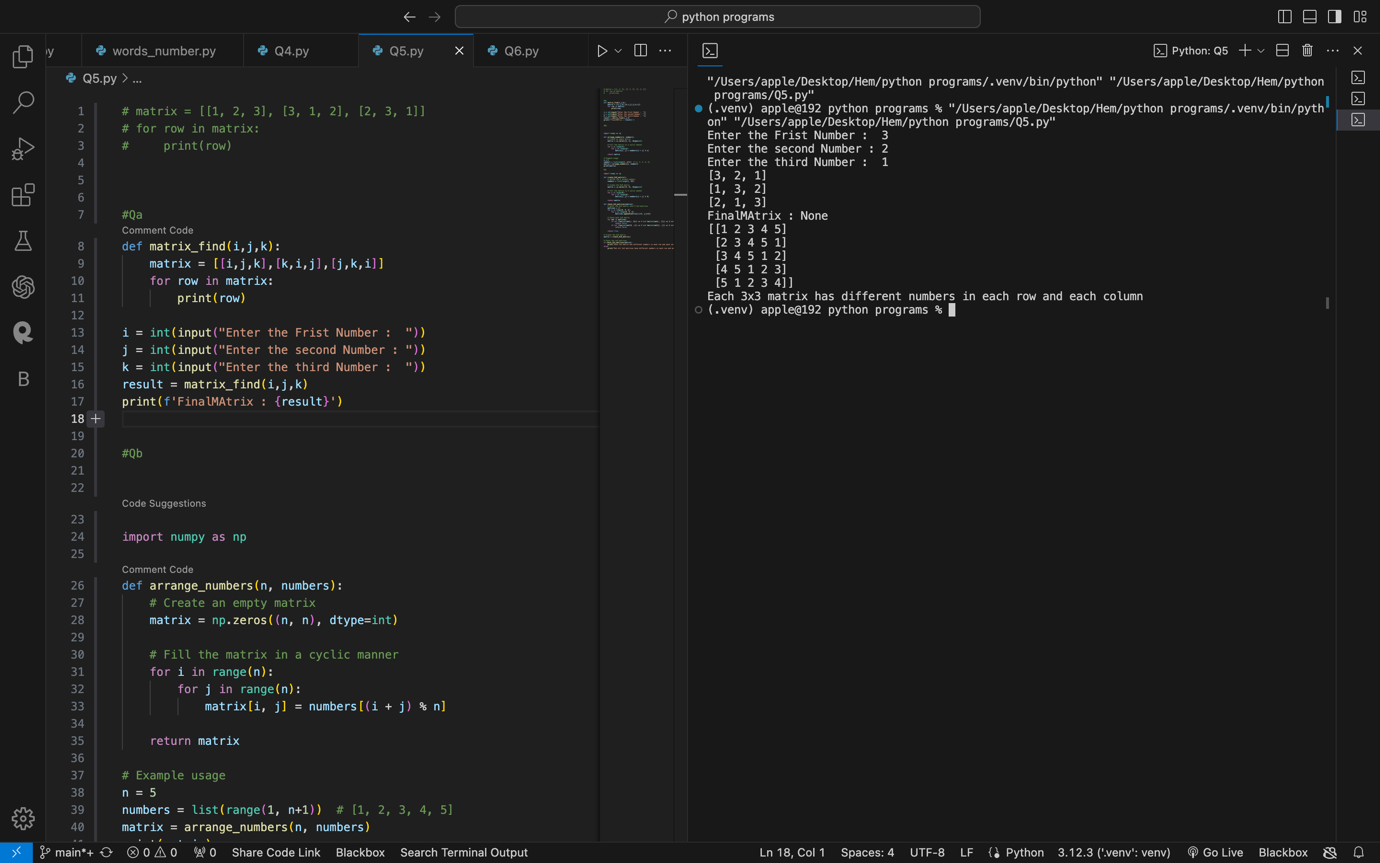
if check\_3x3\_matrices(matrix):

print("Each 3x3 matrix has different numbers in each row and each column")

else:

print("Not all 3x3 matrices have different numbers in each row and each column")

Output:



Q6-

(a) Convert each word (for a digit) to its corresponding digit and find frequency of

each digit.

(b) Draw two histograms using ‘\*’ symbols – one horizontal and one vertical.

Code :

#Qa

import collections

# Define a dictionary to map words to digits

word\_to\_digit = {

'zero': 0, 'one': 1, 'two': 2, 'three': 3, 'four': 4,

'five': 5, 'six': 6, 'seven': 7, 'eight': 8, 'nine': 9

}

# Define a list of words

words = ['one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight', 'nine', 'zero']

# Convert each word to its corresponding digit

digits = [word\_to\_digit[word] for word in words]

# Find the frequency of each digit

frequency = collections.Counter(digits)

# Print the frequency of each digit

for digit, count in frequency.items():

print(f"Digit {digit}: {count}")

#Qb

def draw\_horizontal\_histogram(values):

max\_value = max(values)

for value in values:

print("\*" \* value + " " \* (max\_value - value))

def draw\_vertical\_histogram(values):

max\_value = max(values)

for i in range(max\_value, 0, -1):

for value in values:

if value >= i:

print("\*", end=" ")

else:

print(" ", end=" ")

print()

# Example usage

values = [3, 5, 2, 7, 1, 4]

print("Horizontal Histogram:")

draw\_horizontal\_histogram(values)

print("Vertical Histogram:")

draw\_vertical\_histogram(values)

Output:

