



THE M.S. UNIVERSITY OF BARODA

Faculty of Tech. & Engg.- Vadodara

NAME: ABDEALI KHUJEMABHAI CHHARCHHODAWALA

BRANCH: COMPUTER SCIENCE & ENGINEERING

PRN NO. : 2020033800098465

SEAT NO. : 612006(SEM-6) **CLASS :** BE-IV SEM-VII

MOBILE NO : 7434870160

E-MAIL ID: abdealiking786@gmail.com

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 :

```
C:\Users\abdea\Desktop\Study\Sem 7\Python\Labs\Lab-1>python Lab-1_1.py
Welcome Ruchita to Salary Management
Enter your Location : bangalore
Enter your salary : 75000
I can't work here
```

Output 1.2 :

```
C:\Users\abdea\Desktop\Study\Sem 7\Python\Labs\Lab-1>python Lab-1_1.py
Welcome Ruchita to Salary Management
Enter your Location : hyderabad
Enter your salary : 2000000
Yes I am ready to work here in  HYDERABAD
```

Output 1.3 :

```
C:\Users\abdea\Desktop\Study\Sem 7\Python\Labs\Lab-1>python Lab-1_1.py
Welcome Ruchita to Salary Management
Enter your Location : surat
Enter your salary : 2000000
Yes I am ready to work here in  SURAT
```

Output 1.4 :

```
C:\Users\abdea\Desktop\Study\Sem 7\Python\Labs\Lab-1>python Lab-1_1.py
Welcome Ruchita to Salary Management
Enter your Location : space
Enter your salary : 2000
Yes I am very happy to work in Space
```

Output 1.5 :

```
C:\Users\abdea\Desktop\Study\Sem 7\Python\Labs\Lab-1>python Lab-1_1.py
Welcome Ruchita to Salary Management
Enter your Location : dahod
Enter your salary : 1500000
I can't work here
```

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 :

```
C:\Users\abdea\Desktop\Study\Sem 7\Python\Labs\Lab-1>python Lab-1_2.py
['e', 'k', 'h', 'y', 'd', 't']
```

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 :

```
C:\Users\abdea\Desktop\Study\Sem 7\Python\Labs\Lab-1>python Lab-1_3.py
[10, 20, [300, 400, [5000, 6000, 7000], 500], 30, 40]
```

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 :

```
abdea@Abdeali-PC MINGW64 ~/Desktop/Study/Sem 7/Python/Labs/Lab-1
$ python Lab-1_4.py
Enter your String : Hello my name is Abdeali!

Before Removing vowels :
Hello my name is Abdeali!

After Removing vowels :
Hll my nm s bdl!
```

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 :

```
abdea@Abdeali-PC MINGW64 ~/Desktop/Study/Sem 7/Python/Labs/Lab-1
$ python temp.py
Enter total length : 5
Enter number : 1
Enter number : 2
Enter number : 3
Enter number : 4
Enter number : 5
[1, 4, 9, 16, 25]
```

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 :

```
abdea@Abdeali-PC MINGW64 ~/Desktop/Study/Sem 7/Python/Labs/Lab-1
$ python Lab-1_6.py
10
5.0
6.0
3.0
4.0
2.0
1.0
2.0
1.0
2.0
1.0
```

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 $\text{current_savings} * r / 12$ 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 ($\text{annual_salary} / 12$).
- 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 : "))

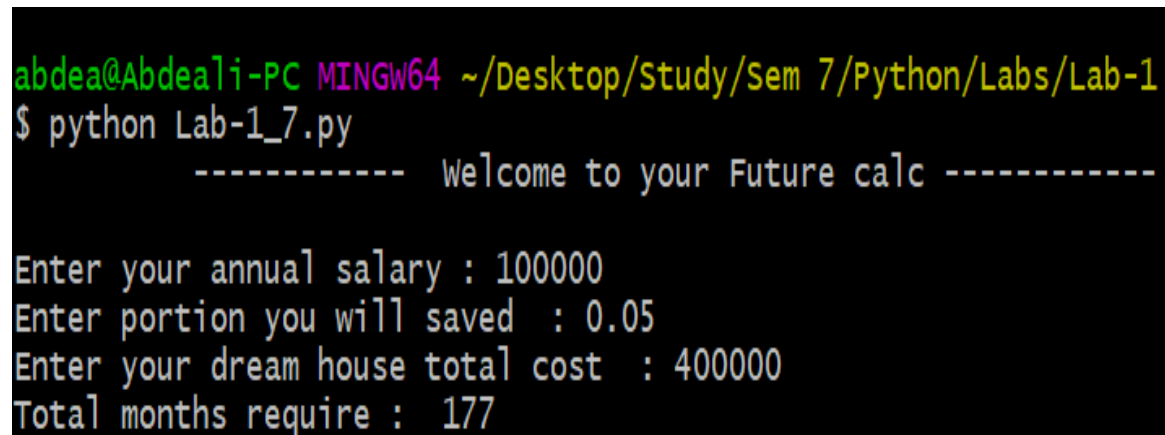
current_saving = 0
total_months_to_reach = 0

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 :



```
abdea@Abdeali-PC MINGW64 ~/Desktop/Study/Sem 7/Python/Labs/Lab-1
$ python Lab-1_7.py
----- Welcome to your Future calc -----

Enter your annual salary : 100000
Enter portion you will saved : 0.05
Enter your dream house total cost : 400000
Total months require : 177
```


Q 7.2)

In Part A, we unrealistically assumed that your salary didn't change. But clearly you are going to be worth 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 to 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:

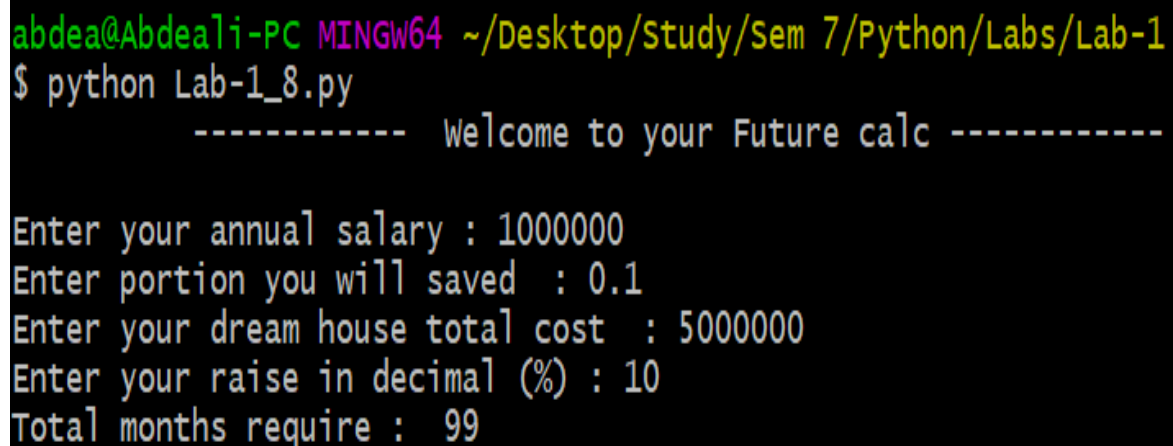
    if total_months_to_reach in [6, 12, 18]:
        annual_salary += annual_salary * semi_annual_raise
        portion_saved_salary = (annual_salary / 12) *
portion_saved

        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 :



```
abdea@Abdeali-PC MINGW64 ~/Desktop/Study/Sem 7/Python/Labs/Lab-1
$ python Lab-1_8.py
----- Welcome to your Future calc -----

Enter your annual salary : 1000000
Enter portion you will saved : 0.1
Enter your dream house total cost : 5000000
Enter your raise in decimal (%) : 10
Total months require : 99
```

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 :

```
abdea@Abdeali-PC MINGW64 ~/Desktop/Study/Sem 7/Python/Labs/Lab-1
$ python Lab-1_9.py
Enter a string : mam is si mam
Enter total length of List : 5
Enter string : mom
Enter string : jack
Enter string : nun
Enter string : help
Enter string : abba
Test : mamissimam

String is Palindrome

=== Total Palindrome strings ===
['mom', 'nun', 'abba']
```

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 :

```
abdea@Abdeali-PC MINGW64 ~/Desktop/Study/Sem 7/Python/Labs/Lab-1
$ python Lab-1_10.py
Original list :
[34, 78, -12, 44, 78, 91, 60, -34, 88]

[474552, 474552, 753571, 216000, 681472]
[34, 78, 44, 78, 91, 60, 88]
[34, 78, -12, 78, 91, 60, -34, 88]
[34, 78, -12, 78, 91, 60, -34]
```

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 :

```
abdea@Abdeali-PC MINGW64 ~/Desktop/Study/Sem 7/Python/Labs/Lab-1
$ python Lab-1_11.py
Enter your Sting : Hello, My name is Abdeali

Input String : Hello, My name is Abdeali
List of String : ['Hello,', 'My', 'name', 'is', 'Abdeali']

Enter total length of List : 4
Enter string in list : My
Enter string in list : name
Enter string in list : is
Enter string in list : Mariyam

Input list : ['My', 'name', 'is', 'Mariyam']
String from list : My name is Mariyam
```

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 :

```
abdea@Abdeali-PC MINGW64 ~/Desktop/Study/Sem 7/Python/Labs/Lab-2  
$ python Lab-2_1.py  
  
Enter total length of List : 5  
Enter number in list : 23  
Enter number in list : 78  
Enter number in list : 96  
Enter number in list : 45  
Enter number in list : 90  
Maximum Number by Recursion : 96
```

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:

```

abdea@Abdeali-PC MINGW64 ~/Desktop/Study/Sem 7/Python/Labs/Lab-2
$ python Lab-2_2.py

Enter total length of List : 5
Enter number in list : 89
Enter number in list : 56
Enter number in list : 89
Enter number in list : 23
Enter number in list : 12
Enter Remove item element : 89
New List after removing [ 89 ] : [56, 23, 12]

```

Q 3)

Write a function `get_digit(n, left, right)` . The function returns

- the `d`-th digit of `n` from the left if the function was called as `get_digit(n, left=d)` . If the required digit doesn't exist, the function must return -1.
- the `d`-th digit of `n` from the right if the function was called as `get_digit(n, right=d)` . If the required digit doesn't exist, the function must return 0.
- -1 otherwise, i.e., for calls `get_digit(n)` and `get_digit(n, left=d1, right=d2)` .

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 :

```
abdea@Abdeali-PC MINGW64 ~/Desktop/Study/Sem 7/Python/Labs/Lab-2
$ python Lab-2_3.py
Enter number : 7896452
Enter direction (left or right) : left
Enter position : 2
=> 2 th digit from left of 7896452 is : 8
Test case value : -1

abdea@Abdeali-PC MINGW64 ~/Desktop/Study/Sem 7/Python/Labs/Lab-2
$ python Lab-2_3.py
Enter number : 7896452
Enter direction (left or right) : right
Enter position : 3
=> 3 th digit from left of 7896452 is : 4
Test case value : -1
```

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 :

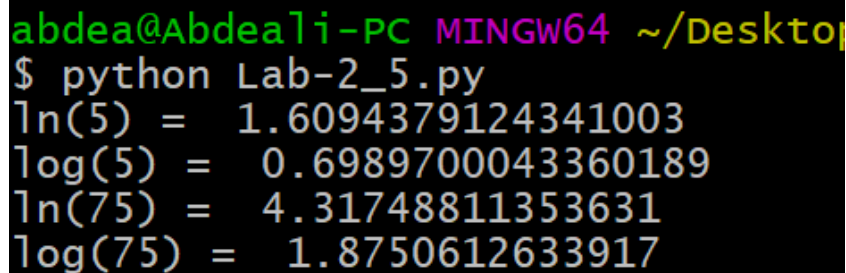
```
abdea@Abdeali-PC MINGW64 ~/Desktop/Study/Sem 7/Python/Labs/Lab-2
$ python Lab-2_4.py
=> LCM is : 3418800
=> GCD is : 2
```

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 :



```
abdea@Abdeali-PC MINGW64 ~/Desktop
$ python Lab-2_5.py
ln(5) = 1.6094379124341003
log(5) = 0.6989700043360189
ln(75) = 4.31748811353631
log(75) = 1.8750612633917
```

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 assignment2 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

Name	Date modified	Type	Size
Assignment2_10072023.docx	10-Jul-23 3:29 PM	Microsoft Word D...	60 KB
Lab-2_1.py	10-Jul-23 8:27 PM	PY File	1 KB
Lab-2_2.py	10-Jul-23 3:27 PM	PY File	1 KB
Lab-2_3.py	10-Jul-23 8:32 PM	PY File	1 KB
Lab-2_4.py	10-Jul-23 3:56 PM	PY File	1 KB
Lab-2_5.py	10-Jul-23 3:59 PM	PY File	1 KB
Lab-2_6.py	10-Jul-23 8:43 PM	PY File	2 KB
Lab-2_7.py	10-Jul-23 6:30 PM	PY File	1 KB










Output 6.2 : Inputs

```

abdea@Abdeali-PC MINGW64 ~/Desktop/Study/Sem 7/Python/Labs/Lab-2
$ python Lab-2_6.py
Enter student Name : Abdeali
Enter birthday (dd-mm-YYYY) : 28-12-2002
Enter student Name : Husain
Enter birthday (dd-mm-YYYY) : 16-10-2002
Enter student Name : Mohmmad
Enter birthday (dd-mm-YYYY) : 30-08-2002
File created: StudentDetails.txt
Enter student Name : Yusuf
Enter birthday (dd-mm-YYYY) : 25-02-2000
Enter student Name : Mustan
Enter birthday (dd-mm-YYYY) : 12-01-2000
Name: Abdeali, Birthdate: 28-12-2002
Name: Husain, Birthdate: 16-10-2002
Name: Mohmmad, Birthdate: 30-08-2002
Name: Yusuf, Birthdate: 25-02-2000
Name: Mustan, Birthdate: 12-01-2000
Student Names: ['Abdeali', 'Husain', 'Mohmmad', 'Yusuf', 'Mustan']

```

Output 6.3 : txt file exist

Name	Date modified	Type	Size
 Assignment2_10072023.docx	10-Jul-23 3:29 PM	Microsoft Word D...	60 KB
 Lab-2_1.py	10-Jul-23 8:27 PM	PY File	1 KB
 Lab-2_2.py	10-Jul-23 3:27 PM	PY File	1 KB
 Lab-2_3.py	10-Jul-23 8:32 PM	PY File	1 KB
 Lab-2_4.py	10-Jul-23 3:56 PM	PY File	1 KB
 Lab-2_5.py	10-Jul-23 3:59 PM	PY File	1 KB
 Lab-2_6.py	10-Jul-23 8:43 PM	PY File	2 KB
 Lab-2_7.py	10-Jul-23 6:30 PM	PY File	1 KB
 StudentDetails.txt	10-Jul-23 8:47 PM	Text Document	1 KB

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 :

```
abdea@Abdeali-PC MINGW64 ~/Desktop/Study/Sem 7/Python/Labs/Lab-2
$ python Lab-2_7.py
Enter Number : 4562
Original value : 4562
Reverse value : 2654

abdea@Abdeali-PC MINGW64 ~/Desktop/Study/Sem 7/Python/Labs/Lab-2
$ python Lab-2_7.py
Enter Number : -4562
Original value : -4562
Reverse value : -2654
```

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 :

```
"C:\Program Files\Python310\python.exe" "C:/Users/abdea/Desktop/Study/Sem 7/Python/Labs/Lab-3/Ass-3-Using-CSV-Module.py"
Longest Distance Journey in miles: 23364.0

Longest return journey in miles: 23364.0

Longest single journey in miles: 5035.0

Carrier whom we paid the most: BRITISH AIRWAYS

Carriers whom we flew the most number of times: AIR ASIA
```

```
List of all distinct carriers: {'EASJET', 'ICELANDAIR', 'UNITED AIRLINES', 'CONTINENTAL AIRLINES', 'RYANAIR', 'SINGAPORE AIRLINES', 'AIR LANKA', 'EMIRATES', 'KENYA AIRWAYS', 'LAN CHILE', 'AMERICAN AIRLINES', 'DELTA AIR LINES INC.', 'SOUTH AFRICAN AIRWAYS', 'SN BRUSSELS AIRLINES', 'KOREAN AIR', 'THOMAS COOK AIRLINES', 'AIR INDIA', 'EASTERN AIRWAYS', 'ASIANA AIRLINES INC.', 'AIR BERLIN', 'AIR TRANSAT', 'ETIHAD AIRWAYS', 'MALAYSIA AIRLINES SYSTEMS', 'CATHAY PACIFIC', 'CYPRUS AIRWAYS', 'ALITALIA', 'MISC SUPPLIERS', 'EL AL ISRAEL AIR LINES', 'AIR ASIA', 'HANK', 'AIR CANADA', 'VIRGIN ATLANTIC', 'SCANDINAVIAN AIRLINES', 'EGYPTAIR', 'BHI BABY', 'THOMSON FLY', 'CHINA SOUTHERN AIRLINES', 'QATAR AIRWAYS (W.L.L)', 'PULKOVO AVIATION ENTERPR', 'AUSTRIAN AIRLINES', 'BRITISH AIRWAYS INTERNET BOOKI', 'TURKISH AIRLINES INC.', 'VIETNAM AIRLINES', 'JET 2', 'AIR FRANCE', 'KLM', 'TAP - AIR PORTUGAL', 'JET AIRWAYS', 'IBERIA', 'AURIGNY AIR SERVICES', 'TAM - LINHAS AEREAS', 'LUFTHANSA', 'AIR CHINA', 'YUGOSLAV AIRLINES', 'AEGEAN AVIATION S.A', 'QANTAS', 'VIRGIN NIGERIA AIRWAYS', 'INDIGO', 'AIR BALTIC', 'ROYAL AIR MAROC', 'BRITISH AIRWAYS', 'Selly Helicopters', 'FINNAIR', 'SWISS AIRLINES', 'AER LINGUS', 'WIZZAIR', 'AIR SOUTHWEST'}
```

```
List of all distinct destinations: {'HONG KONG', 'KOS', 'DURBAN', 'BANGALORE', 'ACCRA', 'MALPENZA', 'FRANKFURT', 'STANSTED', 'HOUSTON', 'LEEDS', 'BENBECULA', 'PESCARA', 'KABUL', 'LISBON', 'DENVER', 'SINGAPORE', 'WASHINGTON', 'BELGRADE', 'LERWICK', 'STOCKHOLM', 'GUERNSEY', 'DUBAI', 'NEW ORLEANS', 'EL PASO', 'PALERMO', 'KINGSTON', 'PRAGUE', 'TRIESTE', 'BERGEN', 'GLASGOW', 'ATHENS', 'DAMMAM', 'REGGIO CALABRIA', 'BIRMINGHAM', 'BELFAST INTL', 'BERLIN', 'GIBRALTAR', 'PARIS', 'FREETOWN', 'RHODES', 'KANSAS CITY', 'SEOUL', 'DUSHANBE', 'GRAND CAYMEN', 'BRUSSELS', 'INVERNESS', 'PERTH AUS', 'KRISTIANSUND', 'OUARZAZATE', 'OMAHA', 'CAMPBELTOWN', 'HELSINKI', 'MUNICH', 'STORNOWAY', 'SAN FRANCISCO', 'CAPE TOWN', 'CHICAGO', 'SYDNEY', 'BRISBANE', 'DAKAR', 'LUTON', 'TOULOUSE', 'OTTAWA', 'LUXEMBOURG', 'INDIANAPOLIS', 'HALIFAX CANADA', 'KIGALI RWANDA', 'BELFAST CITY', 'CAIRO', 'MELBOURNE', 'GENEVA', 'TIREE', 'JOBURG', 'PAPHOS', 'DUBLIN', 'FLORENCE', 'CRETE', 'HANOI', 'TALLINN', 'MUSCAT', 'TAIPEI', 'KEFLAVIK INTERNATIONAL', 'SAN DIEGO', 'FUERTEVENTURA', 'BEIJING', 'ISLE OF MAN', 'ISTANBUL', 'ALBUQUERQUE', 'BRISTOL', 'LONDON - LGW', 'GDANSK', 'TOKYO - NARITA', 'TRONDHEIM', 'SANTIAGO', 'LIVERPOOL', 'COLONSAY IS', 'KUALA LUMPUR', 'HANOVER', 'SOUTHAMPTON', 'NAIROBI', 'EXETER', 'KIEV', 'TORONTO', 'COLOGNE', 'LARNACA', 'PITTSBURGH', 'COLOMBO', 'BUDAPEST', 'SOFIA', 'BUCHAREST', 'HAMBURG', 'CORK', 'MANCHESTER', 'TEL AVIV YAFD', 'LAGOS', 'CALGARY', 'RIGA', 'GUARULHOS INTL', 'HERAKLION', 'NULL', 'LEIPZIG', 'PENZANCE', 'VIENNA', 'WARSAW', 'MALAGA', 'AMSTERDAM', 'BOSTON', 'BASLE', 'BARCELONA', 'KIRKWALL', 'COPENHAGEN', 'JERSEY', 'ABERDEEN', 'ROME - CIAMPINO', 'ROME', 'AUCKLAND', 'RENNES', 'HUMBERSIDE', 'MANILA', 'NEW YORK - JFK', 'NEWCASTLE', 'LOS ANGELES', 'ST PETERSBURG', 'MUMBAI', 'ISLES OF SCILLY', 'NEWARK', 'PALMA MALLORCA', 'EAST MIDLANDS', 'LONDON - CITY', 'NORWICH', 'GLENEGDALE', 'DUSSELDORF', 'NEWQUAY', 'MONTREAL', 'ZURICH', 'STAVANGER', 'HILO', 'LONDON - LHR', 'OLANCHITO', 'SAO TOME', 'OSLO', 'BANGKOK', 'SEATTLE', 'BOLOGNA', 'VENICE', 'EDINBURGH', 'WASHINGTON - NATIONAL', 'CARDIFF'}
```

```
Mean price of all single route ticket prices: 119.54755208333343
```

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 :

```
"C:\Program Files\Python310\python.exe" "C:/Users/abdea/Desktop/Study/Sem 7/Python/Labs/Lab-3/Ass-3-Using-Json.py"
[{'Customer': 'Met Office', 'Ticket Price ex VAT': 194.48, 'Number of Travellers': 1, 'Total Cost ex VAT': 194.48, 'Ticket Type': 'Ticket', 'Ticket Single or Return': 'Single', 'Travel Class': 'ECONOMY',
'Travel Date': '02-02-2011', 'Miles Travelled': 440, 'Journey Start Point': 'SOUTHAMPTON', 'Journey Finish Point': 'FRANKFURT', 'Air Carrier': 'AIR ASIA'}, {'Customer': 'Met Office', 'Ticket Price ex
VAT': -93.99, 'Number of Travellers': -1, 'Total Cost ex VAT': -93.99, 'Ticket Type': 'Ticket', 'Ticket Single or Return': 'Single', 'Travel Class': 'ECONOMY', 'Travel Date': '02-02-2011', 'Miles
Travelled': -440, 'Journey Start Point': 'SOUTHAMPTON', 'Journey Finish Point': 'FRANKFURT', 'Air Carrier': 'AIR ASIA'}, {'Customer': 'Met Office', 'Ticket Price ex VAT': 2.0, 'Number of Travellers': 1,
'Total Cost ex VAT': 2.0, 'Ticket Type': 'Ticket', 'Ticket Single or Return': 'Single', 'Travel Class': 'ECONOMY', 'Travel Date': '07-02-2011', 'Miles Travelled': 0, 'Journey Start Point': 'EXETER',
'Journey Finish Point': 'JERSEY', 'Air Carrier': 'AIR ASIA'}, {'Customer': 'Met Office', 'Ticket Price ex VAT': 93.03, 'Number of Travellers': 1, 'Total Cost ex VAT': 93.03, 'Ticket Type': 'Ticket',
'Ticket Single or Return': 'Return', 'Travel Class': 'ECONOMY', 'Travel Date': '21-02-2011', 'Miles Travelled': 219, 'Journey Start Point': 'GLASGOW', 'Journey Finish Point': 'BELFAST INTL', 'Air
Carrier': 'EASYJET'}, {'Customer': 'Met Office', 'Ticket Price ex VAT': 55.51, 'Number of Travellers': 1, 'Total Cost ex VAT': 55.51, 'Ticket Type': 'Ticket', 'Ticket Single or Return': 'Single', 'Travel
Class': 'ECONOMY', 'Travel Date': '05-03-2011', 'Miles Travelled': 256, 'Journey Start Point': 'NEWCASTLE', 'Journey Finish Point': 'BRISTOL', 'Air Carrier': 'EASYJET'}, {'Customer': 'Met Office',
'Ticket Price ex VAT': 94.03, 'Number of Travellers': 1, 'Total Cost ex VAT': 94.03, 'Ticket Type': 'Ticket', 'Ticket Single or Return': 'Return', 'Travel Class': 'ECONOMY', 'Travel Date': '20-03-2011',
'Miles Travelled': 858, 'Journey Start Point': 'INVERNESS', 'Journey Finish Point': 'BRISTOL', 'Air Carrier': 'EASYJET'}, {'Customer': 'Met Office', 'Ticket Price ex VAT': 344.38, 'Number of Travellers':
1, 'Total Cost ex VAT': 344.38, 'Ticket Type': 'Ticket', 'Ticket Single or Return': 'Return', 'Travel Class': 'ECONOMY', 'Travel Date': '07-04-2011', 'Miles Travelled': 4311, 'Journey Start Point':
'MANCHESTER', 'Journey Finish Point': 'LARNACA', 'Air Carrier': 'AIR INDIA'}, {'Customer': 'Met Office', 'Ticket Price ex VAT': 318.03, 'Number of Travellers': 1, 'Total Cost ex VAT': 318.03, 'Ticket
Type': 'Ticket', 'Ticket Single or Return': 'Return', 'Travel Class': 'ECONOMY', 'Travel Date': '10-04-2011', 'Miles Travelled': 470, 'Journey Start Point': 'BENBECULA', 'Journey Finish Point':
```

ASSIGNMENT-4

Q 1) Create a class Account with 2 instance attributes 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 python and comment?
- ➔ Check if over riding is allowed is allowed in python 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)

```

Theoretical 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

```
"C:\Program Files\Python310\python.exe" "C:/Users/abdea/Desktop/Study/Sem
```

```
5500 accepted successfully!!
```

```
Total balance : 6000
```

```
500 withdraw successfully!!
```

```
Total balance : 5500
```

```
4000 accepted successfully!!
```

```
Total balance : 5000
```

```
4000 withdraw successfully!!
```

```
Total balance : 1500
```

```
4000 /- transfer successfully to Husain
```

```
Total balance : 1500
```

```
Balance of Abdeali : 1500
```

```
Balance of Husain : 5000
```

```
-----
```

```
Saving Account holder name 'Mohammad' and current balance is ' 1000 /- '
```

```
Interest added , current balance : 1050.0
```

```
500 accepted successfully!!
```

```
Total balance : 5500
```

```
500 withdraw successfully!!
```

```
Total balance : 550.0
```

```
500 /- transfer successfully to Husain
```

```
Total balance : 550.0
```

```
Process finished with exit code 0
```

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")

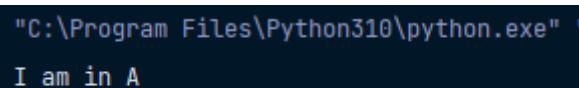
class C(A, B):
    pass

c = C()
c.say_name()
```

Theoretical 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 :



```
"C:\Program Files\Python310\python.exe" "  
I am in A
```


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 :

```
"C:\Program Files\Python310\python.exe" "C:/Users/abdea/Desktop

-> Distance between Point-1 and Point-2 : 10

Point count : 2

Distance finding with Manhattan"
```

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

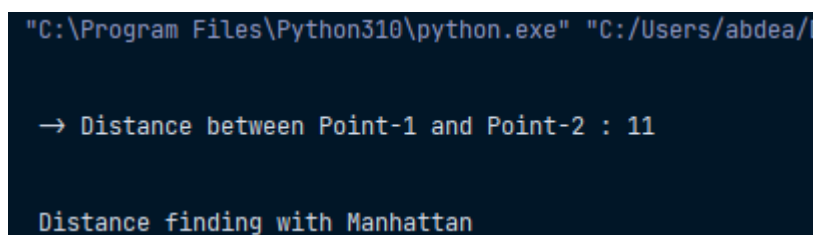
    @staticmethod
    def get_method_name():
        print("\n Name is Manhattan")

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 :



```
"C:\Program Files\Python310\python.exe" "C:/Users/abdea/...

→ Distance between Point-1 and Point-2 : 11

Distance finding with Manhattan
```

ASSIGNMENT-5

Q1) 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 :

```
Enter number of rows(x) : 3
Enter number of columns(x) : 3

Generated 2D array :

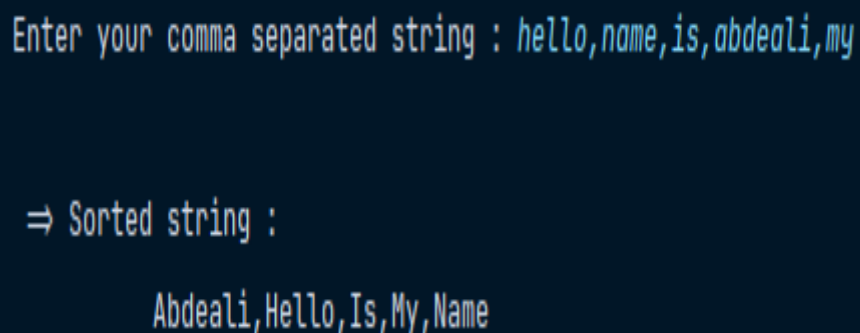
[[0 0 0]
 [0 1 2]
 [0 2 4]]
```

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 :

A screenshot of a terminal window with a dark blue background. The text is displayed in a light blue monospaced font. The first line shows the prompt 'Enter your comma separated string :' followed by the input 'hello,name,is,abdeali,my'. The second line shows the output '=> Sorted string :'. The third line shows the sorted and capitalized output 'Abdeali,Hello,Is,My,Name' with tabs between the words.

```
Enter your comma separated string : hello,name,is,abdeali,my  
  
=> Sorted string :  
    Abdeali,Hello,Is,My,Name
```

Q3) 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 :

```
Enter your comma separated list : 1,2,3,4,5  
  
=> Output :  
      1,9,25
```

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

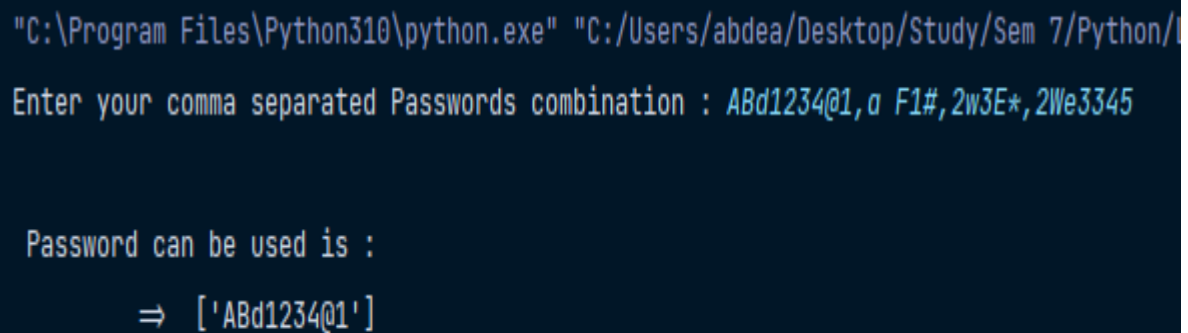
    return False
```

```
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 :



```
"C:\Program Files\Python310\python.exe" "C:/Users/abdea/Desktop/Study/Sem 7/Python/1
Enter your comma separated Passwords combination : ABd1234@1,a F1#,2w3E*,2We3345

Password can be used is :
      ⇒ ['ABd1234@1']
```

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 :

```
"C:\Program Files\Python310\python.exe" "C:/Users/abdea/Desktop/Study/Sem 7/Python/Labs/Lab-5/Lab_5_5.py"
Enter your comma separated data with white space : Tom,19,80 John,20,90 Jony,17,91 Jony,17,93 Json,21,85

Answer :
      [('John', '20', '90'), ('Jony', '17', '91'), ('Jony', '17', '93'), ('Json', '21', '85'), ('Tom', '19', '80')]
```


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) :

```
"C:\Program Files\Python310\python.exe" "C:/Users/abdea/Desktop/Study/Sem 7/Pyt
Enter your comma separated data with sorted order : 11,21,31,41,51
Enter number to find index : 11

Index of '11' is : 0
```

Output (Middle element) :

```
"C:\Program Files\Python310\python.exe" "C:/Users/abdea/Desktop/Study/Sem 7/Pyt
Enter your comma separated data with sorted order : 11,21,31,41,51
Enter number to find index : 31

Index of '31' is : 2
```

Output (Last element) :

```
"C:\Program Files\Python310\python.exe" "C:/Users/abdea/Desktop/Study/Sem 7/Pyt
Enter your comma separated data with sorted order : 11,21,31,41,51
Enter number to find index : 51

Index of '51' is : 4
```

Output (Element not available) :

```
"C:\Program Files\Python310\python.exe" "C:/Users/abdea/Desktop/Study/Sem 7/Pyt
Enter your comma separated data with sorted order : 11,21,31,41,51
Enter number to find index : 61

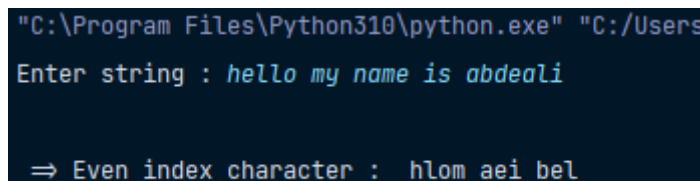
Number 61 is not in list
```

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 :



```
"C:\Program Files\Python310\python.exe" "C:/Users  
Enter string : hello my name is abdeali  
  
=> Even index character : h l o m a e i b e l
```

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 :

```
"C:\Program Files\Python310\python.exe" "C:/Users/abdea/Desktop/Study/Sem 7/
Enter your data to be encrypt : My name is Abdeali

⇒ Encrypted Data : 6d17916e16116d16516917316116216416516116c1691

⇒ Original data : mynameisabdeali
```

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 :

```
"C:\Program Files\Python310\python.exe" "C:/Users/abdea/Desktop/Study/S
Enter string : Hello My name is Abdeali
String is accepted
```

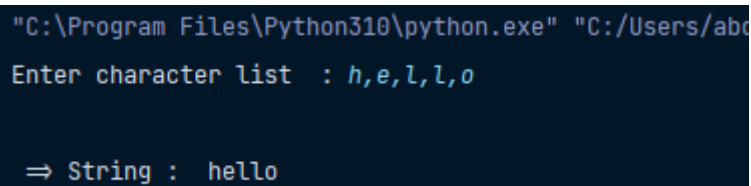
Q 10) Write a Python program to convert a list of characters into a string.

Code :

```
op_string = "".join([x for x in input("Enter character list : ").replace(" ", "").split(",")])

print("\n => String : ", op_string)
```

Output :



```
"C:\Program Files\Python310\python.exe" "C:/Users/ab...
Enter character list : h,e,l,l,o

=> String : hello
```

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

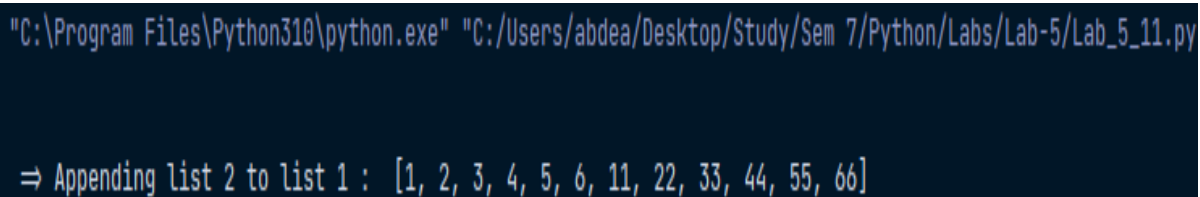
Code :

```
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 :



```
"C:\Program Files\Python310\python.exe" "C:/Users/abdea/Desktop/Study/Sem 7/Python/Labs/Lab-5/Lab_5_11.py

=> Appending list 2 to list 1 : [1, 2, 3, 4, 5, 6, 11, 22, 33, 44, 55, 66]
```

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 :

```
"C:\Program Files\Python310\python.exe" "C:/Users/abdea/Desktop/Study/Sem
Enter word : Abdeali

=> Reverse Word : ilaedbA
```

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 :

```
"C:\Program Files\Python310\python.exe" "C:/Users/abdea/Desktop/
Enter sentence : Hello My name is Abdeali

=> Smallest word : My
```

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:

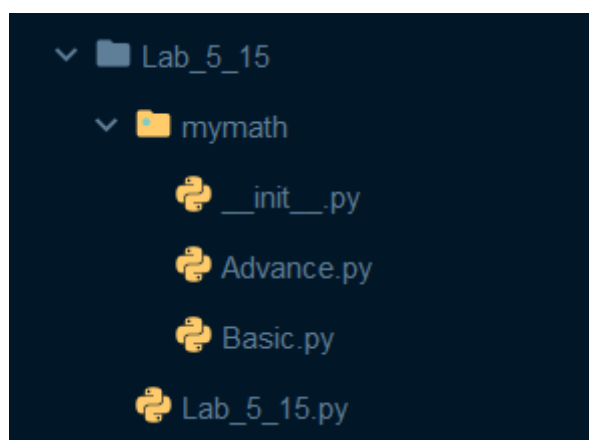
    print(f" => Character : {x}  -> Ascii : {ord(x)} ")
```

Output :

```
⇒ Enter your String : hello
⇒ Character : h  → Ascii : 104
⇒ Character : e  → Ascii : 101
⇒ Character : l  → Ascii : 108
⇒ Character : l  → Ascii : 108
⇒ Character : o  → Ascii : 111
```

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

Folder Structure :



Advance.py :

```
import math

def getGCD(a, b):
    return math.gcd(a, b)

def getLCM(a, b):
    return math.lcm(a, b)
```

Baisc.py :

```
def addNumber(a, b):
    return a + b

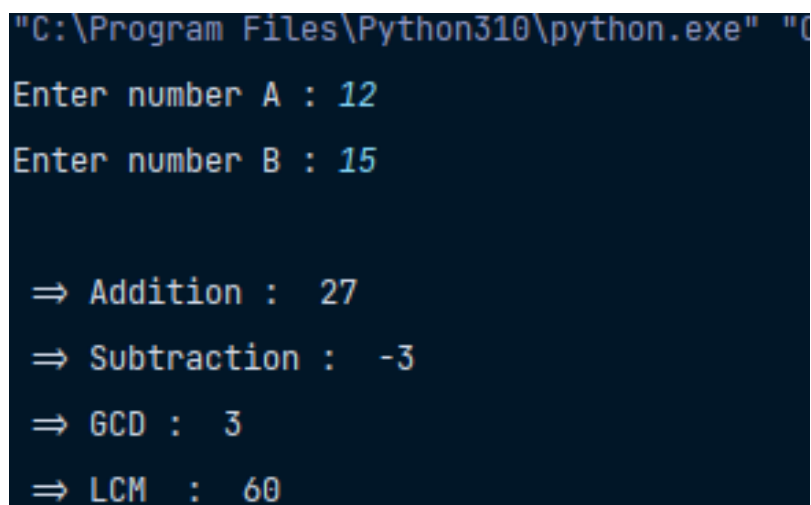
def subNumber(a, b):
    return a - b
```

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 :

```
C:\Program Files\Python310\python.exe "C:\Program Files\Python310\python.exe"
Enter number A : 12
Enter number B : 15

=> Addition : 27
=> Subtraction : -3
=> GCD : 3
=> LCM : 60
```

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

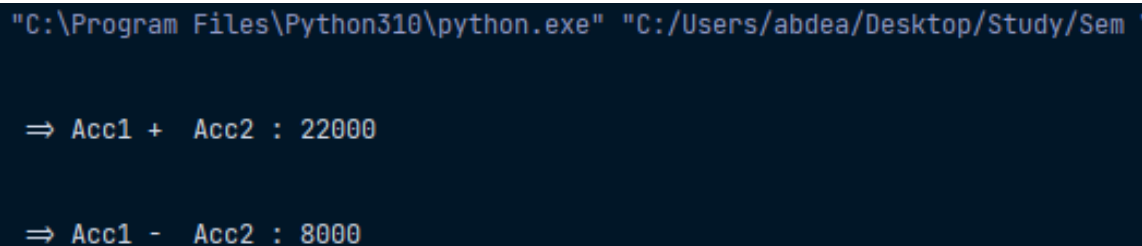
    def __add__(self, other):
        return self.balance + other.balance

    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 :



```
"C:\Program Files\Python310\python.exe" "C:/Users/abdea/Desktop/Study/Sem 1

=> Acc1 +  Acc2 : 22000

=> Acc1 -  Acc2 : 8000
```

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 :

```
"C:\Program Files\Python310\python.exe" "C:/Users/abdea/Desktop/Study/Sem 7/Python/

Area of square : 25
Perimeter of square : 20

Area of Rectangle : 35
Perimeter of Rectangle : 24
```

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 :

```

"C:\Program Files\Python310\python.exe" "C:/Users/
=> Mean is : 43.38461538461539

=> Median is : 44

```

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 :

```
"C:\Program Files\Python310\python.exe" "C:/Users/abdea/Desktop/S
[1] Mustan has an average score above 90: 97.0
[2] Yusuf has an average score above 90: 92.0
[3] Mohammad has an average score above 90: 91.0
```

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 :

```
"C:\Program Files\Python310\python.exe" "C:/Users/abdea/Desktop/S
=> Mapped Array : [4, 8, 12, 16, 20]
```

User define Filter :

```
ip_number = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

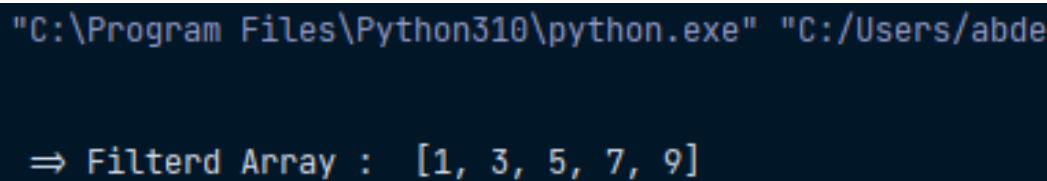
```
def isOdd(num):  
    return num % 2 == 1
```

```
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 :



```
"C:\Program Files\Python310\python.exe" "C:/Users/abde  
  
=> Filterd Array : [1, 3, 5, 7, 9]
```


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) :

The screenshot shows the 'Router Manager' application window. It features a search section with 'Search by Host-Name' and 'Search by Query' options. Below this are input fields for 'Id', 'Host-Name', 'Brand', 'RAM', and 'Flash'. At the bottom, there are buttons for 'Add Router', 'Remove Router', 'Update Router', and 'Clear Input'. A table displays the current data:

id	Hostname	Brand	Ram	Flash
56	hello	google	78	56
78	Musatr	78	50	41
90	Yusuf	all	50	41
487	ok	ko	56	56

Output (Search by hostname) :

The screenshot shows the 'Router Manager' application window after a search for 'Musatr' in the 'Host-Name' field. The table now displays only the results for 'Musatr':

id	Hostname	Brand	Ram	Flash
78	Musatr	78	50	41

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 :

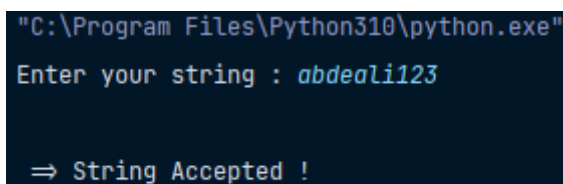
```
import re

pattern_string = r'[a-zA-z0-9]+'
pattern = re.compile(pattern_string)

ip_string = input("Enter your string : ")

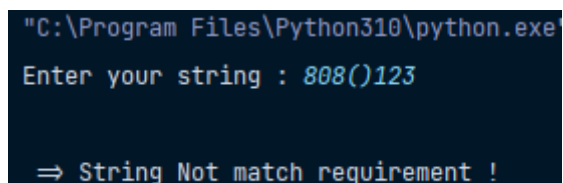
if pattern.fullmatch(ip_string):
    print("\n => String Accepted !")
else:
    print("\n => String Not match requirement !")
```

Output :



```
"C:\Program Files\Python310\python.exe"
Enter your string : abdeali123

=> String Accepted !
```



```
"C:\Program Files\Python310\python.exe"
Enter your string : 808()123

=> String Not match requirement !
```

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 : ")

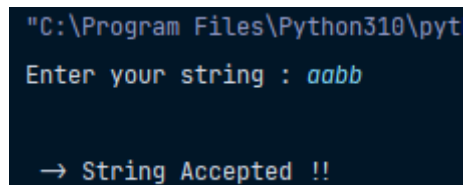
pattern_string = r'(ab*)'

pattern = re.compile(pattern_string, re.IGNORECASE)

if pattern.search(input_string):
    print("\n -> String Accepted !!")

else:
    print("\n -> String not Accepted")
```

Output :



```
"C:\Program Files\Python310\python.exe"
Enter your string : aabb

-> String Accepted !!
```

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 : ")

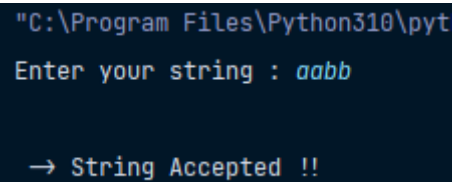
pattern_string = r'(ab+)'

pattern = re.compile(pattern_string, re.IGNORECASE)

if pattern.search(input_string):
    print("\n -> String Accepted !!")
```

```
else:  
    print("\n -> String not Accepted")
```

Output :



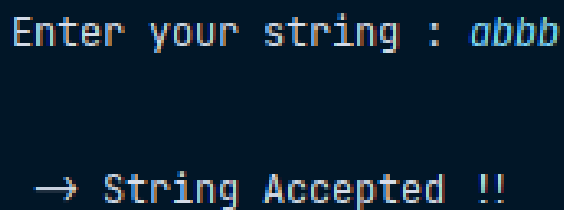
```
"C:\Program Files\Python310\pyt  
Enter your string : aabb  
  
→ String Accepted !!
```

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 :



```
Enter your string : abbb  
  
→ String Accepted !!
```

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 : ")

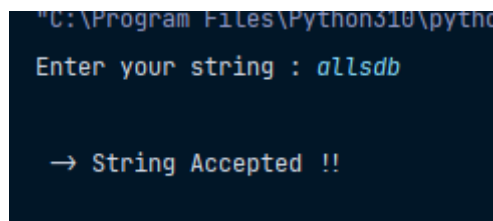
pattern_string = r'a[a-z]*b$'

pattern = re.compile(pattern_string, re.IGNORECASE)

if pattern.search(input_string):
    print("\n -> String Accepted !!")

else:
    print("\n -> String not Accepted")
```

Output :



```
"C:\Program Files\Python310\python
Enter your string : allbdb

-> String Accepted !!
```

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 :

```
"C:\Program Files\Python310\python.exe"  
Enter your string : Hello_wordl  
  
→ String Accepted !!
```

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]$\n'  
  
pattern = re.compile(pattern_string)  
  
if pattern.search(input_string):  
    print("\n -> String Accepted !!")  
  
else:  
    print("\n -> String not Accepted")
```

Output :

```
"C:\Program Files\Python310\python.exe"  
Enter your string : Hello123  
  
→ String Accepted !!
```

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

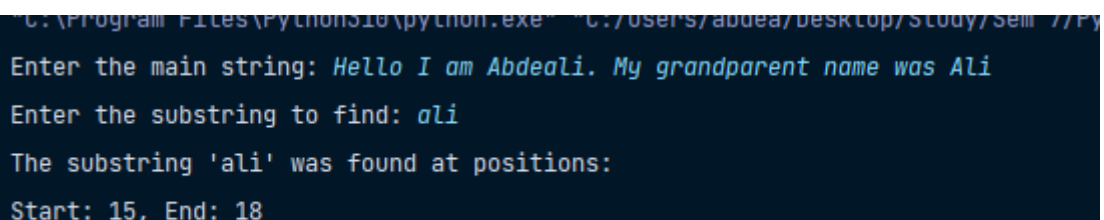
def main():
    main_string = input("Enter the main string: ")
    substring = input("Enter the substring to find: ")

    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 :



```
C:\Program Files\Python310\python.exe - "C:/Users/abdea/Desktop/Study/Sem 7/Py
Enter the main string: Hello I am Abdeali. My grandparent name was Ali
Enter the substring to find: ali
The substring 'ali' was found at positions:
Start: 15, End: 18
```


ASSIGNMENT-8

Q1) 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 :

```
"C:\Program Files\Python310\python.exe" "C:/Users/abdea/Desktop/Study/Sem 7/Python/Labs/Lab-
Time taken using list comprehension: 0.3403390999925527
Time taken using for loop: 0.36695240000153717

----- Profile -----

      5 function calls in 0.002 seconds

Ordered by: standard name

ncalls  tottime  percall  cumtime  percall filename:lineno(function)
      1   0.000    0.000    0.000    0.000 <string>:1(<module>)
      1   0.000    0.000    0.000    0.000 Lab_8_1.py:11(squares_list_comprehension)
      1   0.000    0.000    0.000    0.000 Lab_8_1.py:12(<listcomp>)
      1   0.001    0.001    0.002    0.002 {built-in method builtins.exec}
      1   0.000    0.000    0.000    0.000 {method 'disable' of '_lsprof.Profiler' object}

    1004 function calls in 0.001 seconds

Ordered by: standard name

ncalls  tottime  percall  cumtime  percall filename:lineno(function)
      1   0.000    0.000    0.001    0.001 <string>:1(<module>)
      1   0.001    0.001    0.001    0.001 Lab_8_1.py:16(squares_for_loop)
      1   0.000    0.000    0.001    0.001 {built-in method builtins.exec}
```

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 :

```
"C:\Program Files\Python310\python.exe" "C:/Users/abdea/De
----- Time -----
Time taken using join method: 0.0022801000013714656
Time taken using custom method: 0.002695000001040171

----- Result -----
Join method is more efficient.
```

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 :

```
"C:\Program Files\Python310\python.exe" "C:/Users/abdea/Desktop

----- Time -----
Time taken using old method :  0.002188200000091456
Time taken using efficient method :  0.0010270999991917051

----- Result -----
Efficient method is more efficient.
```

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 :

```
"C:\Program Files\Python310\python.exe" "C:/Users/abdea/Desktop

----- Time -----
Time taken using old method :  0.002188200000091456
Time taken using efficient method :  0.0010270999991917051

----- Result -----
Efficient method is more efficient.
```

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 :

```
"C:\Program Files\Python310\python.exe" "C:/Users/abdea/Desktop

----- Time -----
Time taken using old method :  0.002188200000091456
Time taken using efficient method :  0.0010270999991917051

----- Result -----
Efficient method is more efficient.
```

Q4) '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 :

```

---- Prime numbers between 2 to 4999 ----
2 , 3 , 5 , 7 , 11 , 13 , 17 , 19 , 23 , 29 , 31 , 37 , 41 , 43 , 47 , 53 , 59 , 61 , 67 , 71 , 73 , 79 , 83 , 89 , 97 , 101 , 103 , 107 , 109 , 113 , 127 , 131 , 137 , 139 , 149 , 151 , 157 , 163 , 167 ,
173 , 179 , 181 , 191 , 193 , 197 , 199 , 211 , 223 , 227 , 229 , 233 , 239 , 241 , 251 , 257 , 263 , 269 , 271 , 277 , 281 , 283 , 293 , 307 , 311 , 313 , 317 , 331 , 337 , 347 , 349 , 353 , 359 , 367 ,
373 , 379 , 383 , 389 , 397 , 401 , 409 , 419 , 421 , 431 , 433 , 439 , 443 , 449 , 457 , 461 , 463 , 467 , 479 , 487 , 491 , 499 , 503 , 509 , 521 , 523 , 541 , 547 , 557 , 563 , 569 , 571 , 577 , 587 ,
593 , 599 , 601 , 607 , 613 , 617 , 619 , 631 , 641 , 643 , 647 , 653 , 659 , 661 , 673 , 677 , 683 , 691 , 701 , 709 , 719 , 727 , 733 , 739 , 743 , 751 , 757 , 761 , 769 , 773 , 787 , 797 , 809 , 811 ,
821 , 823 , 827 , 829 , 839 , 853 , 857 , 859 , 863 , 877 , 881 , 883 , 887 , 907 , 911 , 919 , 929 , 937 , 941 , 947 , 953 , 967 , 971 , 977 , 983 , 991 , 997 , 1009 , 1013 , 1019 , 1021 , 1031 ,
1033 , 1039 , 1049 , 1051 , 1061 , 1063 , 1069 , 1087 , 1091 , 1093 , 1097 , 1103 , 1109 , 1117 , 1123 , 1129 , 1151 , 1153 , 1163 , 1171 , 1181 , 1187 , 1193 , 1201 , 1213 , 1217 , 1223 , 1229 , 1231 ,
1237 , 1249 , 1259 , 1277 , 1279 , 1283 , 1289 , 1291 , 1297 , 1301 , 1303 , 1307 , 1319 , 1321 , 1327 , 1361 , 1367 , 1373 , 1381 , 1399 , 1409 , 1423 , 1427 , 1429 , 1433 , 1439 , 1447 , 1451 , 1453 ,
1459 , 1471 , 1481 , 1483 , 1487 , 1489 , 1493 , 1499 , 1511 , 1523 , 1531 , 1543 , 1549 , 1553 , 1559 , 1567 , 1571 , 1579 , 1583 , 1597 , 1601 , 1607 , 1609 , 1613 , 1619 , 1621 , 1627 , 1637 , 1657 ,

```

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 :

```

C:\Program Files
0
2
4
6
8

```

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

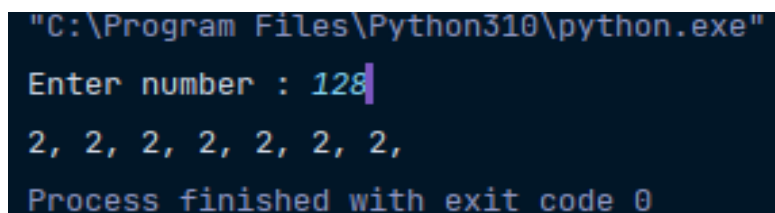
while True:
    number_to_factor = int(input("Enter number : "))

    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 :



```
"C:\Program Files\Python310\python.exe"
Enter number : 128
2, 2, 2, 2, 2, 2, 2,
Process finished with exit code 0
```

ASSIGNMENT-9

Q1) 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')

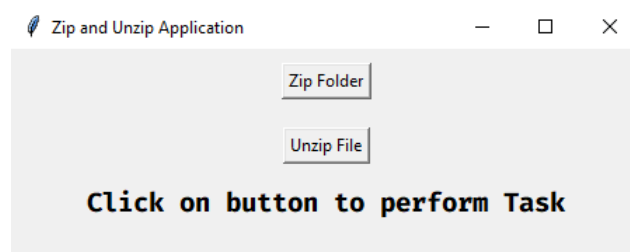
# buttons
zip_button = tk.Button(main_window, text="Zip Folder",
command=zip_folders)
unzip_button = tk.Button(main_window, text="Unzip File",
command=unzip_file)

# status label
status_label = tk.Label(main_window, text="Click on button to perform
Task", fg="black", font=("Fira Code", 14, "bold"))

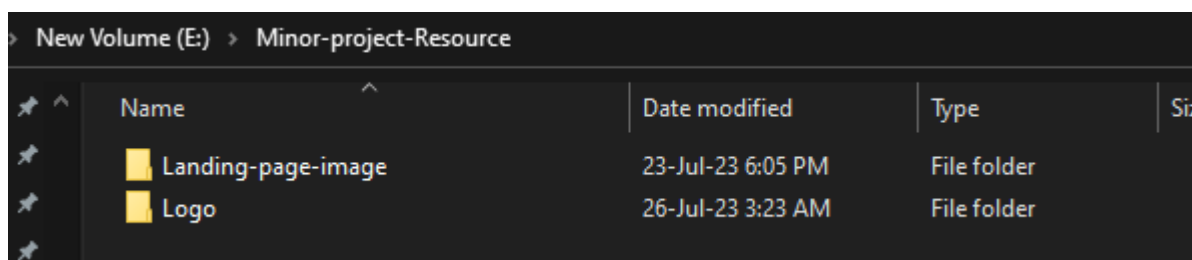
# Place widgets on the window
zip_button.pack(pady=10)
unzip_button.pack(pady=10)
status_label.pack()

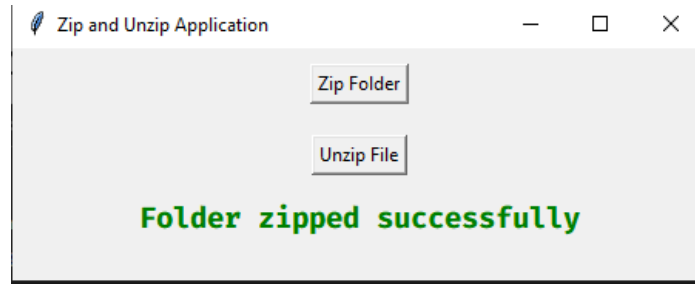
# important (to keep window on)
main_window.mainloop()
```

Output (GUI) :



Output (Zipping folder) :

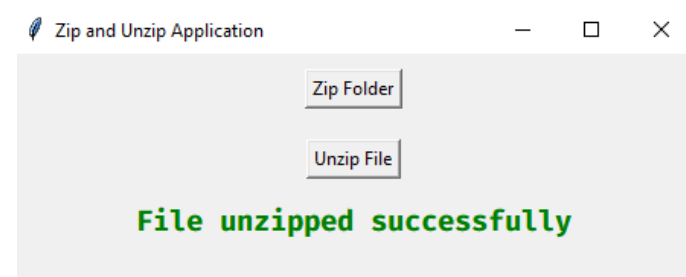




New Volume (E:) > Minor-project-Resource

Name	Date modified	Type	Size
Landing-page-image	23-Jul-23 6:05 PM	File folder	
Logo	26-Jul-23 3:23 AM	File folder	
Zip-file-image.zip	28-Sep-23 11:51 PM	WinRAR ZIP archive	7,863 KB

Output (UnZipping file) :



New Volume (E:) > Minor-project-Resource

Name	Date modified	Type	Size
Landing-page-image	23-Jul-23 6:05 PM	File folder	
Logo	26-Jul-23 3:23 AM	File folder	
1dfb2f1eb1.jpg	28-Sep-23 11:53 PM	JPG File	
6a4266f518.jpg	28-Sep-23 11:53 PM	JPG File	
7d02a6583b.jpg	28-Sep-23 11:53 PM	JPG File	
44cdc95f88.jpg	28-Sep-23 11:53 PM	JPG File	
2926c87f5b.jpg	28-Sep-23 11:53 PM	JPG File	
9204d1b269.jpg	28-Sep-23 11:53 PM	JPG File	
79842fe5bf.jpg	28-Sep-23 11:53 PM	JPG File	
981082d550.jpg	28-Sep-23 11:53 PM	JPG File	
dcc87664ed.jpg	28-Sep-23 11:53 PM	JPG File	
e4bcb9a26d.jpg	28-Sep-23 11:53 PM	JPG File	
ee16c72beb.jpg	28-Sep-23 11:53 PM	JPG File	
Landing Page.png	28-Sep-23 11:53 PM	PNG File	
page bg.png	28-Sep-23 11:53 PM	PNG File	
Untitled design (3).png	28-Sep-23 11:53 PM	PNG File	
Zip-file-image.zip	28-Sep-23 11:51 PM	WinRAR ZIP archive	

ASSIGNMENT-10

Q1) 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 :

```
----- Original tuple -----

(('a', 53), ('b', 37), ('c', 23), ('d', 1), ('e', 18))
(('d', 1), ('e', 18), ('c', 23), ('b', 37), ('a', 53))

----- Sorted by second element -----

[('d', 1), ('e', 18), ('c', 23), ('b', 37), ('a', 53)]
[('d', 1), ('e', 18), ('c', 23), ('b', 37), ('a', 53)]
```

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 :

```
----- Replace 10 at last -----
[(5, 2, 3), (4, 7, 6), (8, 9, 10)]
```

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 :

```

----- Original tuple -----

[(4, 5, 5, 7), (1, 3, 7, 4), (19, 4, 5, 3), (1, 2)]
[(19, 4, 5, 3), (4, 5, 5, 7), (1, 3, 7, 4), (1, 2)]

----- Sorted by number of element -----

[(1, 2), (4, 5, 5, 7), (1, 3, 7, 4), (19, 4, 5, 3)]
[(1, 2), (19, 4, 5, 3), (4, 5, 5, 7), (1, 3, 7, 4)]

```

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 :

```

"C:\Program Files\Python310\python.exe" "C:
----- Filtered List -----
[('D', 70), ('A', 67)]

```

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(","))

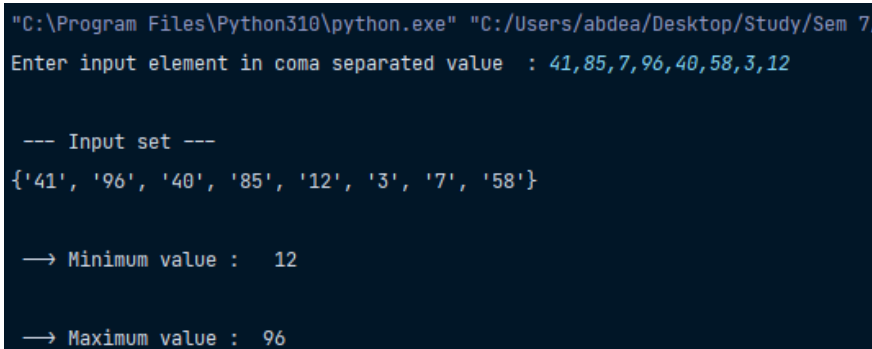
ip_sorted_set = sorted(ip_set)

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

print("\n --> Minimum value : ", ip_sorted_set[0])

print("\n --> Maximum value : ", ip_sorted_set[-1])
```

Output :



```
"C:\Program Files\Python310\python.exe" "C:/Users/abdea/Desktop/Study/Sem 7/
Enter input element in coma separated value : 41,85,7,96,40,58,3,12

--- Input set ---
{'41', '96', '40', '85', '12', '3', '7', '58'}

--> Minimum value : 12

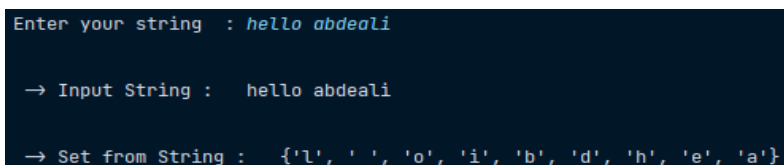
--> Maximum value : 96
```

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 :



```
Enter your string : hello abdeali

-> Input String : hello abdeali

-> Set from String : {'l', ' ', 'o', 'i', 'b', 'd', 'h', 'e', 'a'}
```

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

Code :

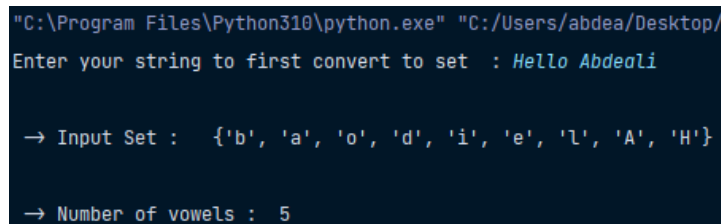
```
set_from_str = set(input("Enter your string to first convert to set :"))
print("\n -> Input Set : ", set_from_str)

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 :



```
"C:\Program Files\Python310\python.exe" "C:/Users/abdea/Desktop/"
Enter your string to first convert to set : Hello Abdeali

→ Input Set :  {'b', 'a', 'o', 'd', 'i', 'e', 'l', 'A', 'H'}

→ Number of vowels : 5
```

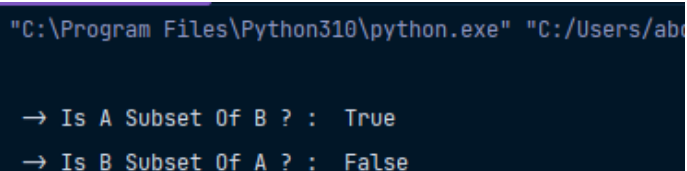
Q 8) Write a Python program to Check if a set a subset of another set.

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 :



```
"C:\Program Files\Python310\python.exe" "C:/Users/ab..."

→ Is A Subset Of B ? : True
→ Is B Subset Of A ? : False
```