Q1) from math import sqrt

def perimeter\_area(s1, s2, s3):

*# Asserting validity of sides*

    assert s1 + s2 > s3 and s1 + s3 > s2 and s2 + s3 > s1, "Invalid sides"

    perimeter = s1 + s2 + s3

    s = perimeter / 2

    area = sqrt(s \* (s - s1) \* (s - s2) \* (s - s3))

    return perimeter, area

a = float(input("Enter side 1 : "))

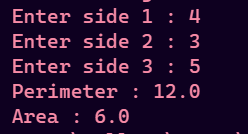
b = float(input("Enter side 2 : "))

c = float(input("Enter side 3 : "))

perimeter,area = perimeter\_area(a,b,c)

print(f"Perimeter : {perimeter}")

print(f"Area : {area}")



Q2)

def salesman\_info(sales\_per\_week):

    salesman\_count = len(sales\_per\_week)

    total\_monthly\_sales = [4\*x for x in sales\_per\_week]

    commissions = [0.05\*x if x > 5000 else 0 for x in total\_monthly\_sales]

    remarks = []

    for sale in total\_monthly\_sales:

        if sale >= 80000:

            remarks.append("Excellent")

        elif sale < 80000 and sale >= 60000:

            remarks.append("Good")

        elif sale < 60000 and sale >= 40000:

            remarks.append("Average")

        else:

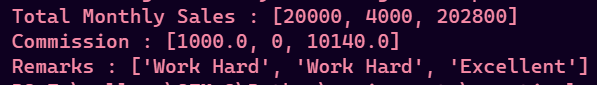
            remarks.append("Work Hard")

    print(f"Total Monthly Sales : {total\_monthly\_sales}")

    print(f"Commission : {commissions}")

    print(f"Remarks : {remarks}")

salesman\_info([5000, 1000, 50700])



Q3)

def factorial(n):

    if n <= 1:

        return n

    return n \* factorial(n - 1)

def fibonacci(n):

    if n <= 1:

        return n

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

def fib\_fac(n):

    result = [fibonacci(n), factorial(n)]

    return result

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

    n = int(input("Enter a number : "))

    print(f"fibonacci and factorial : {fib\_fac(n)}")



Q4)

def get\_digits(n):

    assert n >= 10, "Number should be >= 10"

*# return set(str(n))*

    digits = set()

    str\_num = str(n)

    for digit in str\_num:

        digits.add(int(digit))

    return digits

n = int(input("Enter a number : "))

print(f"Digits : {get\_digits(n)}")



Q5)

from q3 import factorial

def series(x, n):

    result = 1

    for i in range(1, n):

        term = (x \*\* (2\*i)) / factorial(2\*i)

        if i % 2 == 0:

            result += term

        else:

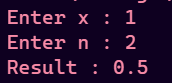
            result -= term

    return result

x = int(input("Enter x : "))

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

print(f"Result : {series(x,n)}")



Q6)

t1 = (1, 2, 5, 7, 9, 2, 4, 6, 8, 10)

*# Print another tuple whose values are even numbers in the given tuple*

l1 = list(t1)

l2 = [x for x in t1 if x % 2 == 0]

t3 = tuple(l2)

print(f"Even tuple : {t3}")

*# Concatenate a tuple t2=(11,13,15) with t1*

t2 = (11, 13, 15)

l1 = list(t1)

l2 = list(t2)

l1.extend(l2)

t1 = tuple(l1)

print(f"New t1 : {t1}")



Q7)

def isPalindrome(s):

    return s == s[::-1]

def replaceString(s):

    result = s

    vowels = "aeiou"

    for vowel in vowels:

        result = result.replace(vowel, "#")

    return result

while True:

    print("------String MENU-----")

    print("1. Length of string")

    print("2. Max of 3 strings")

    print("3. Replace vowels with '#'")

    print("4. Number of words in a string")

    print("5. Checking if string is a palindrome")

    print("6. Exit")

    choice = input("Enter your choice: ")

    if choice == '6':

        print("Exiting program")

        break

    elif choice == '1':

        s = input("Enter your string: ")

        n = len(s)

        print(f"{s} has length {n}")

    elif choice == '2':

        s1 = input("Enter your string 1: ")

        s2 = input("Enter your string 2: ")

        s3 = input("Enter your string 3: ")

        max\_str = max(s1, s2, s3)

        print(f"Max string is {max\_str}")

    elif choice == '3':

        s = input("Enter your string to replace vowels in it : ")

        modified = replaceString(s)

        print(f"{s} is now {modified}")

    elif choice == '4':

        s = input("Enter the string: ")

        words = s.split()

        words\_count = len(words)

        print(f"{s} has {words\_count} words")

    elif choice == '5':

        s = input("Enter the string: ")

        if isPalindrome(s):

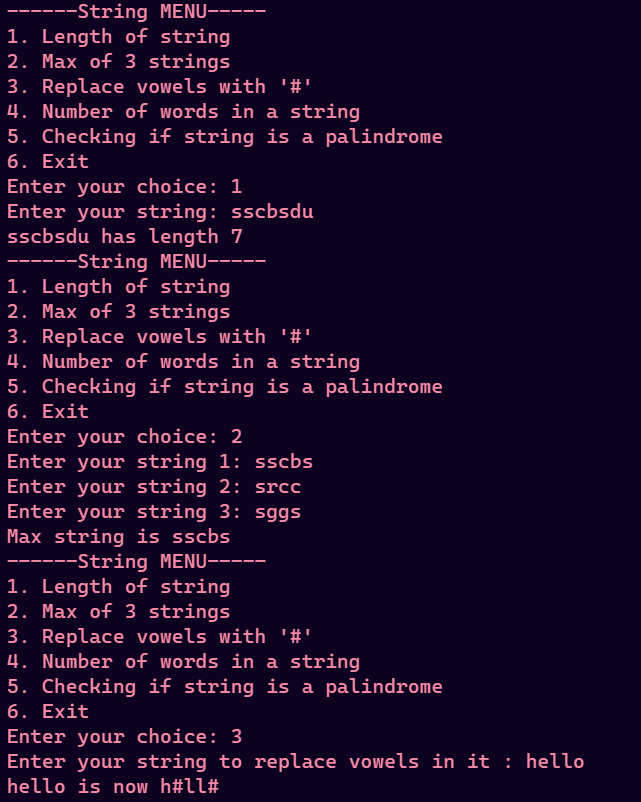
            print(f"{s} is a palindrome.")

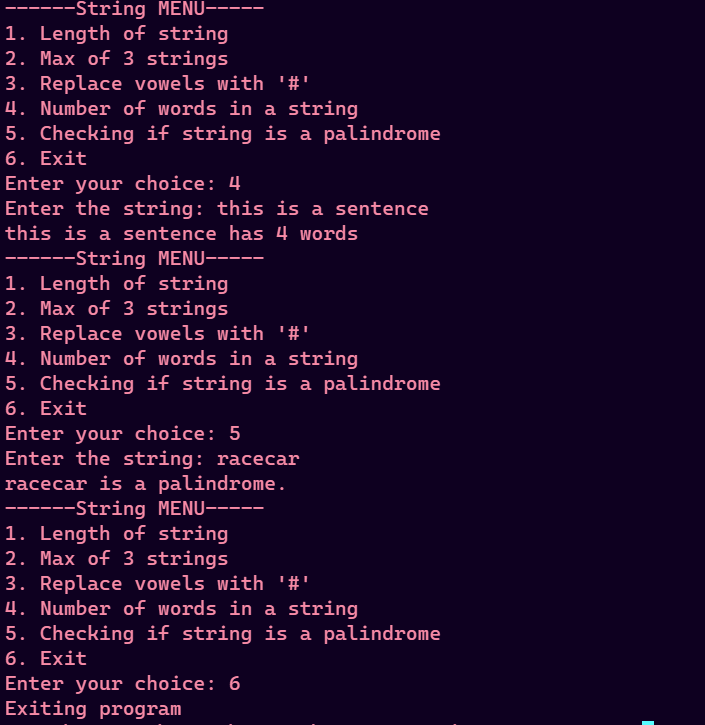
        else:

            print(f"{s} is not a palindrome.")

    else:

        print("Make a valid choice")





Q8)

def isNums(arr):

    for element in arr:

        is\_int = isinstance(element, int)

        is\_float = isinstance(element, float)

        if not (is\_int or is\_float):

            return False

    return True

def isStrings(arr):

    for element in arr:

        if not isinstance(element, str):

            return False

    return True

def countOdds(arr):

    count = 0

    for num in arr:

        if num % 2 != 0:

            count += 1

    return count

def find(arr, x):

    if x not in arr:

        return -1

    return arr.index(x)

def deleteElement(arr, x):

    if x not in arr:

        print(f"Can't remove {x} from list as it doesn't exist")

        return

    arr.remove(x)

    print(f"Removed {x} from list : {arr}")

def common\_members(arr1, arr2):

    common\_elements = []

    for element in arr1:

        if element in arr2 and element not in common\_elements:

            common\_elements.append(element)

    return common\_elements

def input\_list(arr, n):

    for i in range(n):

        element = eval(input(f"Enter {i+1}th element : "))

        arr.append(element)

my\_list\_1 = []

n = int(input("Enter number of elements : "))

input\_list(my\_list\_1, n)

if isNums(my\_list\_1):

    print("It is a numeric list")

    print(f"Odd values : {countOdds(my\_list\_1)}")

if isStrings(my\_list\_1):

    print("It is a string list")

    print(f"Largest string : {max(my\_list\_1)}")

print(f"Reversed list : {my\_list\_1[::-1]}")

x = eval(input("Enter element to be searched : "))

pos = find(my\_list\_1, x)

if pos == -1:

    print(f"{x} not found in list")

else:

    print(f"{x} found in list at {pos}")

if isNums(my\_list\_1) or isStrings(my\_list\_1):

    my\_list\_1.sort(reverse=True)

    print(f"List sorted in descending order : {my\_list\_1}")

else:

    print("Can't sort mixed list")

print("Finding common members")

l1 = []

n1 = int(input("Enter number of elements of list 1 : "))

input\_list(l1, n1)

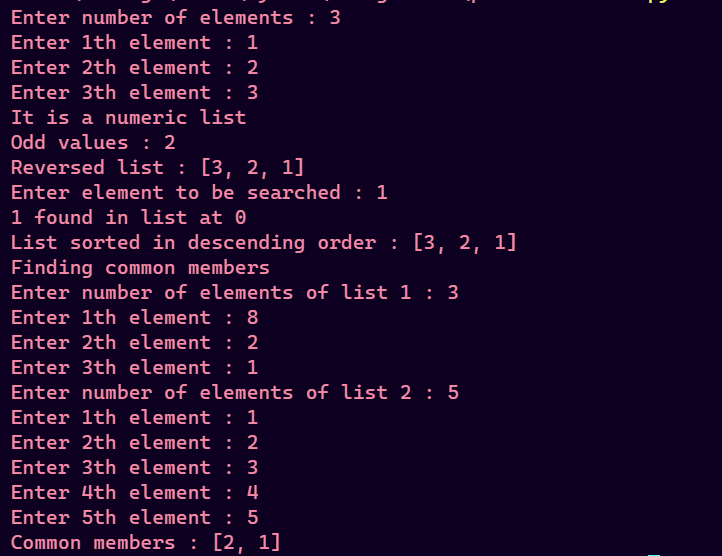
l2 = []

n2 = int(input("Enter number of elements of list 2 : "))

input\_list(l2, n2)

commons = common\_members(l1, l2)

print(f"Common members : {commons}")



Q9)

'''

    marks dict

        - key is student name (str)

        - value is list of 4 marks

'''

student\_marks = {}

def add\_student():

    name = input("Enter name of student : ")

    if name in student\_marks:

        print("Name exist already")

        return

    marks = []

    for i in range(4):

        mark = float(input(f"Enter student marks in subject {i+1} : "))

        marks.append(mark)

*# add record of student*

    student\_marks[name] = marks

def highest\_scorer():

    if len(student\_marks) == 0:

        print("No student")

        return

    highest\_scorer = ""

    highest\_percent = 0

    for student in student\_marks:

        percent = sum(student\_marks[student])/4

        if percent > highest\_percent:

            highest\_percent = percent

            highest\_scorer = student

    print(f"{highest\_scorer} scored highest marks : {highest\_percent}%")

while True:

    print("------MENU--------")

    print("1. Add student")

    print("2. View records")

    print("3. Find highest scorer")

    print("4. Exit")

    choice = input("Make a choice : ")

    if choice == "4":

        print("Exiting program")

        break

    elif choice == "1":

        add\_student()

    elif choice == "2":

        print("Records")

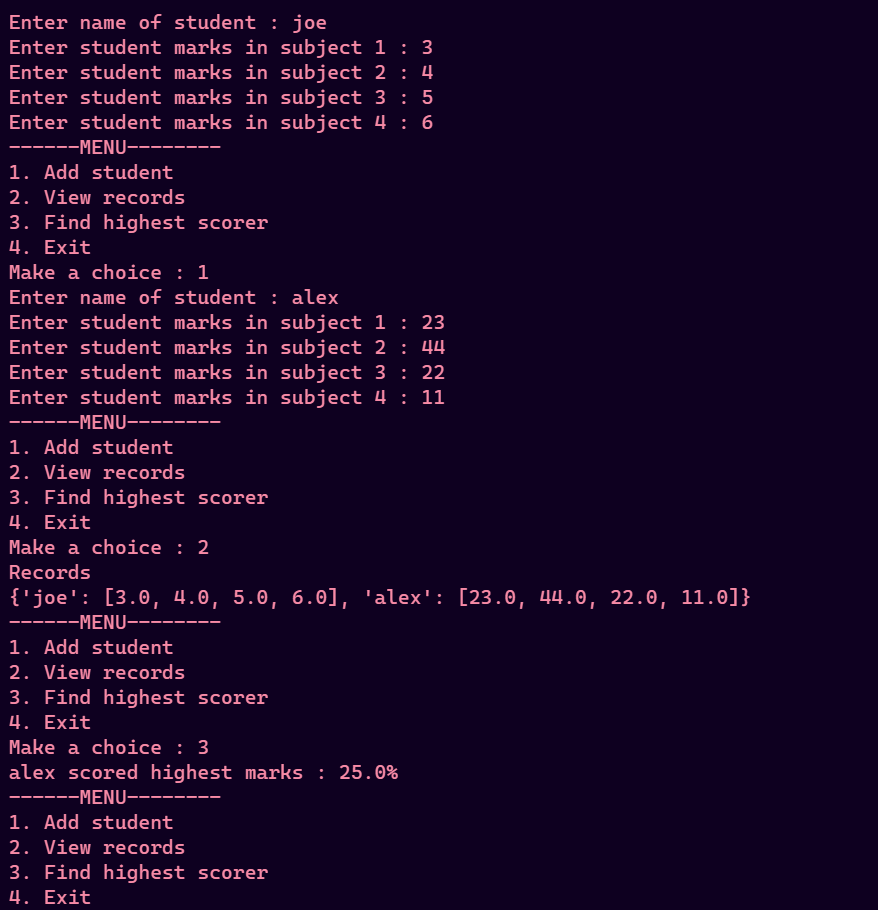
        print(student\_marks)

    elif choice == "3":

        highest\_scorer()

    else:

        print("Make a valid choice")



Q10)

'''

    frequency\_table = {}

        - key is a character a-z

        - value is int no. of times we see it

'''

frequency\_table = {}

sentence = input("Enter sentence\n")

for char in sentence:

    if char not in frequency\_table:

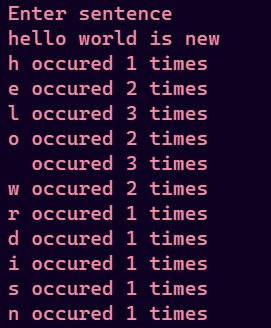
        frequency\_table[char] = 1  *# add first time to table*

    else:

        frequency\_table[char] += 1  *# increment its occurence*

for char in frequency\_table:

    print(f"{char} occured {frequency\_table[char]} times")



Q11)

def bubble\_sort(arr):

    n = len(arr)

    for i in range(n):

        for j in range(n-i-1):

            if arr[j] > arr[j+1]:

                arr[j], arr[j+1] = arr[j+1], arr[j]

def selection\_sort(arr):

    n = len(arr)

    for i in range(n):

        min\_idx = i

        for j in range(i, n):

            if arr[j] < arr[min\_idx]:

                min\_idx = j

        arr[i], arr[min\_idx] = arr[min\_idx], arr[i]

def insertion\_sort(arr):

    n = len(arr)

    for i in range(1, n):

        curr = arr[i]

        j = i - 1

        while j >= 0 and arr[j] > curr:

            arr[j+1] = arr[j]

            j -= 1

        arr[j+1] = curr

def linear\_search(arr, x):

    for i in range(len(arr)):

        if arr[i] == x:

            return i

    return -1

def isSort(arr):

    n = len(arr)

    for i in range(n-1):

        if arr[i] > arr[i+1]:

            return False

    return True

def binary\_search(arr, x):

    left = 0

    right = len(arr)

    while left <= right:

        mid = (left + right) // 2

        if arr[mid] == x:

            return mid

        elif arr[mid] > x:

            right = mid - 1

        else:

            left = mid + 1

    return -1

def input\_array(arr, n):

    for i in range(n):

        element = input(f"Enter {i+1}th student : ")

        arr.append(element)

n = int(input("Enter number of students : "))

students = []

input\_array(students, n)

while True:

    print("---------MENU---------")

    print("1.Linear Search")

    print("2.Binary Search")

    print("3.Bubble Sort")

    print("4.Insertion Sort")

    print("5.Selection Sort")

    print("6.Exit")

    choice = int(input("Enter your choice : "))

    if choice == 6:

        print("Exiting program")

        break

    elif choice == 1:

        student = input("Enter student name to be searched : ")

        pos = linear\_search(students, student)

        if pos == -1:

            print(f"{student} not found")

        else:

            print(f"{student} found at {pos}")

    elif choice == 2:

        if not isSort(students):

            print("Can't perform a binary search on unsorted array")

            continue

        student = input("Enter student name to be searched : ")

        pos = binary\_search(students, student)

        if pos == -1:

            print(f"{student} not found")

        else:

            print(f"{student} found at {pos}")

    elif choice == 3:

        bubble\_sort(students)

        print(f"Bubble Sorted : {students}")

    elif choice == 4:

        insertion\_sort(students)

        print(f"Insertion Sorted : {students}")

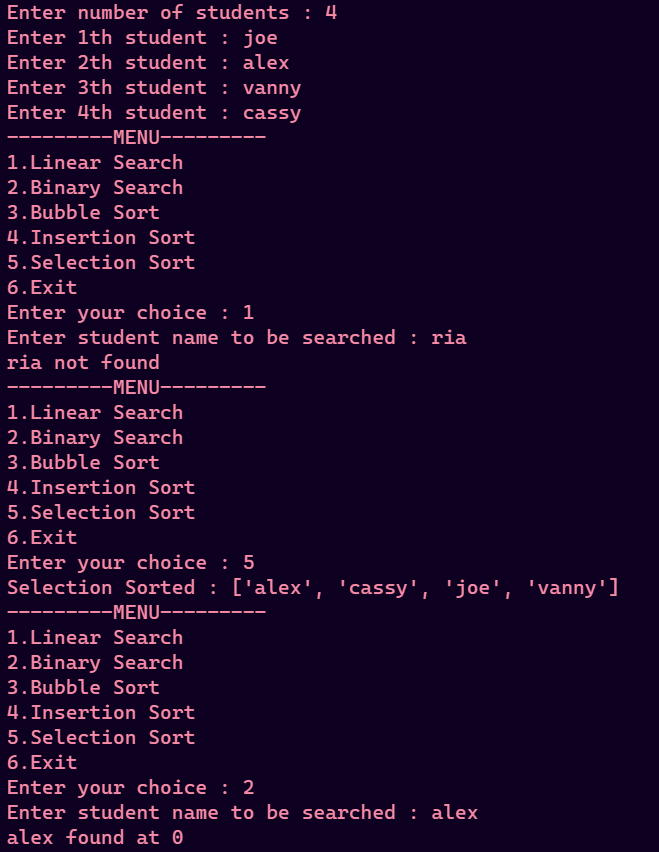
    elif choice == 5:

        selection\_sort(students)

        print(f"Selection Sorted : {students}")

    else:

        print("Make a valid choice")



Q12)

from matplotlib import pyplot as plt

def make\_histogram(nums):

    plt.hist(nums)

    plt.xlabel("Value")

    plt.ylabel("Frequency")

    plt.xlim(min(nums)-1, max(nums)+1)

    plt.title("Histogram")

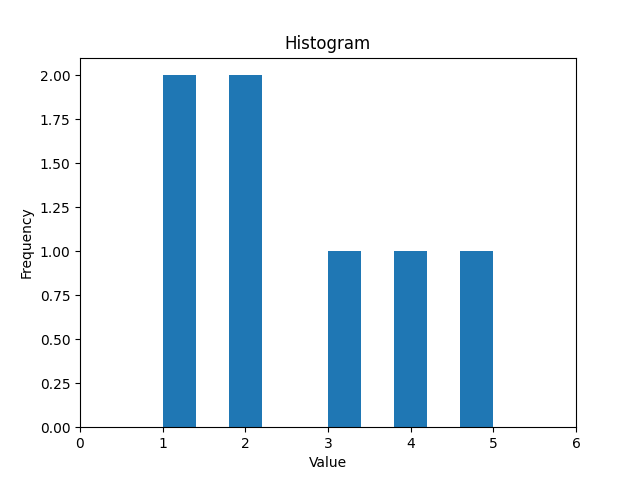
    plt.show()

num\_str = input("Enter space separated numbers to plot histogram\n")

data = num\_str.split()

data = [int(i) for i in data]

make\_histogram(data)



Q13)

from matplotlib import pyplot as plt

from math import sin, cos, exp

def plotFunctions():

    x = range(0, 100)

    y\_sin = [sin(i) for i in x]

    y\_cos = [cos(i) for i in x]

    y\_poly = [i\*\*5 for i in x]

    y\_exp = [exp(i) for i in x]

    plt.subplot(2, 2, 1)

    plt.plot(x, y\_sin)

    plt.xlabel("x")

    plt.ylabel("sin(x)")

    plt.title("x vs sin(x)")

    plt.subplot(2, 2, 2)

    plt.plot(x, y\_cos)

    plt.xlabel("x")

    plt.ylabel("cos(x)")

    plt.title("x vs cos(x)")

    plt.subplot(2, 2, 3)

    plt.plot(x, y\_exp)

    plt.xlabel("x")

    plt.ylabel("exp(x)")

    plt.title("x vs exp(x)")

    plt.subplot(2, 2, 4)

    plt.plot(x, y\_poly)

    plt.xlabel("x")

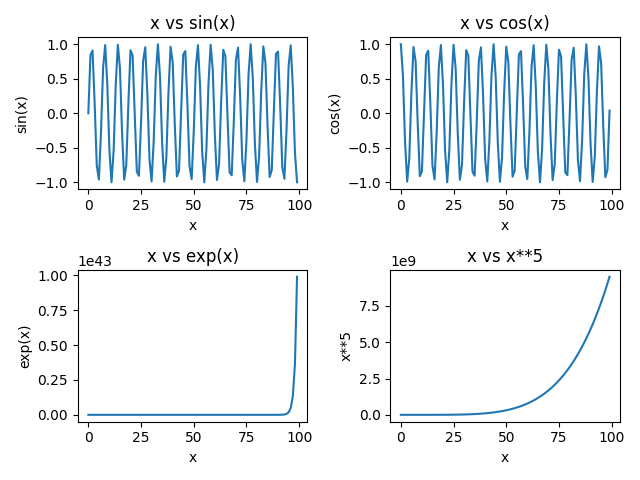
    plt.ylabel("x\*\*5")

    plt.title("x vs x\*\*5")

    plt.tight\_layout()

    plt.show()

plotFunctions()



Q14)

try:

    src = input("Enter name of source file : ")

    dest = input("Enter name of destination file : ")

    with open(src, "r") as f1:

        with open(dest, "w") as f2:

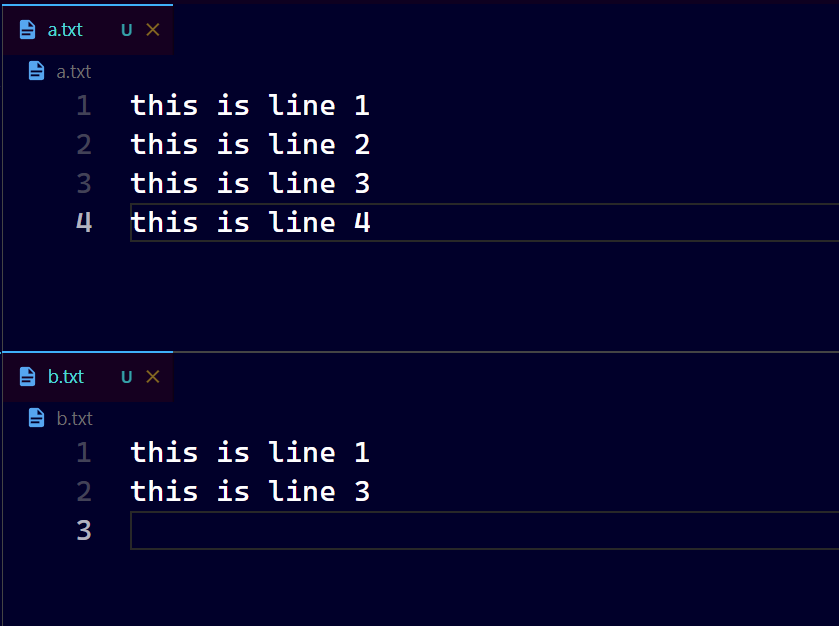
            lines = f1.readlines()

            alternate\_lines = lines[::2]

            f2.writelines(alternate\_lines)

except IOError:

    print("Error opening your files")



Q15)

class Student:

    max\_average = 0

    def \_\_init\_\_(self, name, marks):

*self*.name = name

*self*.marks = marks

*self*.average = sum(marks) / len(marks)

*# Updating CLASS variable max\_average using Student. notation not self.*

        Student.max\_average = max(Student.max\_average, *self*.average)

    def display(self):

        print(f"Name : {*self*.name}")

        print(f"Marks : {*self*.marks}")

        print(f"Average: {*self*.average}")

        print(f"Class Max average : {*self*.max\_average}\n")

george = Student("george", [4, 5, 2])

mili = Student("mili", [14, 5, 2])

corpse = Student("corpse", [4, 15, 2])

george.display()

mili.display()

corpse.display()

