

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **SR.NO** | **LAB** | **Page-No** | **Date** | **Teacher Sign** |
| 1 | Lab-1 | 1 | 03/07/2023 |  |
| 2 |  |  |  |  |
| 3 |  |  |  |  |
| 4 |  |  |  |  |
| 5 |  |  |  |  |
| 6 |  |  |  |  |
| 7 |  |  |  |  |
| 8 |  |  |  |  |
| 9 |  |  |  |  |
| 10 |  |  |  |  |
| 11 |  |  |  |  |
| 12 |  |  |  |  |
| 13 |  |  |  |  |
| 14 |  |  |  |  |
| 15 |  |  |  |  |
| 16 |  |  |  |  |

**Q 1)** Ruchita is looking for her dream job, but has some choices and restrictions. She loves Bangalore and would take a job there if paid over 15,00,000 per year. She hates Hyderabad and demands at least 20,00,000 per year. Any other place she is content to work for 17,00,000 a year, unless she can work in space in which case she would work for free. Write code for the same.

**Code :**

print("Welcome Ruchita to Salary Management")

location = input("Enter your Location : ").upper()

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

if salary >= 1500000 and location == "BANGALORE":

    print("Yes I am ready to work here in ", location)

elif salary >= 2000000 and location == "HYDERABAD":

    print("Yes I am ready to work here in ", location)

elif salary >= 1700000 and (location != "BANGALORE" and location != "HYDERABAD"):

    print("Yes I am ready to work here in ", location)

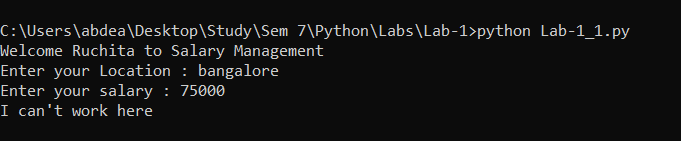
elif salary >= 0 and location == "SPACE":

    print("Yes I am very happy to work in Space")

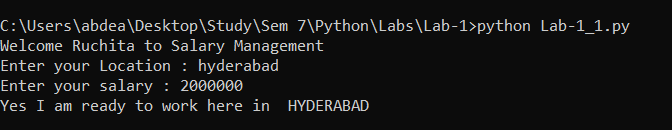
else:

    print("I can't work here")

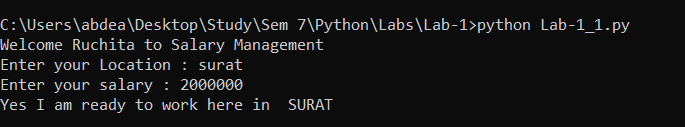
**Output 1.1 :**



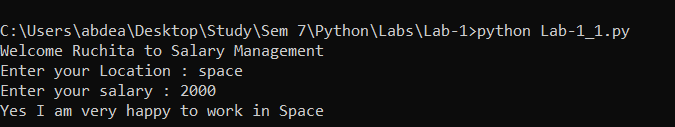
**Output 1.2 :**



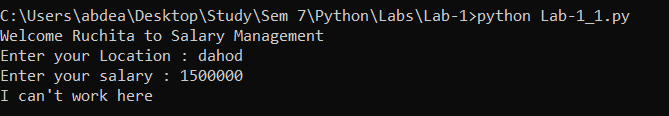
**Output 1.3 :**



**Output 1.4 :**



**Output 1.5 :**



**Q 2)** Given a list L1=[“Test”,”Find”,”Try”,”Search”,”Think”,”Innovate”]

Output = [‘e’,’k’,’h’,’y’,’d’,’t’]

Reverse the list and take only last character of each string in the list.

**Code :**

inputList = ["Test", "Find", "Try", "Search", "Think", "Innovate"]

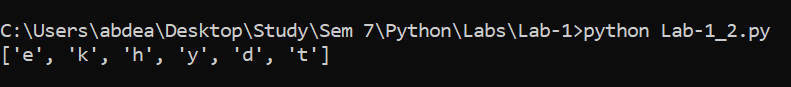
op = []

for x in reversed(inputList):

    op.append(x[len(x) - 1])

print(op)

**Output 2 :**



**Q 3)**  Given a list List1=[10,20,[300,400,[5000,6000],500],30,40]

Output list=[10,20,[300,400,[5000,6000,7000],500],30,40]

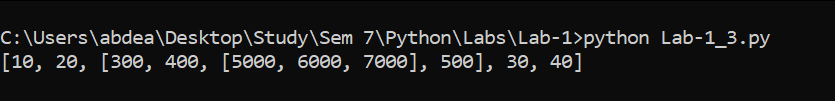
**Code :**

inputList = [10, 20, [300, 400, [5000, 6000], 500], 30, 40]

inputList[2][2].append(7000)

print(inputList)

**Output 3 :**



**Q 4)** Given a string, remove all vowels from the string.

**Code :**

inputString = input("Enter your String : ")

print("\n Before Removing vowels : ")

print(inputString)

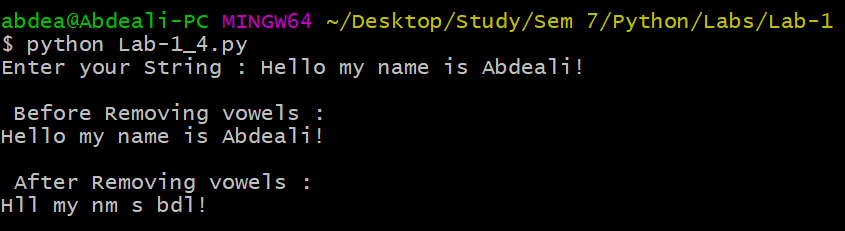
vowels = ['a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O', 'U']

inputString = "".join([ch for ch in inputString if ch not in vowels])

print("\n After Removing vowels : ")

print(inputString)

**Output 4 :**



**Q 5)** Given a list of numbers, return the list containing only square of positive numbers from given list.

**Code :**

inputList = []

N = int(input("Enter total length : "))

for i in range(N):

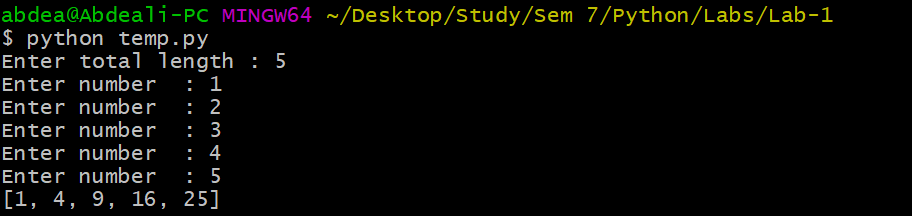
    number = int(input("Enter number  : "))

    inputList.append(number)

inputList = [x \* x for x in inputList]

print(inputList)

**Output 5 :**



**Q 6)** Remove Bug from while loop.

**Code :**

n = 10

i = 10

while i > 0 and n >= 0:

    print(i)

    n = n - 1

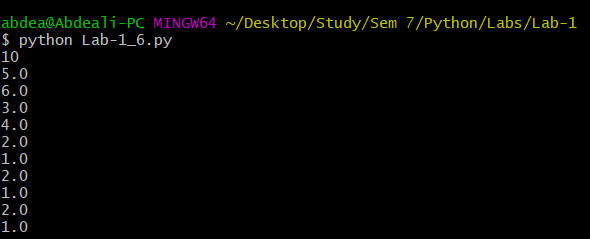
    if i % 2 == 0:

        i = i / 2

    else:

        i = i + 1

**Output 6 :**



**Q 7.1)** You have graduated from MSU and now have a great job! You move to the Bangalore and decide that you want to start saving to buy a house.  As housing prices are very high in the Bangalore, you realize you are going to have to save for several years before you can afford to make the down payment on a house. In Part A, we are going to determine how long it will take you to save enough money to make the down payment given the following assumptions:

1. Call the cost of your dream home total\_cost​.

2. Call the portion of the cost needed for a down payment portion\_down\_payment​. For simplicity, assume that portion\_down\_payment = 0.25 (25%).

3. Call the amount that you have saved thus far current\_savings​. You start with a current savings of $0.

4. Assume that you invest your current savings wisely, with an annual return of r ​(in other words, at the end of each month, you receive an additional current\_savings\*r/12​ funds to put into your savings – the 12 is because r​ is an annual rate).

 Assume that your investments earn a  return of r = 0.04 (4%).

5. Assume your annual salary is annual\_salary​.

6.Assume you are going to dedicate a certain amount of your salary each month to saving for  the down payment. Call that portion\_saved​. This variable should be in decimal form (i.e. 0.1 for 10%).

7. At the end of each month, your savings will be increased by the return on your investment, plus a percentage of your monthly salary ​(annual salary / 12). Write a program to calculate how many months it will take you to save up enough money for a down payment. You will want your main variables to be floats, so you should cast user inputs to floats.    1 Your program should ask the user to enter the following variables:

1. The starting annual salary (annual\_salary)

2. The portion of salary to be saved (portion\_saved)

3. The cost of your dream home (total\_cost)

**Code :**

print("\t  -------  Welcome to your Future calc -------- \n")

returnOfInvestment = 0.04

returnPortion = returnOfInvestment / 12

portion\_down\_payment = 0.25

annual\_salary = float(input("Enter your annual salary : "))

monthly\_salary = annual\_salary / 12

portion\_saved = float(input("Enter portion you will saved  : "))

portion\_saved\_salary = monthly\_salary \* portion\_saved

total\_cost\_of\_house = float(input("Enter your dream house total cost  : "))

current\_saving = 0

total\_months\_to\_reach = 0

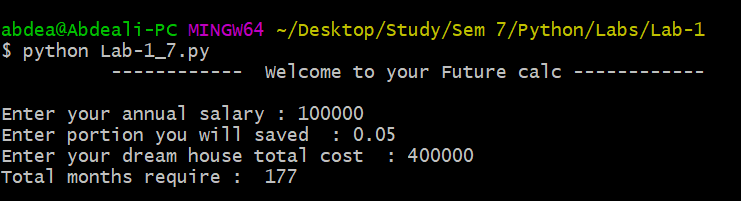
while current\_saving <= (total\_cost\_of\_house \* portion\_down\_payment):

    current\_saving = current\_saving + (current\_saving \* returnPortion) + portion\_saved\_salary

    total\_months\_to\_reach = total\_months\_to\_reach + 1

print("Total months require : ", total\_months\_to\_reach)

**Output 7 .1 :**



**Q 7.2)** In Part A, we unrealistically assumed that your salary didn’t change.  But clearly you are going to be orth more to your company over time! So we are going to build on your solution to Part A by factoring in a raise every six months. Modify your program to include the following :

1. Have the user input a semi-annual salary raise semi\_annual\_raise​ (as a decimal percentage)

2. After the 6th month, increase your salary by that percentage.  Do the same after the 12 month, the 18  month, and so on.  Write a program to calculate how many months it will take you save up enough money for a down\_payment.  LIke before, assume that your investments earn a return of r​ = 0.04 (or 4%) and the required down payment percentage is 0.25 (or 25%).  Have the user enter the following variables:

1. The starting annual salary (annual\_salary)

2. The percentage of salary to be saved (portion\_saved)

3. The cost of your dream home (total\_cost)

4. The semi­annual salary raise (semi\_annual\_raise)

**Code :**

print("\t  ------  Welcome to your Future calc --------- \n")

# constant

returnOfInvestment = 0.04

returnPortion = returnOfInvestment / 12

portion\_down\_payment = 0.25

# input variables

annual\_salary = float(input("Enter your annual salary : "))

portion\_saved = float(input("Enter portion you will saved  : "))

portion\_saved\_salary = (annual\_salary / 12) \* portion\_saved

total\_cost\_of\_house = float(input("Enter your dream house total cost  : "))

down\_payment = total\_cost\_of\_house \* portion\_down\_payment

semi\_annual\_raise = float(input("Enter your raise in decimal (%) : ")) / 100

current\_saving = 0

# Output variables

total\_months\_to\_reach = 0

# Calculation :

while current\_saving <= down\_payment:

    if total\_months\_to\_reach in [6, 12, 18]:

        annual\_salary += annual\_salary \* semi\_annual\_raise

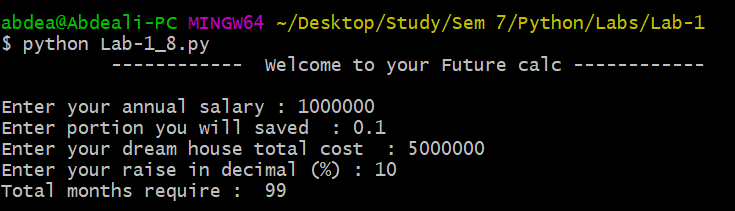
        portion\_saved\_salary = (annual\_salary / 12) \* portion\_saved

    current\_saving = current\_saving + (current\_saving \* returnPortion) + portion\_saved\_salary

    total\_months\_to\_reach = total\_months\_to\_reach + 1

print("Total months require : ", total\_months\_to\_reach)

**Output 7.2 :**



**Q 8)** Check and return the palindromes in a string and list both.

**Code :**

stringInput = input("Enter a string : ")

N = int(input("Enter total length of List : "))

listInput = []

for index in range(N):

    listInput.append(input("Enter string : "))

stringInput = stringInput.replace(" ", "")

if stringInput[::-1] == stringInput:

    print("\n String is Palindrome")

else:

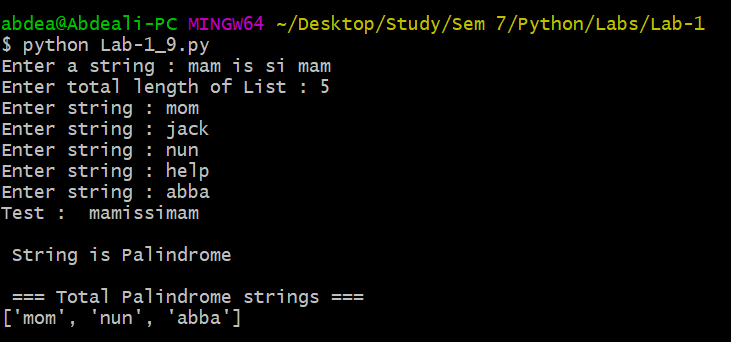
    print("\n String is not palindrome")

print("\n === Total Palindrome strings ===")

listInput = [word for word in listInput if word[::-1] == word]

print(listInput)

**Output 8 :**



**Q 9)** Given a list

L1=[34,78,-12,44,78,91,60,-34,88]

1. Return the cube of all list elements greater than 50.
2. Remove all negative elements from list.
3. Remove the element at index 4 from the list
4. Pop the last element the from the list.

**Code :**

lists = [34, 78, -12, 44, 78, 91, 60, -34, 88]

print("Original list  : \n", lists, "\n")

cube = [x \*\* 3 for x in lists if x > 50]

print(cube)

positives = [x for x in lists if x >= 0]

print(positives)

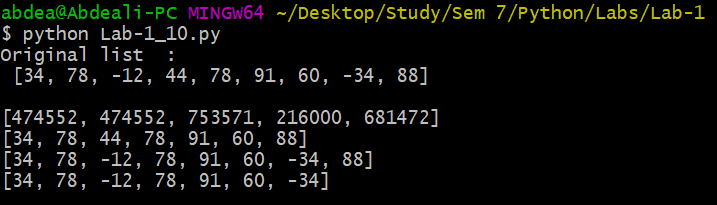
lists.remove(lists[3])

print(lists)

lists.pop()

print(lists)

**Output 9 :**



**Q 10)** Create a list from the given string.

Create a string from the given list.

**Code :**

# Create a list from the given string.

stringInput = input("Enter your Sting : ")

listFromString = stringInput.split(" ")

print("\n Input String : ", stringInput)

print("List of String : ", listFromString)

# Create a string from the given list.

N = int(input("Enter total length of List : "))

listInput = []

for index in range(N):

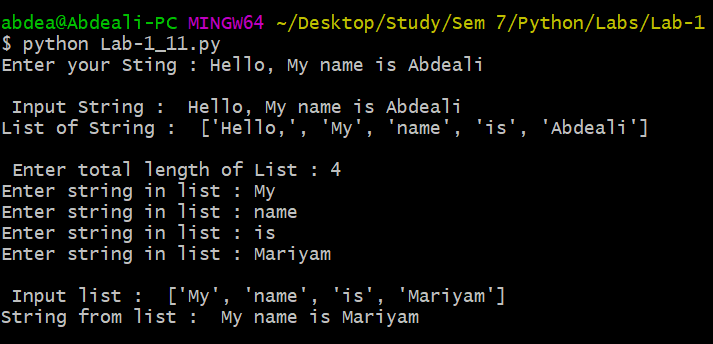
    listInput.append(input("Enter string in list : "))

stringFromList = str(" ".join(listInput))

print("\n Input list : ", listInput)

print("String from list : ", stringFromList)

**Output 10 :**



**ASSIGNMENT-2**

**Q 1)** Write a recursive python function that takes a parameter list and returns the maximum number stored in the list.

**Code :**

N = int(input("\n Enter total length of List : "))

listInput = []

for index in range(N):

    listInput.append(int(input("Enter number in list : ")))

def find\_max(elements, i, maxi):

    if i >= len(elements):

        return maxi

    if elements[i] > maxi:

        maxi = elements[i]

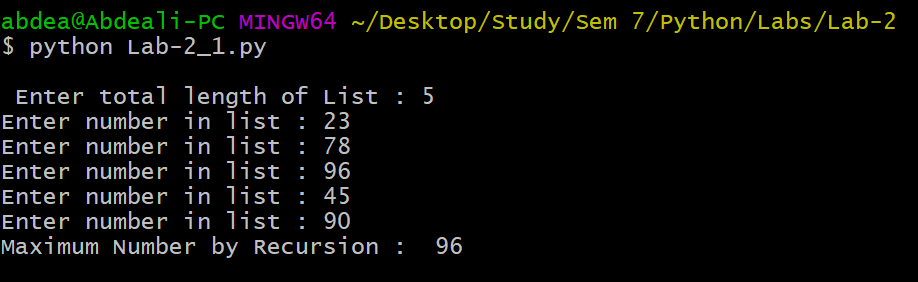
    maxi = find\_max(elements, i + 1, maxi)

    return maxi

maxNumber = find\_max(listInput, 0, listInput[0])

print("Maximum Number by Recursion : ", maxNumber)

**Output 1 :**



**Q 2)**  Remove all the occurrences of a specific item from the list.

**Code :**

N = int(input("\n Enter total length of List : "))

listInput = []

for index in range(N):

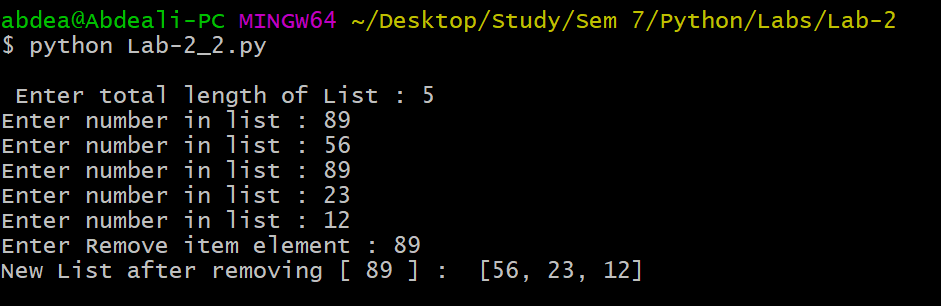
    listInput.append(int(input("Enter number in list : ")))

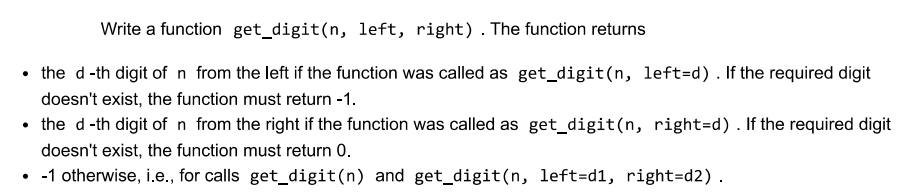
item = int(input("Enter Remove item element : "))

listInput = [x for x in listInput if x != item]

print("New List after removing [", item, "] : ", listInput)

**Output 2:**



**Q 3)** 

**Code :**

digits = input("Enter number : ")

direction = input("Enter direction (left or right) : ")

d = int(input("Enter position : "))

def get\_digit(digitsIn, left=-1, right=-1):

    if (left == -1 and right == -1) or (left != -1 and right != -1):

        return -1

    elif left != -1 and right==-1:

        return digitsIn[left - 1]

    else:

        return digitsIn[right \* -1]

if direction == "left":

    val = get\_digit(digits, left=d)

    print("=> ", d, "th digit from left of ", digits, " is : ", val)

else:

    val = get\_digit(digits, right=d)

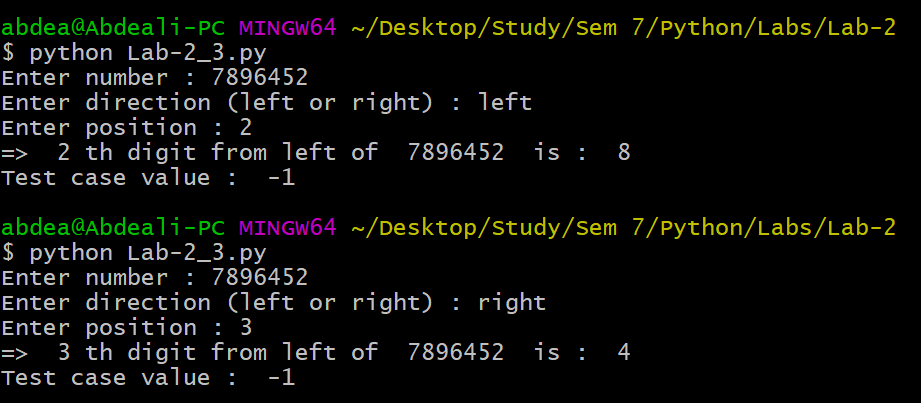
    print("=> ", d, "th digit from left of ", digits, " is : ", val)

# Test case

val = get\_digit(digits, 4, 5)

print("Test case value : ", val)

**Output 3 :**



**Q 4)** Use the gcd function to find the gcd for given numbers 200,444,66, 28,48. Also find the lcm of given numbers using relevant function.

**Code :**

import math

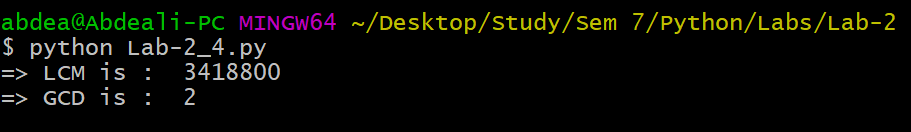
lcmOfNumber = math.lcm(200, 444, 66, 28, 48)

gcdOfNumber = math.gcd(200, 444, 66, 28, 48)

print("=> LCM is : ", lcmOfNumber)

print("=> GCD is : ", gcdOfNumber)

**Output 4 :**





**Q 5)** Find the natural log and log base 10 of the 5 and 75 using relevant function from math module.

**Code :**

import math

naturalLog = math.log(5)

tenLog = math.log10(5)

print("ln(5) = ", naturalLog)

print("log(5) = ", tenLog)

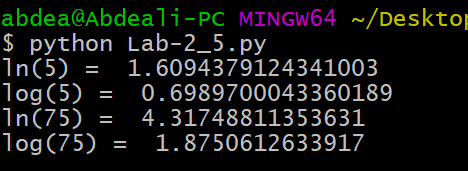
naturalLog = math.log(75)

tenLog = math.log10(75)

print("ln(75) = ", naturalLog)

print("log(75) = ", tenLog)

**Output 5 :**



**Q 6)** Use the OS module and files functions to check if the text file named StudentDetails.txt exists In the parent folder of the assignnment2 folder, if it does not exist create the text file. Add names of 3 students and their birthdates to the file and then close the file. Open the file again and add two more student details to the file. Finally read the details of all the students from the file and display properly on screen. Return the names of all students in a list named snames.

**Code :**

import os

parent\_folder = os.path.abspath(os.path.join(os.getcwd(), os.pardir))

file\_path = os.path.join(parent\_folder, "StudentDetails.txt")

if not os.path.isfile(file\_path):

    with open(file\_path, 'w') as file:

        for i in range(3):

            name = input("Enter student Name : ")

            birthdate = input("Enter birthday (dd-mm-YYYY) : ")

            file.write(f"{name}, {birthdate} \n")

    print("File created: StudentDetails.txt")

else:

    print("File already exists: StudentDetails.txt")

with open(file\_path, 'a') as file:

    for i in range(2):

        name = input("Enter student Name : ")

        birthdate = input("Enter birthday (dd-mm-YYYY) : ")

        file.write(f"{name}, {birthdate} \n")

snames = []

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

    lines = file.readlines()

    for line in lines:

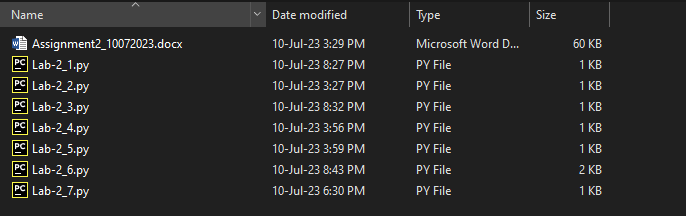
        name, birthdate = line.strip().split(', ')

        snames.append(name)

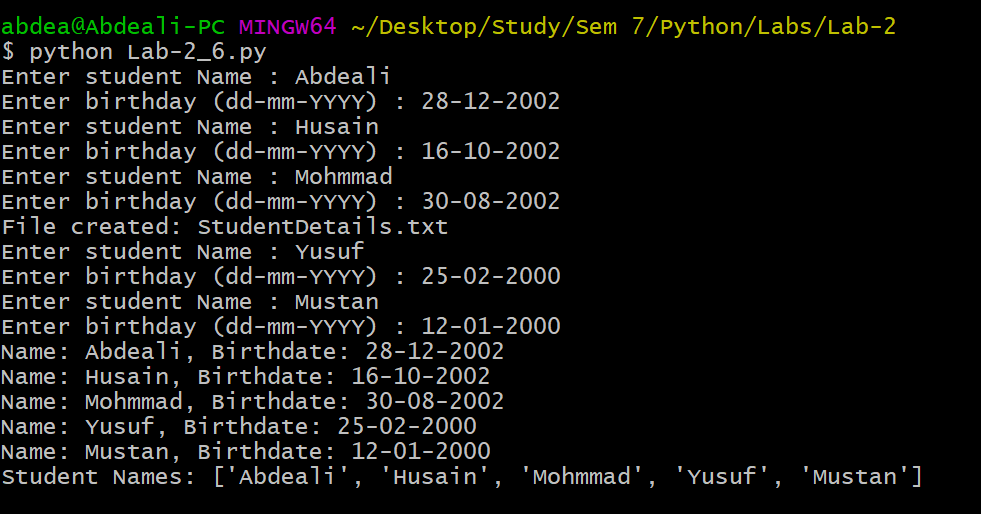
        print(f"Name: {name}, Birthdate: {birthdate}")

print("Student Names:", snames)

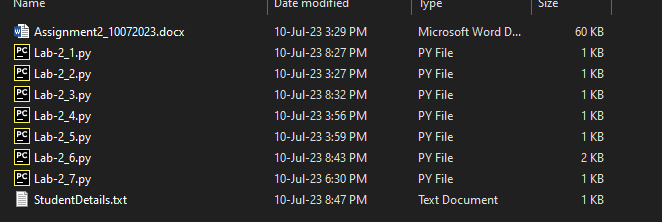
**Output 6.1 : txt file not exist**



**Output 6.2 : Inputs**



**Output 6.3 : txt file exist**



**Q 7)** Create a function rever(n) in python that returns the value of the integer argument n with its digits order reversed and sign unchanged (1719 -> 9171)

**Code :**

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

print("Original value : ", n)

def reverse\_number(ip):

    negative = False

    if ip < 0:

        negative = True

        ip = abs(ip)

    rev = 0

    while ip != 0:

        last\_digit = ip % 10

        rev = rev \* 10 + last\_digit

        ip = int(ip / 10)

    if negative:

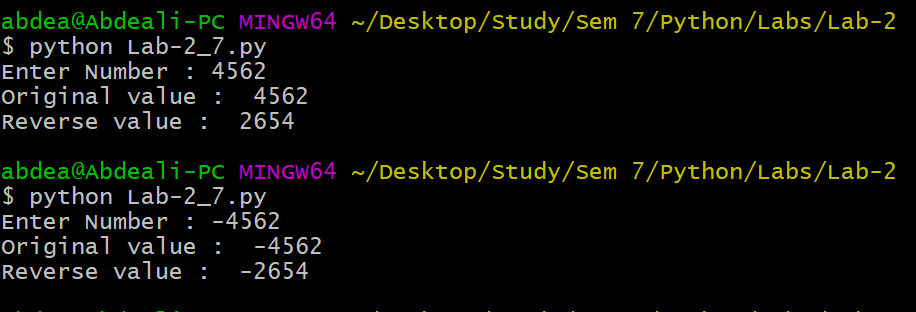
        rev = -rev

    return rev

rev\_op = reverse\_number(n)

print("Reverse value : ", rev\_op)

**Output 7 :**



**ASSIGNMENT-3**

**Q 1)** CSV queries to be solved by csv module and pandas library :

1. Longest Distance Journey in miles?

2. Longest return journey in miles?

3. Longest single journey in miles?

4. Carrier whom we paid the most?

5. Carries whom we flew the most number of times?

6.list of all distinct carriers

7. list of all distinct destinations

8. Mean price of all single route ticket prices?

9. Create new csv file for destination Amsterdam with all columns.

10. Display month of each journey in name

11. Display the weekday of each journey

**Code :**

import csv

from datetime import datetime

# Step 1: Read the CSV file

data = []

with open('Met\_Office\_2011\_Air\_Data.csv', 'r') as csvfile:

    csv\_reader = csv.reader(csvfile)

    headers = next(csv\_reader)  # Read the headers and skip the first row

    for row in csv\_reader:

        data.append(row)

# Step 2: Perform the required queries

# 1. Longest Distance Journey in miles?

longest\_distance\_journey = max(float(row[8]) for row in data)

# 2. Longest return journey in miles?

longest\_return\_journey = max(float(row[8]) for row in data if row[5] == "Return")

# 3. Longest single journey in miles?

longest\_single\_journey = max(float(row[8]) for row in data if row[5] == "Single")

# 4. Carrier whom we paid the most?

carrier\_paid\_most = max(data, key=lambda x: float(x[3]))[11]

# 5. Carriers whom we flew the most number of times?

carriers\_flew\_most = max(((row[11], row[0]) for row in data), key=lambda x: x[1])[0]

# 6. List of all distinct carriers

distinct\_carriers = set(row[11] for row in data)

# 7. List of all distinct destinations

distinct\_destinations = set(row[10] for row in data)

# 8. Mean price of all single route ticket prices?

single\_prices = [float(row[1]) for row in data if row[4] == "Ticket" and row[5] == "Single"]

mean\_single\_price = sum(single\_prices) / len(single\_prices)

# 9. Create a new CSV file for destination Amsterdam with all columns.

amsterdam\_data = [row for row in data if row[10] == "AMSTERDAM"]

with open('amsterdam.csv', 'w', newline='') as csvfile:

    csv\_writer = csv.writer(csvfile)

    csv\_writer.writerow(headers)

    csv\_writer.writerows(amsterdam\_data)

# 10 & 11 . Creating 2D array with column : Journey  Month WeekDay

journey\_combinations = set()

months = {1: 'January', 2: 'February', 3: 'March', 4: 'April', 5: 'May', 6: 'June', 7: 'July', 8: 'August', 9: 'September', 10: 'October', 11: 'November', 12: 'December'}

weekdays = {0: 'Monday', 1: 'Tuesday', 2: 'Wednesday', 3: 'Thursday', 4: 'Friday', 5: 'Saturday', 6: 'Sunday'}

for row in data:

    journey\_date = datetime.strptime(row[7], '%d-%m-%Y')

    journey\_month = months[journey\_date.month]

    journey\_weekday = weekdays[journey\_date.weekday()]

    journey\_combination = (row[9] + ' to ' + row[10], journey\_month, journey\_weekday)

    journey\_combinations.add(journey\_combination)

# Convert the set of tuples to a 2D list

journey\_data = list(journey\_combinations)

# Print or use the results as needed

print("Longest Distance Journey in miles:", longest\_distance\_journey,"\n")

print("Longest return journey in miles:", longest\_return\_journey,"\n")

print("Longest single journey in miles:", longest\_single\_journey,"\n")

print("Carrier whom we paid the most:", carrier\_paid\_most,"\n")

print("Carriers whom we flew the most number of times:", carriers\_flew\_most,"\n")

print("List of all distinct carriers:", distinct\_carriers,"\n")

print("List of all distinct destinations:", distinct\_destinations,"\n")

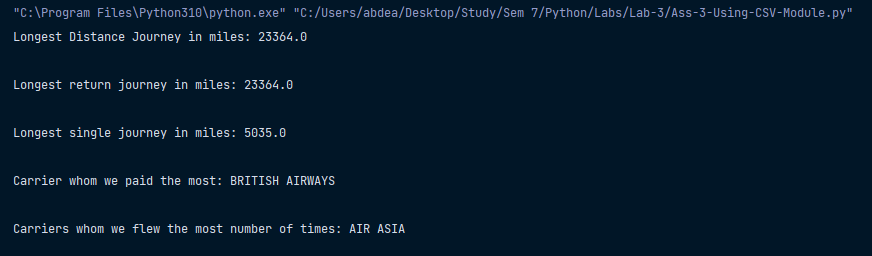
print("Mean price of all single route ticket prices:", mean\_single\_price,"\n")

print("Journey Date : ","\n")

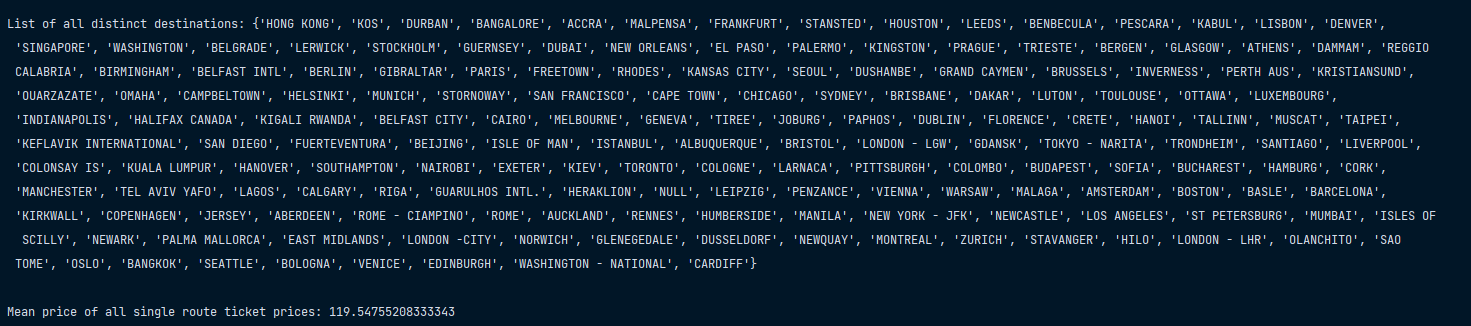
for row in journey\_data:

    print(row)

**Output :**







**Q 2)** Create a json file from given CSV file and use json module to read the json file and read the file into your python program and display the details of first 5 rows on screen.

Read and display all the distinct destinations on the screen.

Write one new row to the json file (I mean detail of one more journey)

What is the total number of miles travelled?

**Code :**

import pandas as pd

import json

# Read the CSV file into a DataFrame

df = pd.read\_csv('Met\_Office\_2011\_Air\_Data.csv')

# Convert DataFrame to JSON and save it to a file

df.to\_json('JsonCreated.json', orient='records', lines=True)

# Read JSON file into a list of dictionaries

with open('JsonCreated.json', 'r') as json\_file:

    data = [json.loads(line) for line in json\_file]

print(data)

# Display the details of the first 5 rows

for row in data[:5]:

    print(row)

distinct\_destinations = set(row['Journey Finish Point'] for row in data)

print("Distinct Destinations:", distinct\_destinations)

# Create a new journey row as a dictionary

new\_journey = {

    "Customer": "Abdeali",

    "Ticket Price ex VAT": 120.50,

    "Number of Travellers": 1,

    "Total Cost ex VAT": 120.50,

    "Ticket Type": "Ticket",

    "Ticket Single or Return": "Return",

    "Travel Class": "ECONOMY",

    "Travel Date": "01-09-2023",

    "Miles Travelled": 500,

    "Journey Start Point": "Dahod",

    "Journey Finish Point": "Mumbai",

    "Air Carrier": "AIR INDIA"

}

# Append the new journey to the data list

data.append(new\_journey)

# Write the updated data list to the JSON file

with open('JsonCreated.json', 'w') as json\_file:

    for row in data:

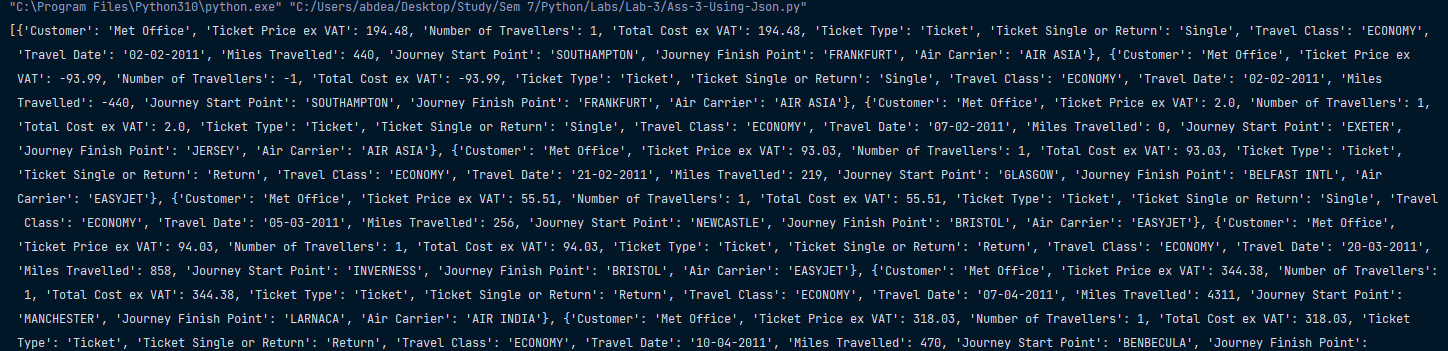
        json.dump(row, json\_file)

        json\_file.write('\n')

total\_miles\_traveled = sum(row['Miles Travelled'] for row in data)

print("Total Number of Miles Traveled:", total\_miles\_traveled)

**Output :**



**ASSIGNMENT-4**

**Q 1)** Create a class Account with 2 instance attribues balance and holdername and 1 class level attribute interest rate.

* Create two methods deposit and withdraw, ensure to catch and throw appropriate exception if a person tries to overdraw from their Account.
* Create a method transferamount to transfer money from 1 Account to another
* Implement --str-- appropriately
* Create getter setter properties for balance attribute using property decorator
* Create a class saving Account that inherits from account class and add some appropriate methods and attributes
* Check if overloading is allowed is allowed in pytjon and comment?
* Check if over riding is allowed is allowed in pytjon and comment?
* Check if static polymorphism and dynamic polymorphism is possible In python and comment?

**Code :**

class InvalidWithdrawAmount(Exception):

    def \_\_init\_\_(self, amount):

        self.amount = amount

    def \_\_str\_\_(self) -> str:

        if self.amount <= 0:

            return "Amount must be greater than zero "

        else:

            return "Invalid amount,check your balance "

class Account:

    # class level attribute

    interest\_rate = 0.05

    def \_\_init\_\_(self, holderName, balance=0):

        # protected

        self.\_balance = balance

        self.holderName = holderName

    def deposit(self, depositMoney):

        try:

            if depositMoney <= 0:

                raise InvalidWithdrawAmount(depositMoney)

            else:

                self.\_balance += depositMoney

                print(f"\n {depositMoney} accepted successfully!!")

                print(f"Total balance : {self.\_balance} ")

        except InvalidWithdrawAmount as i:

            print(i)

    def withdraw(self, withdrawMoney):

        try:

            if withdrawMoney >= self.\_balance or withdrawMoney <= 0:

                raise InvalidWithdrawAmount(withdrawMoney)

            else:

                self.\_balance -= withdrawMoney

                print(f"\n {withdrawMoney} withdraw successfully!!")

                print(f"Total balance : {self.\_balance} ")

        except InvalidWithdrawAmount as i:

            print(i)

    def transferMoney(self, senderAccount, amount):

        try:

            if amount >= self.\_balance or amount <= 0:

                raise InvalidWithdrawAmount(amount)

            else:

                senderAccount.deposit(amount)

                self.withdraw(amount)

                print(f"\n {amount} /- transfer successfully to {senderAccount.holderName}")

                print(f"Total balance : {self.balance} ")

        except InvalidWithdrawAmount as i:

            print(i)

    @property

    def balance(self):

        return self.\_balance

    @balance.setter

    def balance(self, new\_balance):

        if new\_balance < 0:

            raise ValueError("Balance cannot be negative.")

        self.\_balance = new\_balance

    def \_\_str\_\_(self) -> str:

        return f"\n Account holder name '{self.holderName}' and current balance is ' {self.balance} /- '"

ac1 = Account("Abdeali", 500)

ac2 = Account("Husain", 1000)

ac1.deposit(5500)

ac1.withdraw(500)

ac1.transferMoney(ac2, 4000)

# this call @property

print(f"\n Balance of {ac1.holderName} : {ac1.balance}")

print(f"Balance of {ac2.holderName} : {ac2.balance} ")

# Saving Account

class SavingAccount(Account):

    def \_\_init\_\_(self, holderName, balance=0):

        super().\_\_init\_\_(holderName, balance)

    def addInterest(self):

        self.\_balance += self.balance \* Account.interest\_rate

        print(f"\n Interest added , current balance : {self.balance}")

    def \_\_str\_\_(self) -> str:

        return f"Saving Account holder name '{self.holderName}' and current balance is ' {self.balance} /- '"

print("\n ----------------------------------------- \n")

mhd = SavingAccount("Mohammad", 1000)

print(mhd)

mhd.addInterest()

mhd.transferMoney(ac2, 500)

**Theorical Answer :**

"""Method Overloading and Static Polymorphism: While some languages achieve static polymorphism through method

overloading, Python doesn't emphasize this. It doesn't strictly support method overloading based on parameter types.

The latest defined method with the same name will override any previous definitions, making it somewhat different

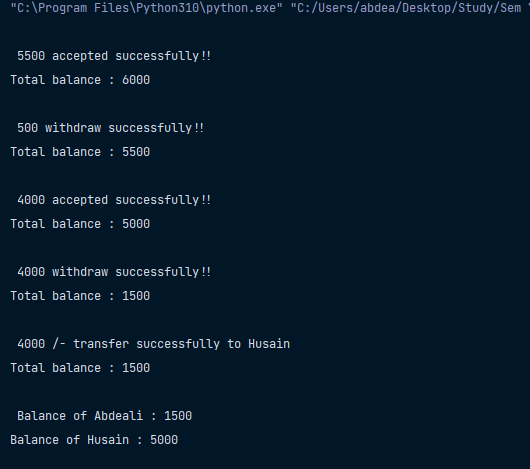
from classic static polymorphism.

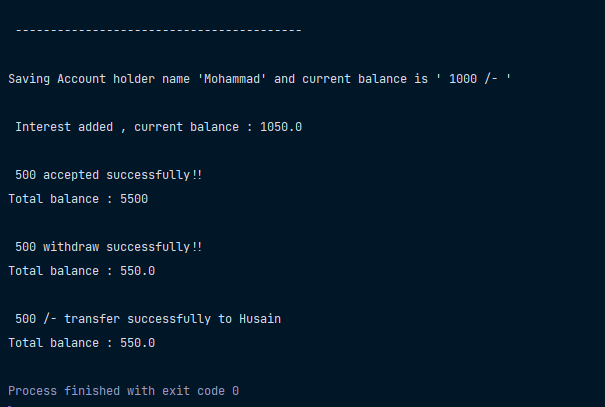
Method Overriding and Dynamic Polymorphism: Yes, in Python, method overriding is a way to achieve dynamic

polymorphism. The specific method to call is determined at runtime based on the object's actual type. """

**Output :**

SEE NEXT PAGE





**Q 2**) Write your own code to test what happens if one class c inherits from 2 classes a and b , and you call a method test on object of c, if test method is defined in both parent classes, which one will be called on object of c.

**Code :**

class A:

    def say\_name(self):

        print("I am in A")

class B:

    def say\_name(self):

        print("I am in B")

class C(A, B):

    pass

c = C()

c.say\_name()

**Theorical Answer :**

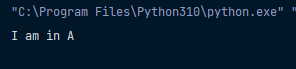
"""

Method Resolution order (MRO) , give priority  left to right when there is a multiple inheritance or same method.

So, Here 'A' is on left hand side so A's method will be called!

"""

**Output :**



**Q 3**) Create a point class with 2 instance attribute x and y, and a class attribute pointcount create instance method distance that find distance of current point to some other Create a classmethod that display value of pointcount variable and also create a staticmethod for the same.

**Code :**

class Point:

pointCount = 2

    def \_\_init\_\_(self, x, y):

        self.x = x

        self.y = y

    def distance\_from(self, other) -> float:

        distance = abs(self.x - other.x) + abs(self.y - other.y)

        return distance

    @classmethod

    def get\_point\_count(cls):

        return cls.pointCount

    @staticmethod

    def get\_distance\_name():

        print("\n Distance finding with Manhattan")

    point1 = Point(5, 7)

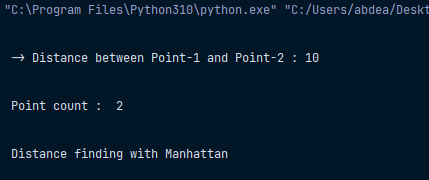
    point2 = Point(12, 4)

    print(f"\n -> Distance between Point-1 and Point-2 : {point1.distance\_from(point2)} ")

    print(f"\n Point count :  {Point.get\_point\_count()}")

    Point.get\_distance\_name()

**Output :**



**Q 4)** Create a 3D-Point class that inherits from point class and add suitable methods and attributes.

**Code :**

from Lab\_4\_3 import Point

class ThreeDPoint(Point):

    def \_\_init\_\_(self, x, y, z=0):

        super(ThreeDPoint, self).\_\_init\_\_(x, y)

        self.z = z

    def distance\_from(self, other) -> float:

        distance = abs(self.x - other.x) + abs(self.y - other.y) + abs(self.z - other.z)

        return distance

    @staticmethod

    def get\_method\_name():

        print("\n Name is Manhattan")

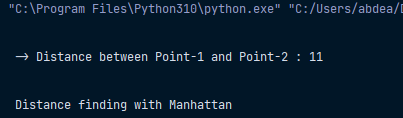
dpoint1 = ThreeDPoint(10, 12, 20)

dpoint2 = ThreeDPoint(9, 7, 15)

print(f"\n -> Distance between Point-1 and Point-2 : {dpoint1.distance\_from(dpoint2)}")

ThreeDPoint.get\_distance\_name()

**Output :**



**ASSIGNMENT-5**

**Q 1)** Write a program which takes 2 digits, X,Y as input and generates a 2-dimensional array.

The element value in the i-th row and j-th column of the array should be i\*j.

Note: i=0,1.., X-1; j=0,1,Y-1.

Suppose the following inputs are given to the program: 3,5

Then, the output of the program should be:

[[0, 0, 0, 0, 0], [0, 1, 2, 3, 4], [0, 2, 4, 6, 8]]

**Code :**

import numpy as np

x = int(input("Enter number of rows(x) : "))

y = int(input("Enter number of columns(x) : "))

def generate\_2D\_array(x, y):

    array\_2d = np.zeros((x, y), dtype=int)

    for i in range(x):

        for j in range(y):

            array\_2d[i][j] = i \* j

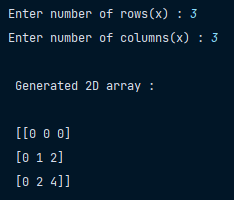
    return array\_2d

two\_D\_array = generate\_2D\_array(x, y)

print("\n Generated 2D array : ")

print("\n", two\_D\_array)

**Output :**



**Q 2)** Write a program that accepts a comma separated sequence of words as input and prints the words in a comma-separated sequence after sorting them alphabetically.

**Code :**

input\_string = str(input("Enter your comma separated string : ")).replace(" ", "").lower()

word\_list = input\_string.split(",")

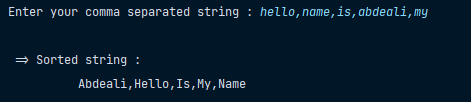
word\_list.sort()

input\_string = ",".join([x.capitalize() for x in word\_list])

print("\n => Sorted string : ")

print("\t\t ", input\_string)

**Output :**



**Q 3)** Use a list comprehension to square each odd number in a list. The list is input by a comma-separated numbers.

**Code :**

input\_string = str(input("Enter your comma separated list : ")).replace(" ", "")

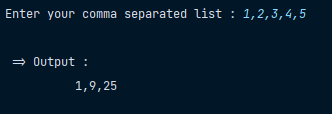
number\_list = input\_string.split(",")

number\_list = ",".join([str(int(x)\*int(x)) for x in number\_list if int(x) % 2 != 0])

print("\n => Output : ")

print("\t\t ", number\_list)

**Output :**



**Q 4)** A website requires the users to input username and password to register. Write a program to check the validity of password

input by users.

Following are the criteria for checking the password:

1. At least 1 letter between [a-z]

2. At least 1 number between [0-9]

1. At least 1 letter between [A-Z]

3. At least 1 character from [$#@]

4. Minimum length of transaction password: 6

5. Maximum length of transaction password: 12

Your program should accept a sequence of comma separated passwords and will check them according to the above criteria.

Passwords that match the criteria are to be printed, each separated by a comma.

Example

If the following passwords are given as input to the program:

ABd1234@1,a F1#,2w3E\*,2We3345

Then, the output of the program should be:

ABd1234@1

**Code :**

import re

input\_string = str(input("Enter your comma separated Passwords combination : "))

password\_list = input\_string.split(",")

def validate\_password(password):

    pattern = re.compile(r"^(?=.\*[$#@])[A-Za-z0-9$#@]{6,12}$")

    if pattern.fullmatch(password):

        return True

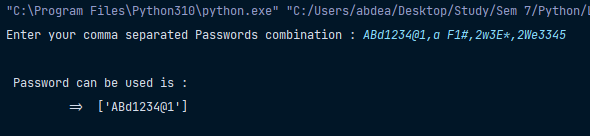
    return False

correct\_password = list(filter(validate\_password, password\_list))

print(f"\n Password can be used {'are' if len(correct\_password) > 1 else 'is'} : ")

print("\t\t => ", correct\_password)

**Output :**



**Q 5)** You are required to write a program to sort the (name, age, height) tuples by ascending order where name is

string, age and height are numbers. The tuples are input by console. The sort criteria is: 1: Sort based on name; 2:

Then sort based on age; 3: Then sort by score. The priority is that name > age > score. If the following tuples are

given as input to the program: Tom,19,80 John,20,90 Jony,17,91 Jony,17,93 Json,21,85 Then, the output of the program.

**Code :**

input\_string = str(input("Enter your comma separated data with white space : "))

studentList = input\_string.split(" ")

studentsData = []

for st in studentList:

    name, age, score = st.split(",")

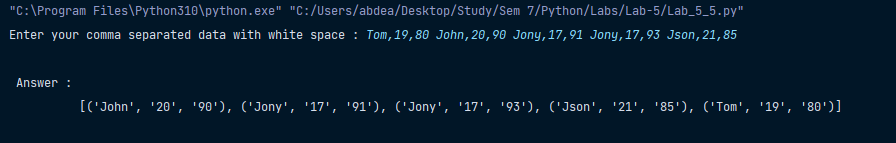
    studentsData.append((name, age, score))

studentsData.sort()

print("\n Answer : ")

print("\t\t ", studentsData)

**Output :**



**Q 6)** Write a binary search function which searches an item in a sorted list. The function should return the index of element to be searched in the list.

**Code :**

data = str(input("Enter your comma separated data with sorted order : ")).replace(" ", "").split(",")

intData = []

for i in data:

    intData.append(int(i))

ip = int(input("Enter number to find index : "))

def binarySearch(toFind, array):

    lower = 0

    upper = len(array) - 1

    middle = int((lower + upper) / 2)

    while lower < upper:

        if array[middle] == toFind:

            return middle

        elif array[middle] > toFind:

            upper = middle - 1

            middle = int((lower + upper) / 2)

        else:

            lower = middle + 1

            middle = int((lower + upper) / 2)

    return -1

index = binarySearch(ip, intData)

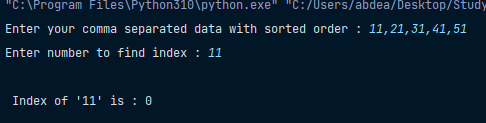
if index == -1:

    print(f"\n Number {ip} is not in list")

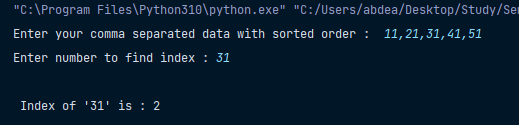
else:

    print(f"\n Index of '{ip}' is : {index}")

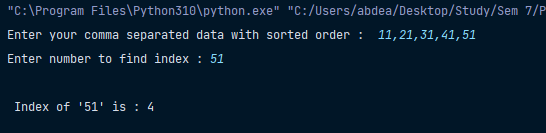
**Output (First element ) :**



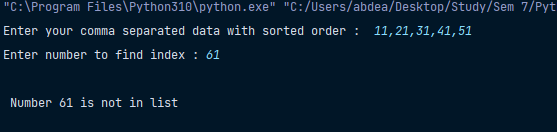
**Output (Middle element ) :**



**Output (Last element ) :**



**Output (Element not available ) :**



**Q 7)** Please write a program which accepts a string from console and print the characters that have even indexes.

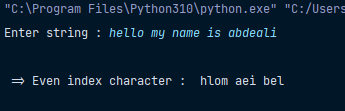
**Code :**

input\_string = str(input("Enter string : "))

even\_index\_char = [x for i, x in enumerate(input\_string) if i % 2 == 0]

print("\n => Even index character : ", "".join(even\_index\_char))

**Output :**



**Q 8)** Given a string S consisting of N, lower case English alphabet, it is also given that a string is encrypted by first replacing every substring of the string consisting of the same character with the concatenation of that character & the hexadecimal representation of the size of the substring and then reversing the whole string, the task is to find the encrypted string.

**Code :**

# getting input

ip\_str = input("Enter your data to be encrypt : ").replace(" ", "").lower()

# getting frequencies

freq\_arr = []

prev\_char = ip\_str[0]

freq = 1

for i in range(1, len(ip\_str)):

    if ip\_str[i] == prev\_char:

        freq += 1

        prev\_char = ip\_str[i]

    else:

        freq\_arr.append({prev\_char: freq})

        freq = 1

        prev\_char = ip\_str[i]

    if i == len(ip\_str) - 1:

        freq\_arr.append({ip\_str[i]: freq})

# 26 hexadecimal for character

hex\_arr = {}

for i in range(97, 123):

    hex\_arr[chr(i)] = format(i, "x")

# iterate freq\_arr and get hexadecimal and combine with frequency

encrypt\_data = ""

for obj in freq\_arr:

    char, freq = list(obj.keys())[0], list(obj.values())[0]

    encrypt\_data += hex\_arr[char] + str(freq)

print("\n => Encrypted Data : ", encrypt\_data)

# Decrypt data :

original\_data = ""

i = 0

while i <= len(encrypt\_data) - 3:

    hex\_val = encrypt\_data[i:i + 2]

    byte\_string = bytes.fromhex(hex\_val)

    char\_val = byte\_string.decode("ASCII")

    freq = int(encrypt\_data[i + 2])

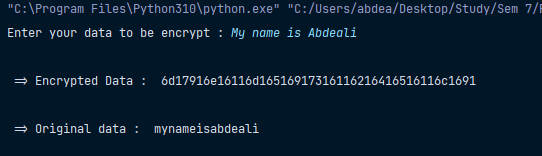
    for s in range(freq):

        original\_data += char\_val

    i += 3

print("\n => Original data : ", original\_data)

**Output :**



**Q 9)** Write a function in Python to check duplicate letters. It must accept a string,The function should return True if the sentence has any word with duplicate letters, else return False.

**Code :**

words = str(input("Enter string : ")).split(" ")

def check\_duplicate\_letter\_in\_word(wordList):

    for w in wordList:

        word\_set = set(w)

        if len(word\_set) != len(w):

            return True

    return False

is\_contain\_duplicate = check\_duplicate\_letter\_in\_word(words)

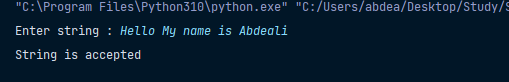
if is\_contain\_duplicate:

    print("String is accepted")

else:

    print("String is not accepted")

**Output :**



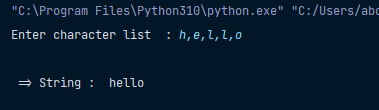
**Q 10)** Write a Python program to convert a list of characters into a string.

**Code :**

op\_string = "".join([x for x in input("Enter character list  : ").replace(" ", "").split(",")])

print("\n => String : ", op\_string)

**Output :**



**Q 11)** Write a Python program to append a list to the second list.

**Code :**

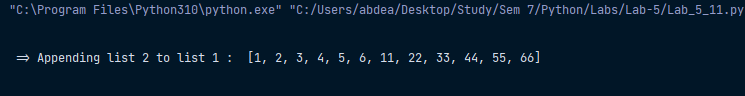
list\_1 = [1, 2, 3, 4, 5, 6]

list\_2 = [11, 22, 33, 44, 55, 66]

list\_1 += list\_2

print("\n => Appending list 2 to list 1 : ", list\_1)

**Output :**



**Q 12)** Write a python program to reverse the given words in a string.

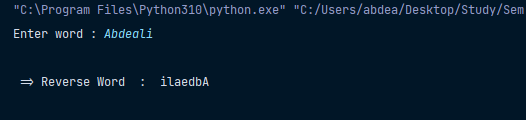
**Code :**

word = input("Enter word : ")

reverse\_word = word[::-1]

print("\n => Reverse Word  : ", reverse\_word)

**Output :**



**Q 13)** Write a program to display the smallest word from a string.

**Code :**

words = input("Enter sentence : ").split(" ")

min\_length = len(words)

min\_word = ""

for w in words:

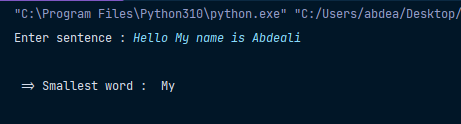
    if len(w) < min\_length:

        min\_length = len(w)

        min\_word = w

print("\n => Smallest word : ", min\_word)

**Output :**



**Q 14)** Write a python program to accept a string and display ascii value of each letter.

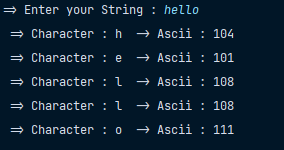
**Code :**

ip\_string = input("=> Enter your String : ")

for x in ip\_string:

    print(f" => Character : {x}  -> Ascii : {ord(x)} ")

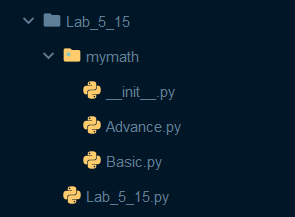
**Output :**





**Q 15)** Create a package in python with at least 2 modules and each module having at least 2 functions.

**Folder Structure :**



**Advance.py : Baisc.py :**

import math

def getGCD(a, b): def addNumber(a, b):

    return math.gcd(a, b) return a + b

def getLCM(a, b): def subNumber(a, b):

    return math.lcm(a, b) return a - b

**Code :**

from mymath import Advance as ad

from mymath import Basic as bs

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

b = int(input("Enter number B : "))

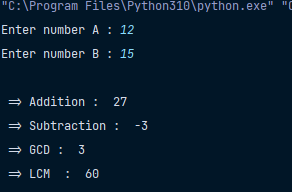
print("\n => Addition : ", bs.addNumber(a, b))

print(" => Subtraction : ", bs.subNumber(a, b))

print(" => GCD : ", ad.getGCD(a, b))

print(" => LCM  : ", ad.getLCM(a, b))

**Output :**



**Q 16)** Implement operator overloading in accounts class using magic methods. + and - operators should be overloaded.

**Code :**

class Account:

    def \_\_init\_\_(self, balance):

        self.balance = balance

    def \_\_add\_\_(self, other):

        return self.balance + other.balance

    def \_\_sub\_\_(self, other):

        return self.balance - other.balance

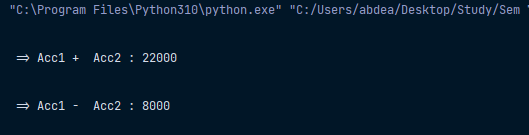
acc1 = Account(15000)

acc2 = Account(7000)

print(f"\n => Acc1 +  Acc2 : {acc1+acc2}")

print(f"\n => Acc1 -  Acc2 : {acc1-acc2}")

**Output :**



**Q 17)** How are abstract classes implemented in python? Implement an abstract class shape in python.

Then inherit concrete classes rectangle and circle from shape class using suitable methods like area

and perimeter.

**Code :**

from abc import ABC, abstractmethod

class Shape(ABC):

    @abstractmethod

    def area(self):

        pass

    @abstractmethod

    def perimeter(self):

        pass

class Rectangle(Shape):

    def \_\_init\_\_(self, x, y):

        self.x = x

        self.y = y

    def area(self):

        return self.x \* self.y

    def perimeter(self):

        return 2 \* (self.x + self.y)

class Square(Shape):

    def \_\_init\_\_(self, x):

        self.x = x

    def area(self):

        return self.x \* self.x

    def perimeter(self):

        return 4 \* self.x

sq = Square(5)

print("\n Area of square : ", sq.area())

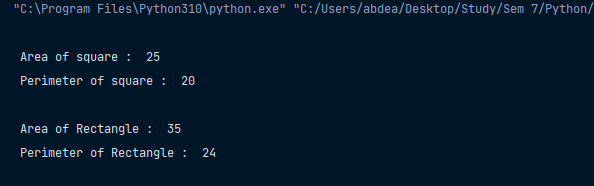
print(" Perimeter of square : ", sq.perimeter())

re = Rectangle(5, 7)

print("\n Area of Rectangle : ", re.area())

print(" Perimeter of Rectangle : ", re.perimeter())

**Output :**





**Q 18)** li1 =[44,66,-75,100,88,43,38,99,-42,101,76,11,15]

Calculate mean and median of the given list using all possible ways you can think of in python and state, best according

to you.

**Code :**

list\_1 = [44, 66, -75, 100, 88, 43, 38, 99, -42, 101, 76, 11, 15]

def getMean(ls1):

    return sum(ls1) / len(ls1)

def getMedian(ls1):

    n = len(ls1)

    ls1.sort()

    if n % 2 == 0:

        m1 = ls1[n // 2]

        m2 = ls1[n // 2 - 1]

        return (m1 + m2) / 2

    return ls1[n // 2]

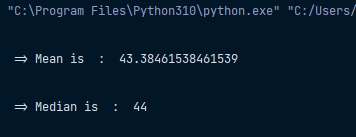
mean = getMean(list\_1)

print("\n => Mean is  : ", mean)

median = getMedian(list\_1)

print("\n => Median is  : ", median)

**Output :**





**Q 19)** You are given an array of objects representing a group of employees, each with a name and a sequence of project

scores. Your task is to use map, filter and reduce (whichever applicable) to calculate the average project scores of

each employee, and then return only those employees who have an average score above 90.

emps= [ { name: 'abc', scores : [59,89,77]}, {name:  'def' , scores:[90,95,80]}...]

**Code :**

from functools import reduce

emps = [

    {

        "name": "Abdeali",

        "scores": [59, 69, 77]

    },

    {

        "name": "Husain",

        "scores": [90, 95, 80]

    },

    {

        "name": "Yusuf",

        "scores": [96, 85, 95]

    },

    {

        "name": "Mustan",

        "scores": [96, 95, 100]

    },

    {

        "name": "Mohammad",

        "scores": [89, 92, 92]

    }

]

# Calculate average score for each employee

def calculate\_average(scores):

    return round(sum(scores) / len(scores), 2)

# Filter employees with average score above 90

def filter\_above\_90(employee):

    return employee["average\_score"] > 90

# Use map to calculate average scores

average\_scores = list(

    map(lambda empl: {"name": empl["name"], "average\_score": calculate\_average(empl["scores"])}, emps))

# Use filter to get employees with average score above 90

high\_scorers = list(filter(filter\_above\_90, average\_scores))

# Print the result

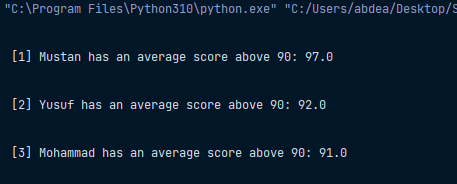
i = 1

for emp in sorted(high\_scorers, key=lambda e: e['average\_score'], reverse=True):

print(f"\n [{i}] {emp['name']} has an average score above 90: {emp['average\_score']}")

i += 1

**Output :**





**Q 20)** Map, filter and reduce are already available in python. But in this exercise you will create mymap,

myfilter and myreduce functions which behave exactly like the built-in functions. and then use the same to solve

problems using your functions and inbuilt functions and check the result obtained.

**User define Map :**

ip\_number = [2, 4, 6, 8, 10]

def convert\_to\_double(num):

    return num \* 2

def myMap(func, ip):

    output = []

    for x in ip:

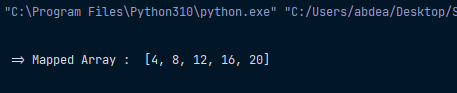
        output.append(func(x))

    return output

arrays = myMap(convert\_to\_double, ip\_number)

print("\n => Mapped Array : ", arrays)

**Output :**



**User define Filter :**

ip\_number = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

def isOdd(num):

    return num % 2 == 1

def myFilter(func, ip):

    output = []

    for x in ip:

        isPositive = func(x)

        if isPositive:

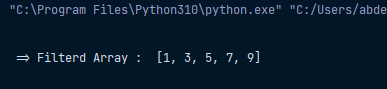
            output.append(x)

    return output

arrays = myFilter(isOdd, ip\_number)

print("\n => Filterd Array : ", arrays)

**Output :**



**ASSIGNMENT-6**

**Q 1) Using TkInter perform CRUD operation.**

from tkinter import \*

from tkinter import messagebox

from tkinter.ttk import Treeview

import sqlite3

# variables

input\_data = []

selected\_router\_id = 0

selected\_row\_number = 0

selected\_tree\_row\_index = ""

class Database:

    # Queries

    fetch\_all\_data\_query = "SELECT \* FROM routers;"

    insert\_query = "INSERT INTO routers VALUES (?,?,?,?,?);"

    update\_query = "UPDATE routers SET hostname=? , brand=? , ram=? , flash=? WHERE id = ?;"

    delete\_query = "DELETE FROM routers WHERE id = ?;"

    hostname\_search\_query = "SELECT \* FROM routers WHERE hostname LIKE ? ;"

    # run only for once

    def \_\_init\_\_(self, database\_name):

        self.database\_name = database\_name

        self.connection = sqlite3.connect(database\_name)

        self.cursor = self.connection.cursor()

        self.cursor.execute(

            """

            CREATE TABLE IF NOT EXISTS

            routers (id INTEGER PRIMARY KEY,

                     hostname TEXT,

                     brand TEXT,

                     ram INTEGER,

                     flash INTEGER)

            """)

        self.connection.commit()

    # Populate data at window loading / Opening

    def fetch\_all\_data(self):

        initial\_data = self.cursor.execute(Database.fetch\_all\_data\_query)

        routers\_data = initial\_data.fetchall()

        if len(routers\_data) != 0:

            for data in routers\_data:

                router\_tree\_view.insert('', 'end', values=data)

    # ---- CRUD Operation Methods (Database)  ---- #

    # 1 : Insert Data

    def insert(self, valid\_input\_data):

        try:

            # save to database

            print(valid\_input\_data)

            self.cursor.execute(Database.insert\_query, valid\_input\_data)

            self.connection.commit()

            # save to tree

            router\_tree\_view.insert('', 'end', values=valid\_input\_data)

        except Exception as e:

            print('Error in saving data : ', e)

            messagebox.showerror('Error', 'Error in saving data,try again...')

    # 2 : Update Data

    def update(self, valid\_input\_data):

        try:

            # update to database

            update\_values = valid\_input\_data[1::]

            update\_values.append(valid\_input\_data[0])

            print(update\_values)

            self.cursor.execute(Database.update\_query, update\_values)

            self.connection.commit()

            # update tree

            router\_tree\_view.item(selected\_tree\_row\_index, values=valid\_input\_data)

        except Exception as e:

            print('Error in updating data : ', e)

            messagebox.showerror('Error', 'Error in updating data,try again...')

    # 3 : Delete Data

    def delete(self, delete\_id):

        try:

            # delete from database

            self.cursor.execute(Database.delete\_query, [delete\_id])

            self.connection.commit()

            # delete temporary

            router\_tree\_view.delete(selected\_tree\_row\_index)

        except Exception as e:

            print('Error in deleting data : ', e)

            messagebox.showerror('Error', 'Error in deleting data,try again...')

    # 4 : search hostname

    def search\_by\_hostname(self, search\_hostname):

        initial\_data = self.cursor.execute(Database.hostname\_search\_query, ('%' + search\_hostname + '%',))

        routers\_data = initial\_data.fetchall()

        if len(routers\_data) != 0:

            for data in routers\_data:

                router\_tree\_view.insert('', 'end', values=data)

    def search\_by\_query(self, search\_query):

        initial\_data = self.cursor.execute(search\_query)

        routers\_data = initial\_data.fetchall()

        if len(routers\_data) != 0:

            for data in routers\_data:

                router\_tree\_view.insert('', 'end', values=data)

# ---- CRUD Operation Functions ---- #

def add\_router():

    isValid = is\_inputs\_valid()

    if isValid:

        database.insert(input\_data)

        clear\_input\_fields()

def update\_router():

    isValid = is\_inputs\_valid()

    if isValid:

        input\_data[0] = selected\_router\_id

        database.update(input\_data)

def remove\_router():

    isValid = is\_inputs\_valid()

    if isValid:

        database.delete(selected\_router\_id)

        clear\_input\_fields()

# ---- Search queries Functions ---- #

def search\_by\_hostname():

    search\_hostname = entry\_search\_by\_hostname.get()

    if search\_hostname == '':

        messagebox.showerror('Error', 'Please provide proper hostname')

        return

    clear\_table\_row()

    database.search\_by\_hostname(search\_hostname)

def execute\_query():

    clear\_table\_row()

    database.search\_by\_query(entry\_search\_by\_query.get())

def search\_by\_queries():

    search\_query = entry\_search\_by\_query.get()

    if search\_query == '':

        messagebox.showerror('Error', 'Please provide proper Query')

        return

    clear\_table\_row()

    database.search\_by\_query(search\_query)

# ---- Helper Functions | Functionality ---- #

def is\_inputs\_valid() -> bool:

    global input\_data

    try:

        new\_id = int(entry\_id.get())

        input\_data = [new\_id, entry\_hostname.get(), entry\_brand.get(), entry\_ram.get(),

                      entry\_flash.get()]

        if '' in input\_data:

            messagebox.showerror('Error', 'Please provide proper inputs')

            return False

    except Exception as e:

        print("Value error in selection : ", e)

        messagebox.showerror('Error', 'Please provide proper inputs')

        return False

    return True

def clear\_input\_fields():

    entry\_id.delete(0, END)

    entry\_hostname.delete(0, END)

    entry\_brand.delete(0, END)

    entry\_ram.delete(0, END)

    entry\_flash.delete(0, END)

def select\_row\_of\_router():

    global selected\_router\_id

    global selected\_tree\_row\_index

    global selected\_row\_number

    try:

        selected\_tree\_row\_index = router\_tree\_view.selection()[0]

        selected\_row\_number = int(selected\_tree\_row\_index[1:])

        selected\_item = router\_tree\_view.item(selected\_tree\_row\_index)['values']

        selected\_router\_id, selected\_hostname, selected\_brand, selected\_ram, selected\_flash = selected\_item

        entry\_id.delete(0, END)

        entry\_id.insert(0, selected\_router\_id)

        entry\_hostname.delete(0, END)

        entry\_hostname.insert(0, selected\_hostname)

        entry\_brand.delete(0, END)

        entry\_brand.insert(0, selected\_brand)

        entry\_ram.delete(0, END)

        entry\_ram.insert(0, selected\_ram)

        entry\_flash.delete(0, END)

        entry\_flash.insert(0, selected\_flash)

        return

    except Exception as e:

        print("Error in selection row  : ", e)

def clear\_queries():

    entry\_search\_by\_query.insert(0, "SELECT \* FROM routers WHERE ")

    entry\_search\_by\_hostname.delete(0, END)

    clear\_table\_row()

    database.fetch\_all\_data()

def clear\_table\_row():

    for item in router\_tree\_view.get\_children():

        router\_tree\_view.delete(item)

# -------------------- Initialize DataBase ------------------- #

database = Database('routers.db')

# -------------------- GUI Application ------------------- #

main\_window = Tk()

main\_window.title('Router Manager')

# ----- Frame 1 :  Search Frame ----- #

frame\_search = Frame(main\_window)

frame\_search.grid(row=0, column=0)

# --> Search by Host Name

lbl\_search\_by\_hostname = Label(frame\_search, text='Search by Host-Name', font=('Fira Code', 12, 'bold'), pady=20)

lbl\_search\_by\_hostname.grid(row=0, column=0, sticky=W, padx=10)

entry\_search\_by\_hostname = Entry(frame\_search, width=40)

entry\_search\_by\_hostname.grid(row=0, column=1, padx=10)

btn\_search\_by\_hostname = Button(frame\_search, text='Search', width=12, command=search\_by\_hostname)

btn\_search\_by\_hostname.grid(row=0, column=2)

clear\_search\_query = Button(frame\_search, text='Clear', width=12, command=clear\_queries)

clear\_search\_query.grid(row=0, column=3, padx=15)

# --> Search by Host Name

lbl\_search\_by\_query = Label(frame\_search, text='Search by Query', font=('Fira Code', 12, 'bold'), pady=20)

lbl\_search\_by\_query.grid(row=1, column=0, sticky=W, padx=10)

entry\_search\_by\_query = Entry(frame\_search, width=40)

entry\_search\_by\_query.insert(0, "SELECT \* FROM routers WHERE ")

entry\_search\_by\_query.grid(row=1, column=1, padx=10)

btn\_search\_by\_query = Button(frame\_search, text='Fire Query', width=12, command=execute\_query)

btn\_search\_by\_query.grid(row=1, column=2)

clear\_search\_query = Button(frame\_search, text='Clear', width=12, command=clear\_queries)

clear\_search\_query.grid(row=1, column=3, padx=15)

# ----- Frame 2 :  Field's Frame ----- #

frame\_fields = Frame(main\_window)

frame\_fields.grid(row=1, column=0)

# -> Id

label\_id = Label(frame\_fields, text='Id', font=('Fira Code', 12, 'bold'))

label\_id.grid(row=0, column=0, sticky=E)

entry\_id = Entry(frame\_fields, width=50)

entry\_id.grid(row=0, column=1, sticky=W, padx=15, pady=15, columnspan=5)

# -> Host  - Name

label\_hostname = Label(frame\_fields, text='Host-Name', font=('Fira Code', 12, 'bold'))

label\_hostname.grid(row=1, column=0, sticky=E)

entry\_hostname = Entry(frame\_fields)

entry\_hostname.grid(row=1, column=1, sticky=W, padx=15)

# -> BRAND

label\_brand = Label(frame\_fields, text='Brand', font=('Fira Code', 12, 'bold'))

label\_brand.grid(row=1, column=2, sticky=E)

entry\_brand = Entry(frame\_fields)

entry\_brand.grid(row=1, column=3, sticky=W, padx=15)

# -> RAM

label\_ram = Label(frame\_fields, text='RAM', font=('Fira Code', 12, 'bold'))

label\_ram.grid(row=2, column=0, sticky=E)

entry\_ram = Entry(frame\_fields)

entry\_ram.grid(row=2, column=1, sticky=W, padx=15)

# -> FLASH

label\_flash = Label(frame\_fields, text='Flash', font=('Fira Code', 12, 'bold'), pady=20)

label\_flash.grid(row=2, column=2, sticky=E)

entry\_flash = Entry(frame\_fields)

entry\_flash.grid(row=2, column=3, sticky=W, padx=15)

# ----- Frame 3 :  Available router frame  ----- #

frame\_crud\_btns = Frame(main\_window)

frame\_crud\_btns.grid(row=3, column=0)

add\_btn = Button(frame\_crud\_btns, text='Add Router', width=12, command=add\_router)

add\_btn.grid(row=0, column=0, padx=15)

remove\_btn = Button(frame\_crud\_btns, text='Remove Router', width=12, command=remove\_router)

remove\_btn.grid(row=0, column=1, padx=15)

update\_btn = Button(frame\_crud\_btns, text='Update Router', width=12, command=update\_router)

update\_btn.grid(row=0, column=2, padx=15)

clear\_btn = Button(frame\_crud\_btns, text='Clear Input', width=12, command=clear\_input\_fields)

clear\_btn.grid(row=0, column=3, padx=15)

# ----- Frame 4 :  Available router frame  ----- #

frame\_available\_router = Frame(main\_window)

frame\_available\_router.grid(row=4, column=0, columnspan=4, rowspan=6, pady=20, padx=20)

# -> Creating tree view (Table)

columns = ['id', 'Hostname', 'Brand', 'Ram', 'Flash']

router\_tree\_view = Treeview(frame\_available\_router, columns=columns, show="headings")

for col in columns:

    router\_tree\_view.column(col, width=120, anchor=CENTER)

    router\_tree\_view.heading(col, text=col)

router\_tree\_view.bind('<<TreeviewSelect>>', select\_row\_of\_router)

router\_tree\_view.pack(side="left", fill="y")

scrollbar = Scrollbar(frame\_available\_router, orient='vertical')

scrollbar.configure(command=router\_tree\_view.yview)

scrollbar.pack(side="right", fill="y")

router\_tree\_view.config(yscrollcommand=scrollbar.set)

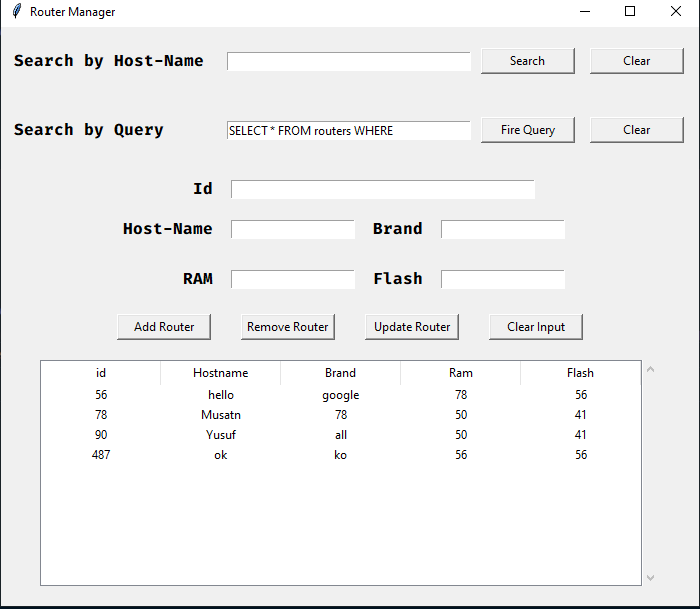
# load already saved data in table

database.fetch\_all\_data()

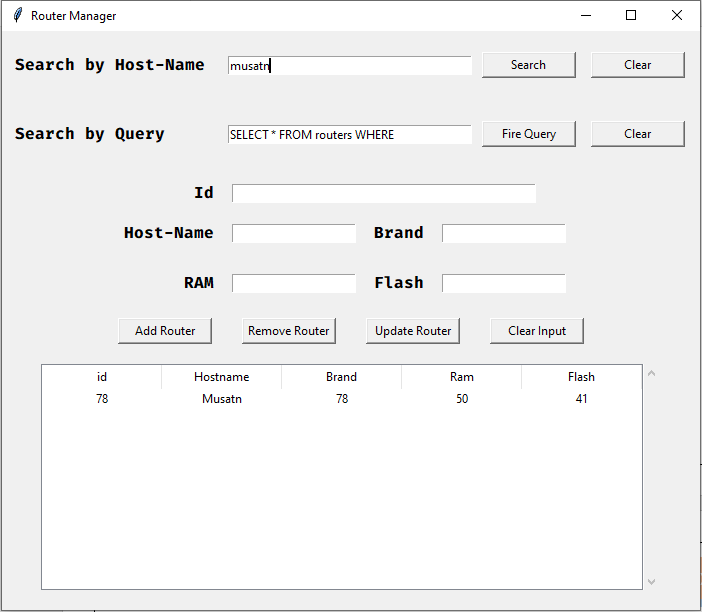
# Start program

main\_window.mainloop()

**Output (Main Screen) :**



**Output (Search by hostname) :**



**ASSIGNMENT-7**

**Q 1)** Write a Python program to check that a string contains only a certain set of characters (in this case a-z,

A-Z and 0-9).2) Write a Python program to check that a string contains only a certain set of characters (in this case

a-z, A-Z and 0-9).

**Code :**

import re

pattern\_string = r'[a-zA-z0-9]+'

pattern = re.compile(pattern\_string)

ip\_string = input("Enter your string : ")

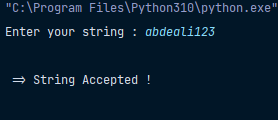
if pattern.fullmatch(ip\_string):

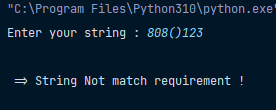
    print("\n => String Accepted !")

else:

    print("\n => String Not match requirement !")

**Output :**





**Q 2)** Write a Python program that matches a string that has an 'a' followed by zero or more b's

**Code :**

import re

input\_string = input("Enter your string : ")

pattern\_string = r'(ab\*)'

pattern = re.compile(pattern\_string, re.IGNORECASE)

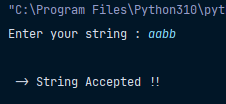
if pattern.search(input\_string):

    print("\n -> String Accepted !!")

else:

    print("\n -> String not Accepted")

**Output :**





**Q 3)** Write a Python program that matches a string that has an 'a' followed by one or more b's

**Code :**

import re

input\_string = input("Enter your string : ")

pattern\_string = r'(ab+)'

pattern = re.compile(pattern\_string, re.IGNORECASE)

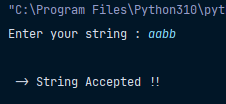
if pattern.search(input\_string):

    print("\n -> String Accepted !!")

else:

    print("\n -> String not Accepted")

**Output :**





**Q 4)** Write a Python program that matches a string that has an 'a' followed by two to three 'b'.

**Code :**

import re

input\_string = input("Enter your string : ")

pattern\_string = r'^ab{2,3}$'

pattern = re.compile(pattern\_string, re.IGNORECASE)

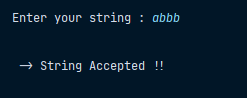
if pattern.search(input\_string):

    print("\n -> String Accepted !!")

else:

    print("\n -> String not Accepted")

**Output :**



**Q 5)** Write a Python program that matches a string that has an 'a' followed by anything, ending in 'b'

**Code :**

import re

input\_string = input("Enter your string : ")

pattern\_string = r'a[a-z]\*b$'

pattern = re.compile(pattern\_string, re.IGNORECASE)

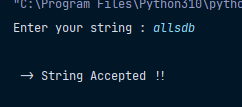
if pattern.search(input\_string):

    print("\n -> String Accepted !!")

else:

    print("\n -> String not Accepted")

**Output :**





**Q 6)** Write a Python program to match a string that contains only upper and lowercase letters, numbers, and underscores.

**Code :**

import re

input\_string = input("Enter your string : ")

pattern\_string = r'^[a-zA-Z\_][a-zA-Z0-9\_]\*[a-zA-Z0-9]$'

pattern = re.compile(pattern\_string)

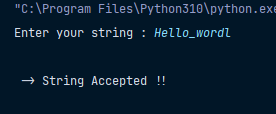
if pattern.search(input\_string):

    print("\n -> String Accepted !!")

else:

    print("\n -> String not Accepted")

**Output :**





**Q 7)** Write a Python program to check for a number at the end of a string.

**Code :**

import re

input\_string = input("Enter your string : ")

pattern\_string = r'[0-9]$'

pattern = re.compile(pattern\_string)

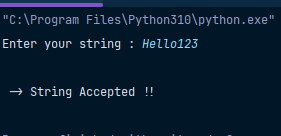
if pattern.search(input\_string):

    print("\n -> String Accepted !!")

else:

    print("\n -> String not Accepted")

**Output :**



**Q 8)** Write a Python program to find the occurrence and position of the substrings within a string.

**Code :**

import re

def find\_substring\_occurrences(main\_string, substring):

    occurrences = [(match.start(), match.end()) for match in re.finditer(substring, main\_string)]

    return occurrences

def main():

    main\_string = input("Enter the main string: ")

    substring = input("Enter the substring to find: ")

    occurrences = find\_substring\_occurrences(main\_string, substring)

    if occurrences:

        print(f"The substring '{substring}' was found at positions:")

        for start, end in occurrences:

            print(f"Start: {start}, End: {end}")

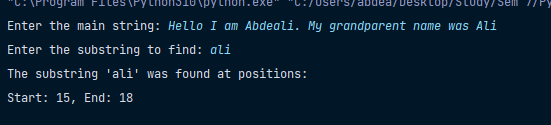
    else:

        print(f"The substring '{substring}' was not found in the main string.")

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

    main()

**Output :**



**ASSIGNMENT-8**

**Q 1)** Find the square of natural numbers 1 to 1000 using list comprehension and using for loop. Use timeit and cprofile to check, which one is more efficient method.

**Code :**

import timeit

import cProfile

# Function to find squares using list comprehension

def squares\_list\_comprehension():

    return [x \*\* 2 for x in range(1, 1001)]

# Function to find squares using a for loop

def squares\_for\_loop():

    squares = []

    for x in range(1, 1001):

        squares.append(x \*\* 2)

    return squares

# Measure time taken using timeit

list\_comp\_time = timeit.timeit(squares\_list\_comprehension, number=1000)

for\_loop\_time = timeit.timeit(squares\_for\_loop, number=1000)

print("Time taken using list comprehension:", list\_comp\_time)

print("Time taken using for loop:", for\_loop\_time)

print("\n ------------------ Profile ------------------ \n")

# Profile the functions using cProfile

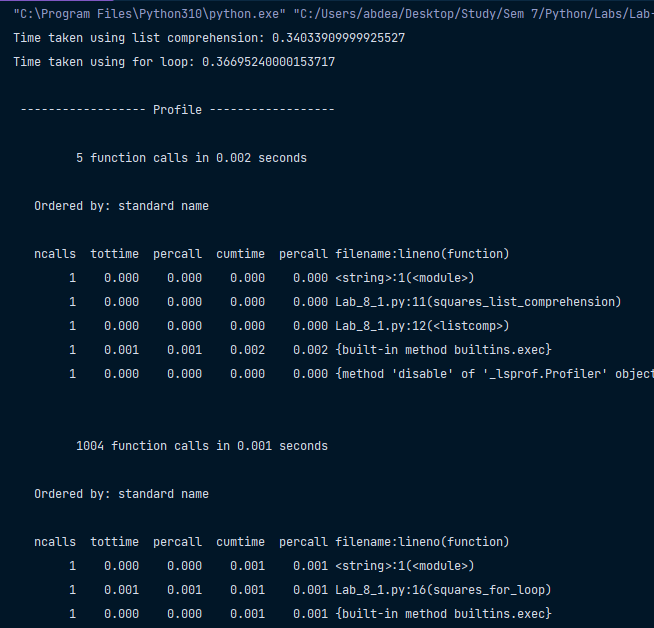
cProfile.run("squares\_list\_comprehension()")

cProfile.run("squares\_for\_loop()")

**Which is More Efficient :**

* results may vary depending on user's system's configuration and Python version.
* But most of the time List comprehension is more Efficient

**Output :**



**Q 2)** Write a python program to concatenate ‘ae’ to name of 15 fruits or trees. Use inbuilt join method in one program and your own method (nothing inbuilt) to do the same in another program. Check which one is more efficient and specify.

**Code :**

import timeit

# List of 15 fruits or trees

fruits\_or\_trees = ["apple", "banana", "cherry", "date", "elderberry", "fig", "grape", "hickory", "juniper", "kiwi"]

# Using the inbuilt join method

def concatenate\_with\_join():

    return ''.join([fruit + 'ae' for fruit in fruits\_or\_trees])

# Using a custom method

def concatenate\_custom():

    result = ''

    for fruit in fruits\_or\_trees:

        result += fruit + 'ae'

    return result

# Measure time taken using timeit

join\_time = timeit.timeit(concatenate\_with\_join, number=1000)

custom\_time = timeit.timeit(concatenate\_custom, number=1000)

print("\n ---------- Time ----------")

print("Time taken using join method:", join\_time)

print("Time taken using custom method:", custom\_time)

print("\n ---------- Result ----------")

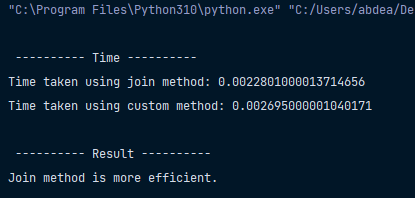
if join\_time < custom\_time:

    print("Join method is more efficient.")

else:

    print("Custom method is more efficient.")

**Output :**



**Q 3.1)**  Given a list,

L1=[“Test”,”Find”,”Try”,”Search”,”Think”,”Innovate”]

Output = [‘e’,’k’,’h’,’y’,’d’,’t’]

Reverse the list and take only last character of each string in the list.

**Code :**

import timeit

def old\_method():

    inputList = ["Test", "Find", "Try", "Search", "Think", "Innovate"]

    op = []

    for x in reversed(inputList):

        op.append(x[len(x) - 1])

    return op

def efficient\_method():

    inputList = ["Test", "Find", "Try", "Search", "Think", "Innovate"]

    Op = [x[-1] for x in reversed(inputList)]

    return Op

old\_method\_time = timeit.timeit(old\_method, number=1000)

efficient\_method\_time = timeit.timeit(efficient\_method, number=1000)

print("\n ---------- Time ----------")

print("Time taken using old method : ", old\_method\_time)

print("Time taken using efficient method : ", efficient\_method\_time)

print("\n ---------- Result ----------")

# Compare efficiency

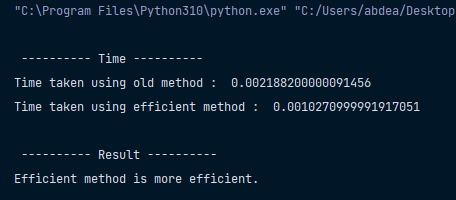
if efficient\_method\_time < old\_method\_time:

    print("Efficient method is more efficient.")

else:

    print("Old method is more efficient.")

**Output :**





**Q 3.2)** Check and return the palindromes in list.

**Code :**

import timeit

words = ["demigod", "rewire", "madam", "freer", "anutforajaroftuna", "kiosk"]

def old\_method():

    return [word for word in words if word[::-1] == word]

def efficient\_method():

    return list(filter(lambda word: word == word[::-1], words))

old\_method\_time = timeit.timeit(old\_method, number=1000)

efficient\_method\_time = timeit.timeit(efficient\_method, number=1000)

print("\n ---------- Time ----------")

print("Time taken using old method : ", old\_method\_time)

print("Time taken using efficient method : ", efficient\_method\_time)

print("\n ---------- Result ----------")

# Compare efficiency

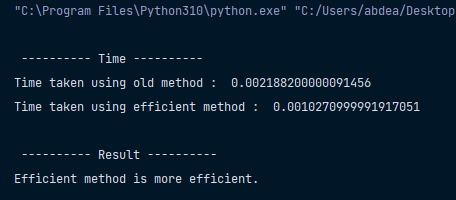
if efficient\_method\_time < old\_method\_time:

    print("Efficient method is more efficient.")

else:

    print("Old method is more efficient.")

**Output :**





**Q 3.3)**  Write a program to display the smallest word from a string.

**Code :**

from functools import reduce

import timeit

words = "Hello name is Abdeali what is your? I am very Happy"

def old\_method():

    min\_length = len(words)

    min\_word = ""

    for w in words.split(" "):

        if len(w) <= min\_length:

            min\_length = len(w)

            min\_word = w

    return min\_word

def efficient\_method():

    min\_word = str(reduce(lambda w1, w2: w1 if len(w1) < len(w2) else w2, words.split(" ")))

    return min\_word

old\_method\_time = timeit.timeit(old\_method, number=1000)

efficient\_method\_time = timeit.timeit(efficient\_method, number=1000)

print("\n ---------- Time ----------")

print("Time taken using old method : ", old\_method\_time)

print("Time taken using efficient method : ", efficient\_method\_time)

print("\n ---------- Result ----------")

# Compare efficiency

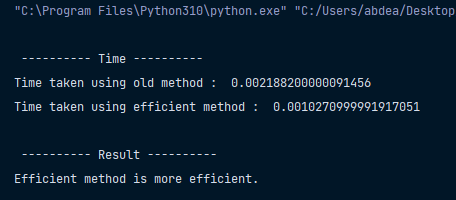
if efficient\_method\_time < old\_method\_time:

    print("Efficient method is more efficient.")

else:

    print("Old method is more efficient.")

**Output :**





**Q 4)** 'Sieve of Eratosthenes' : Algorithm for finding prime numbers (List version)

**Code :**

# Step 1 : Initialization

total\_numbers = [True] \* 5000  # find prime in 2 to 4999

prime\_index = 2

# Step 2 : Generate Primes

while 2 <= prime\_index < 5000:

    # Step 2.1 : Find next prime (Search for true and when it gets stop increasing because it is prime\_index)

    while not total\_numbers[prime\_index] and 2 <= prime\_index < 5000:

        prime\_index += 1

    # Step 2.2 : Remove multiples (numbers which can be divided by prime\_index)

    k = prime\_index + prime\_index

    while k < 5000:

        total\_numbers[k] = False

        k += prime\_index

    # Step 2.3 : Increase Prime Index

    prime\_index += 1

# step 3 : Print all prime number

print("\n\t ---- Prime numbers between 2 to 4999 ---- \n")

prime\_index = 2

while 2 <= prime\_index < 5000:

    # print(total\_numbers[prime\_index])

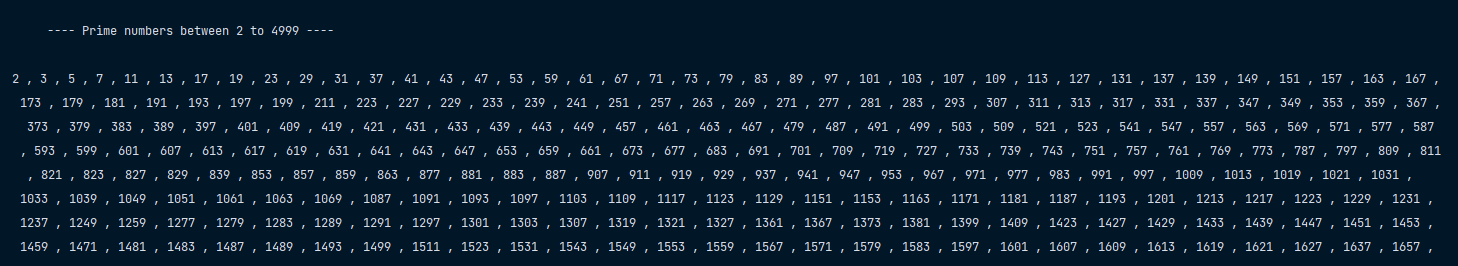
    if total\_numbers[prime\_index]:

        print(f"{prime\_index} ", end=", ")

    prime\_index += 1

print("\n")

**Output :**





**Q 5)**  Define a generator, genrange, which generates the same sequence of values as range, without creating a list object.

**Code :**

def genrange(start, stop, step=1):

    current = start

    while current < stop if step > 0 else current > stop:

        yield current

        current += step

# Example usage (for positive) :

gen = genrange(0, 10, 2)

for i in gen:

    print(i)

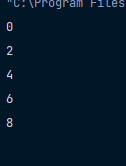
print("\n ----- \n")

# Example usage (for negative) :

for i in genrange(10, 0, -1):

    print(i)

**Output :**



**Q 6)** Prime factors of the number

**Code :**

def factor(n):

    # Handle the case where n is 1 or less

    if n <= 1:

        return

    # Factor out 2 until it's no longer divisible

    while n % 2 == 0:

        yield 2

        n //= 2

# Iterate through odd factors from 3 to sqrt(n) by 2 step cause : 3,5,7,9...

    for f in range(3, int(math.sqrt(n)) + 1, 2):

        while n % f == 0:

            yield f

            n //= f

    # If n is still greater than 1, it's also a prime factor

    if n > 1:

        yield n

while True:

    number\_to\_factor = int(input("Enter number : "))

    if number\_to\_factor <= 1:

        print("Number can't be less than one , re-enter number")

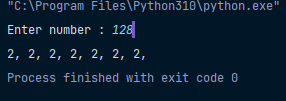
    else:

        break

for prime\_factor in factor(number\_to\_factor):

    print(prime\_factor, end=", ")

**Output :**



**ASSIGNMENT-9**

**Q 1)** Make zip Archiver.

**Code :**

import tkinter as tk

from tkinter import filedialog

import zipfile

import os

def zip\_folders():

    """

    This method is for zip selected folder

    :return: None

    """

    folder\_path = filedialog.askdirectory(title="Select a folder to zip")

    if folder\_path:

        try:

            zip\_file\_path = filedialog.asksaveasfilename(title="Save Zip File As", defaultextension=".zip")

            if zip\_file\_path:

                with zipfile.ZipFile(zip\_file\_path, "w", zipfile.ZIP\_DEFLATED) as zf:

                    for root\_window, dirs, files in os.walk(folder\_path):

                        for file in files:

                            file\_path = os.path.join(root\_window, file)

                            arcname = os.path.relpath(file\_path, folder\_path)

                            zf.write(file\_path, arcname=arcname)

                status\_label.config(text="Folder zipped successfully", fg="green", font=("Fira Code", 14, "bold"))

            else:

                status\_label.config(text="Zip file not selected", fg="red", font=("Fira Code", 14, "bold"))

        except Exception as e:

            status\_label.config(text=f"Error: {str(e)}", fg="red", font=("Fira Code", 14, "bold"))

    else:

        status\_label.config(text="No folder selected", fg="red", font=("Fira Code", 14, "bold"))

def unzip\_file():

    """

    This is method for unzip file at user destination folder

    :return: None

    """

    zip\_file\_path = filedialog.askopenfilename(title="Select a zip file to unzip", filetypes=[("Zip Files", "\*.zip")])

    if zip\_file\_path:

        extract\_path = filedialog.askdirectory(title="Select a location to extract")

        if extract\_path:

            try:

                with zipfile.ZipFile(zip\_file\_path, "r") as zf:

                    zf.extractall(extract\_path)

                status\_label.config(text="File unzipped successfully", fg="green", font=("Fira Code", 14, "bold"))

            except Exception as e:

                status\_label.config(text=f"Error: {str(e)}", fg="red", font=("Fira Code", 14, "bold"))

        else:

            status\_label.config(text="Extraction location not selected", fg="red", font=("Fira Code", 14, "bold"))

    else:

        status\_label.config(text="No zip file selected", fg="red", font=("Fira Code", 14, "bold"))

# ----------------- GUI Application -------------------- #

# The main window

main\_window = tk.Tk()

main\_window.title("Zip and Unzip Application")

main\_window.geometry("450x150")

main\_window.eval('tk::PlaceWindow . center')

#  buttons

zip\_button = tk.Button(main\_window, text="Zip Folder", command=zip\_folders)

unzip\_button = tk.Button(main\_window, text="Unzip File", command=unzip\_file)

# status label

status\_label = tk.Label(main\_window, text="Click on button to perform Task", fg="black", font=("Fira Code", 14, "bold"))

# Place widgets on the window

zip\_button.pack(pady=10)

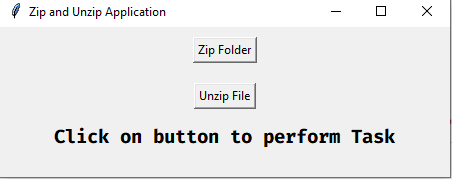
unzip\_button.pack(pady=10)

status\_label.pack()

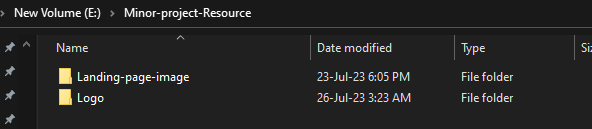
# important (to keep window on)

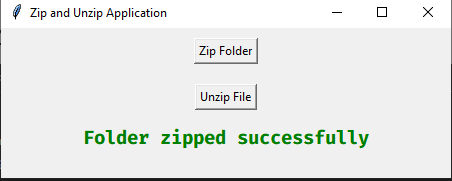
main\_window.mainloop()

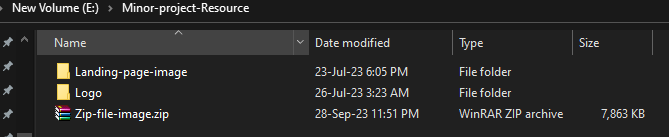
**Output (GUI) :**



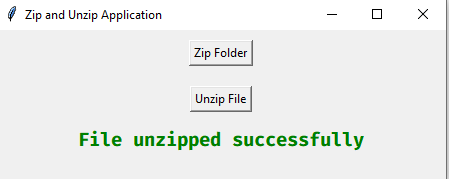
**Output (Zipping folder) :**

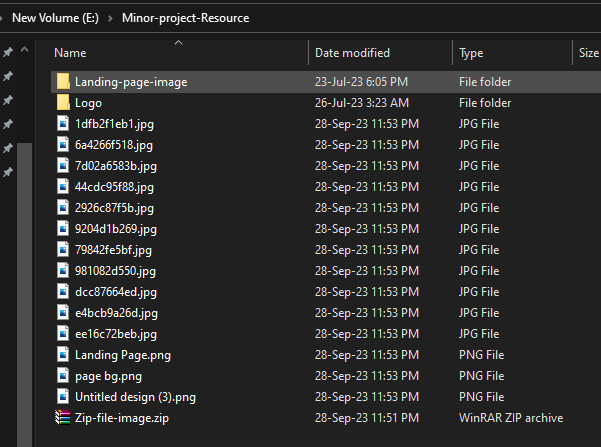






**Output (UnZipping file) :**





**ASSIGNMENT-10**

**Q 1)** Sort tuples by second item in each tuple .

**Code :**

first\_tuple = (('a', 53), ('b', 37), ('c', 23), ('d', 1), ('e', 18))

second\_tuple = (('d', 1), ('e', 18), ('c', 23), ('b', 37), ('a', 53))

first\_tuple\_sorted = sorted(first\_tuple, key=lambda x: x[1])

second\_tuple\_sorted = sorted(second\_tuple, key=lambda x: x[1])

print("\n -------- Original tuple -------- \n")

print(first\_tuple)

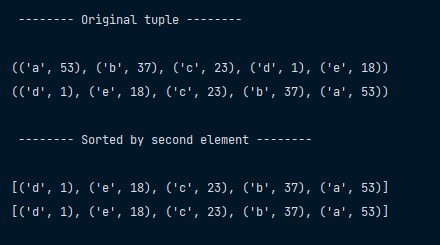
print(second\_tuple)

print("\n -------- Sorted by second element -------- \n")

print(first\_tuple\_sorted)

print(second\_tuple\_sorted)

**Output :**





**Q 2)** Write a Python program to replace last value of tuples in a list

**Code :**

ip\_tuple\_list = [(5, 2, 3), (4, 7, 6), (8, 9, 6)]

last\_tuple = list(ip\_tuple\_list[2])

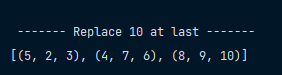
last\_tuple[2] = 10

ip\_tuple\_list[2]=tuple(last\_tuple)

print("\n ------- Replace 10 at last -------  ")

print(ip\_tuple\_list)

**Output :**





**Q 3)** Write a Python program to Sort Tuples by their Maximum element.

**Code :**

first\_tuple\_list = [(4, 5, 5, 7), (1, 3, 7, 4), (19, 4, 5, 3), (1, 2)]

second\_tuple\_list = [(19, 4, 5, 3), (4, 5, 5, 7), (1, 3, 7, 4), (1, 2)]

first\_tuple\_list\_sorted = sorted(first\_tuple\_list, key=lambda x: len(x))

second\_tuple\_list\_sorted = sorted(second\_tuple\_list, key=lambda x: len(x))

print("\n -------- Original tuple -------- \n")

print(first\_tuple\_list)

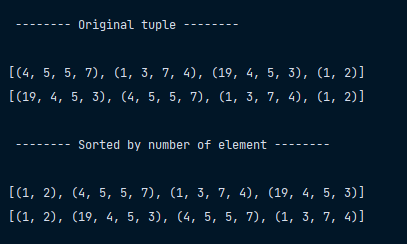
print(second\_tuple\_list)

print("\n -------- Sorted by number of element -------- \n")

print(first\_tuple\_list\_sorted)

print(second\_tuple\_list\_sorted)

**Output :**





**Q 4)** Write a Python program to Filter Tuples by Kth element from List

**Code :**

ip\_tuple\_list = [('B', 68), ('D', 70), ('A', 67), ('C', 69)]

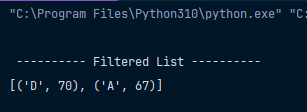
kth\_elements = [67, 70, 71, 75]

filtered\_tuple\_list = list(filter(lambda x: x[1] in kth\_elements, ip\_tuple\_list))

print(" \n ---------- Filtered List ---------- ")

print(filtered\_tuple\_list)

**Output :**





**Q 5)** Write a Python program to find maximum and the minimum value in a set.

**Code :**

ip\_set = set(input("Enter input element in coma separated value  : ").replace(" ", "").split(","))

ip\_sorted\_set = sorted(ip\_set)

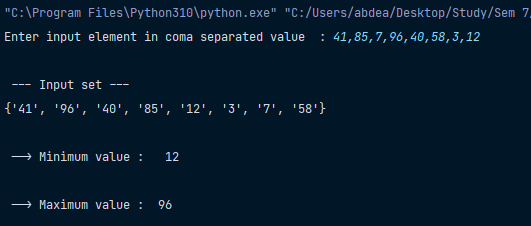
print("\n --- Input set --- ")

print(ip\_set)

print("\n --> Minimum value :  ", ip\_sorted\_set[0])

print("\n --> Maximum value : ", ip\_sorted\_set[-1])

**Output :**





**Q 6)** Write a Python program to Convert String to Set

**Code :**

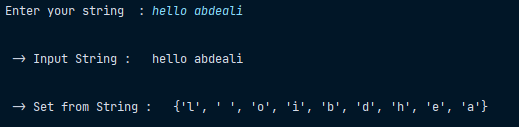
ip\_str = input("Enter your string  : ")

set\_from\_str = set(ip\_str)

print("\n -> Input String :  ", ip\_str)

print("\n -> Set from String :  ", set\_from\_str)

**Output :**



**Q 7)** Program to count number of vowels using sets in given string.

**Code :**

set\_from\_str = set(input("Enter your string to first convert to set :”)

print("\n -> Input Set :  ", set\_from\_str)

vowels = ['a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O', 'U']

vowels\_count = 0

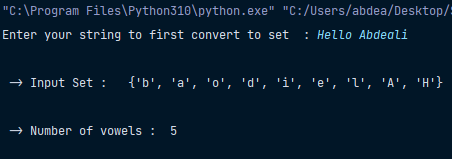
for x in set\_from\_str:

    if x in vowels:

        vowels\_count += 1

print("\n -> Number of vowels : ", vowels\_count)

**Output :**





**Q 8)** Write a Python program to Check if a set a subset of another set.

**Code :**

A = {4, 5}

B = {1, 2, 3, 4, 5}

print("\n -> Is A Subset Of B ? : ", A.issubset(B))

print(" -> Is B Subset Of A ? : ", B.issubset(A))

**Output :**

