Name: Mohammad Zaid Ansari Roll No: 03

[1]. Basic Programming Elements (Operators, if-else, looping,...)

Write Python programs to

Display odd and even numbers between a given range

```
first = int(input("Enter first number of the range: "))
last = int(input("Enter last number of the range: "))

if(first<=last):
    print('The even numbers are: ',end='')
    for i in range(first,last+1):
        if i%2==0:
            print(i, end=' ')
    print()
    print('The odd numbers are: ',end='')
    for i in range(first,last+1):
        if i%2!=0:
            print(i,end=' ')

else:
    print('The first number of range should be smaller than the last number ')
```

OUTPUT:

Enter first number of the range: 5

Enter last number of the range: 25

The even numbers are: 6 8 10 12 14 16 18 20 22 24

The odd numbers are: 5 7 9 11 13 15 17 19 21 23 25

Print an n digit number in reverse order

```
number = int(input('Enter a number: '))
result = 0
lastdigit = 0
while number!=0:
    lastdigit = number%10
    number = number//10
    result = result*10 + lastdigit

print(f"The reverse of the given number is {result}")
```

OUTPUT:

Enter a number: 12345

The reverse of the given number is 54321

Print Fibonacci series up to n members

```
n1 = 0
n2 = 1
n = 0
num = int(input("Enter a number to print fibonnaci series: "))


if num>1:
    print(n1,n2,end=' ')
    for i in range(2,num):
        n=n2+n1
        n1=n2
        n2=n
        print(n,end=' ')
elif num==1:
    print(n1)
else:
    print("Please enter a positive number")
```

OUTPUT:

```
Enter a number to print fibonnaci series: 10

0 1 1 2 3 5 8 13 21 34
```

Print all the numbers between 100 to 200 which are divisible by 4 and 7

```
print("Numbers divisible by 4 are: ",end='')
for i in range(100,201,4):
    print(i,end=' ')
print("\nNumbers divisible by 7 are: ",end='')
for i in range(100,201,7):
    print(i,end=' ')
```

OUTPUT:

```
Numbers divisible by 4 are: 100 104 108 112 116 120 124 128 132 136 140 144 148 152 156 160 164
168 172 176 180 184 188 192 196 200

Numbers divisible by 7 are: 100 107 114 121 128 135 142 149 156 163 170 177 184 191 198
```

[2] Python - LIST

Write Python program to....

Find number of list elements of different data types

```
my_list = ['Zaid',24,28.35,True,'A',4,25.3,False,'M','T',False]

i=s=f=c=b=0

l_int=[]
l_float=[]
l_bool=[]
l_str=[]
l_char=[]

for item in my_list:
    if type(item)==int:
        i+=1
        l_int.append(item)
    elif type(item)==float:
        f+=1
        l_float.append(item)
    elif type(item)==bool:
        b+=1
        l_bool.append(item)
    elif type(item)==str and len(item)>1:
        s+=1
        l_str.append(item)
    else:
        c+=1
        l_char.append(item)

print(f"The number of Integers in the list are {i} and elements are {l_int}")
print(f"The number of Float in the list are {f} and elements are {l_float}")
print(f"The number of Boolean in the list are {b} and elements are {l_bool}")
print(f"The number of Strings in the list are {s} and elements are {l_str}")
print(f"The number of Characters in the list are {c} and elements are
{l_char}")
```

OUTPUT:

```
The number of Integers in the list are 2 and elements are [24, 4]

The number of Float in the list are 2 and elements are [28.35, 25.3]

The number of Boolean in the list are 3 and elements are [True, False, False]

The number of Strings in the list are 1 and elements are ['Zaid']

The number of Characters in the list are 3 and elements are ['A', 'M', 'T']
```

Check the presence/absence of a given value in the list; then display total number of occurrences and index of the position of each occurrence.

```python
my_list=[23,15,49,45,13,4,8,13,8,4,8,4,4,49]
count=0
occur=[]
num = int(input("Enter a number to search: "))
for i in range(len(my_list)):
    if my_list[i]==num:
        count+=1
        occur.append(i)

print(f"The number {num} occured {count} times and the index of each
occurrence are {occur}")
```

OUTPUT:

```
Enter a number to search: 4

The number 4 occured 4 times and the index of each occurrence are [5, 9, 11, 12]
```

Sort list elements; press 1 for ascending order and, press 2 for descending order (choice based)

```python
my_list=[23,99,76,34,15,49,45,13,4,8,13,8,4,8,4,4,49]

temp = 0
asc_list = []
des_list = []

def ascending():
    for i in range(len(my_list)):
        for j in range(i+1,len(my_list)):
            if my_list[i]>my_list[j]:
                temp=my_list[i]
                my_list[i]=my_list[j]
                my_list[j]=temp

def descending():
    for i in range(len(my_list)):
        for j in range(i+1,len(my_list)):
            if my_list[i]<my_list[j]:
                temp=my_list[i]
                my_list[i]=my_list[j]
                my_list[j]=temp


ch=int(input("Press \n1 to sort the list in ascending order \n2 to sort the
list in descending order\n"))
```

```python
if ch == 1:
    ascending()
    print(my_list)
elif ch == 2:
    descending()
    print(my_list)
else:
    print("Wrong input")
```

OUTPUT:

```
Press

1 to sort the list in ascending order

2 to sort the list in descending order

1

[4, 4, 4, 4, 8, 8, 8, 13, 13, 15, 23, 34, 45, 49, 49, 76, 99]



Press

1 to sort the list in ascending order

2 to sort the list in descending order

2

[99, 76, 49, 49, 45, 34, 23, 15, 13, 13, 8, 8, 8, 4, 4, 4, 4]
```

[3] Python – User Defined Methods (UDM)

Write Python program using UDM to....

Find the factorial of a given number

Method

```python
def fact(n):
    result = 1
    for i in range(1,n+1):
        result=result*i
    return result
```

Main

```python
from fact_method import *

n = int(input("Enter a number to calculate a factorial: "))

print(f"The factorial of {n} is {fact(n)}")
```

OUTPUT:

```
Enter a number to calculate a factorial: 5

The factorial of 5 is 120
```

Calculate and return simple interest

Method

```python
def interest(amount,rate):
    result = amount * rate/100
    return result
```

Main

```python
from SI_method import *

amount = int(input("Enter the principal amount: "))
rate = int(input("Enter the rate of interest(In percent): "))

print(f"The interest amount for {amount} and {rate}% rate of interest is 
{interest(amount,rate)}")
```

OUTPUT:

```
Enter the principal amount: 50000

Enter the rate of interest(In percent): 8

The interest amount for 50000 and 8% rate of interest is 4000.0
```

Calculate and display gross salary of two categories of employees, permanent and temporary.

For permanent - gross salary = net salary + Bonus of 15% of net salary, no bonus for temporary employee

Method

```python
def Salary(net):
    gross = net + (net*15)/100
    return gross
```

Main

```python
from GS_method import *

net_salary = int(input("Enter the base salary: "))

print(f"The salary for permanent employee is {Salary(net_salary)}")
print(f"The salary for temporary employee is {net_salary}")
```

OUTPUT:

```
Enter the base salary: 5000000

The salary for permanent employee is 5750000.0

The salary for temporary employee is 5000000
```

[4] Python – Exception Handling

Write a program to demonstrate EH in python for handling two exceptions, namely, IndexError: list index out of range and ZeroDivisionError

```python
from logging import exception


i=["Zero","One","Two","Three"]

def index():
    ch = int(input("Enter the index of the element you want to view: "))
    try:
        print("The element is",i[ch])
    except Exception as e:
        print(e)


def zero():
    a = int(input("Enter the numerator: "))
    b = int(input("Enter the denominator: "))
    try:
        print("Division:",a/b)
    except Exception as e:
        print(e)

# Index out of range exception if input is > 3
index()
# Division by zero exception if denominator is zero
zero()
```

OUTPUT:

```
Enter the index of the element you want to view: 7

list index out of range

Enter the numerator: 25

Enter the denominator: 0

division by zero
```

Write a program to RAISE and handle user defined exception.

```python
class AgeLowError(Exception):
    def __str__(self):
        return "Age is less than 18 you cannot drive"
class AgeHighError(Exception):
    def __str__(self):
        return "Age is greater than 65 you cannot drive"

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

def AgeVerification(age):
    try:
        if age>18 and age<=65:
            print("You are Eligibel to drive")
        elif age<18:
            raise AgeLowError
        elif age>65:
            raise AgeHighError
    except AgeLowError as L:
        print(L)
    except AgeHighError as H:
        print(H)


AgeVerification(age)
```

OUTPUT:

```
Enter your age: 15

Age is less than 18 you cannot drive



Enter your age: 25

You are Eligibel to drive



Enter your age: 70

Age is greater than 65 you cannot drive
```

[5] Python – OOPs

Write an object oriented program to demonstrate use of default and parameterized constructor.

```python
class Students:
    def __init__(self):
        self.name=""
        self.branch=""

    def getData(self):
        self.name=input("Enter student name: ")
        self.branch=input("Enter student branch: ")

    def __str__(self):
        return f"The name of the students is {self.name} and the branch is 
{self.branch}"

stu = int(input("Enter number of student data you want to insert: "))


l = []
for i in range(1,stu+1):
    s = Students()
    print(f"Student[{i}]:")
    s.getData()
    l.append(s)


for i in range(stu):
    print()
    # l[i].display()
    print(l[i])
```

OUTPUT:

```
Enter number of student data you want to insert: 2

Student[1]:

Enter student name: Ansari

Enter student branch: IT

Student[2]:

Enter student name: Zaid

Enter student branch: CS



The name of the students is Ansari and the branch is IT



The name of the students is Zaid and the branch is CS
```

Create two objects distance1 and distance2 for a class called DISTANCE, Accept values of distance1 and distance2 from the user in terms of feet and inches and finally calculate and display the sum of two distances in feet and inches.

```python
class Distance:
    def __init__(self):
        self.d1_feet = 0
        self.d1_inch = 0
        self.d2_feet = 0
        self.d2_inch = 0

    def getData(self):
        print("Enter 1st Distance: ")
        self.d1_feet = int(input("Enter feet: "))
        self.d1_inch = int(input("Enter inch: "))
        print("\nEnter 2nd Distance: ")
        self.d2_feet = int(input("Enter feet: "))
        self.d2_inch = int(input("Enter inch: "))

    def add(self):
        result_feet = self.d1_feet + self.d2_feet
        result_inch = self.d2_inch + self.d1_inch

        while result_inch >=12:
            result_inch-=12
            result_feet+=1

        print(f"\nSum of distance is {result_feet} feet and {result_inch}
inch")

d = Distance()
d.getData()
d.add()
```

OUTPUT:

```
Enter 1st Distance:

Enter feet: 10

Enter inch: 4


Enter 2nd Distance:

Enter feet: 27

Enter inch: 16


Sum of distance is 38 feet and 8 inch
```

Write a program to implement multilevel inheritance (use of super keyword and method overriding is must)

```python
class Box:
    def __init__(self,length,width,height):
        self.length = length
        self.width = width
        self.height = height

class Cuboid(Box):
    def __init__(self,length,width,height):
        super().__init__(length,width,height)

    def area(self):
        return "Area of rect = ",(self.width*self.length + 
self.width*self.height + self.length*self.height)

class Color(Cuboid):
    def __init__(self,length,width,height,color):
        super().__init__(length,width,height)
        self.color = color

    def area(self):
        a = 2*(self.width*self.length + self.width*self.height + 
self.length*self.height)
        return f"Area of cuboid is {a} and color is {self.color}"


length = int(input("Enter the length of cuboid: "))
width = int(input("Enter the width of cuboid: "))
height = int(input("Enter the height of cuboid: "))
color = input("Enter the color of cuboid: ")

rec = Color(length,width,height,color)

print(rec.area())
```

OUTPUT:

```
Enter the length of cuboid: 5

Enter the width of cuboid: 5

Enter the height of cuboid: 5

Enter the color of cuboid: Blue

Area of cuboid is 150 and color is Blue
```

[6] Python – GUI and Database Connection

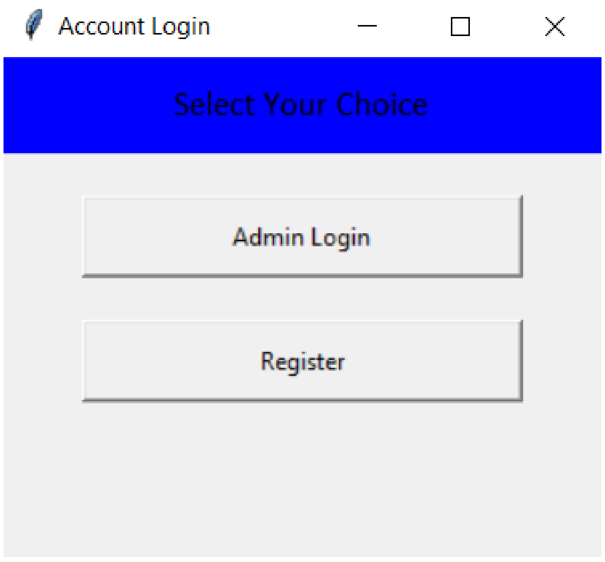
Design two interfaces, one for user registration and the other one for displaying all registered users from the database

```python
from tkinter import *
import mysql.connector

mydb = mysql.connector.connect(
    host="localhost",
    user="root",
    password="",
    database="Users"
)

mycursor = mydb.cursor()
# Designing window for registration

def register():
    global register_screen
    register_screen = Toplevel(main_screen)
    register_screen.title("Register")
    register_screen.geometry("300x250")

    global email
    global username
    global password
    global email_entry
    global username_entry
    global password_entry
    email = StringVar()
    username = StringVar()
    password = StringVar()

    Label(register_screen, text="Please enter details below",
bg="blue").pack()
    Label(register_screen, text="").pack()

    email_lable = Label(register_screen, text="Email")
    email_lable.pack()
    email_entry = Entry(register_screen, textvariable=email)
    email_entry.pack()

    username_lable = Label(register_screen, text="Username")
    username_lable.pack()
    username_entry = Entry(register_screen, textvariable=username)
    username_entry.pack()

    password_lable = Label(register_screen, text="Password")
    password_lable.pack()
    password_entry = Entry(register_screen, textvariable=password, show='*')
    password_entry.pack()
    Label(register_screen, text="").pack()
    Button(register_screen, text="Register", width=10, height=1, bg="blue",
command = register_user).pack()



def login():
    global login_screen
    login_screen = Toplevel(main_screen)
    login_screen.title("Login")
    login_screen.geometry("300x250")
    Label(login_screen, text="Admin login").pack()
    Label(login_screen, text="").pack()

    global username_verify
    global password_verify

    username_verify = StringVar()
    password_verify = StringVar()

    global username_login_entry
    global password_login_entry

    Label(login_screen, text="Username").pack()
    username_login_entry = Entry(login_screen, textvariable=username_verify)
    username_login_entry.pack()
    Label(login_screen, text="").pack()
    Label(login_screen, text="Password").pack()
    password_login_entry = Entry(login_screen, textvariable=password_verify,
show= '*')
    password_login_entry.pack()
    Label(login_screen, text="").pack()
    Button(login_screen, text="Login", width=10, height=1, command =
login_verify).pack()

# Implementing event on register button

def register_user():

    email_info = email.get()
    username_info = username.get()
    password_info = password.get()

    query = "INSERT INTO Register (email, username, password) VALUES (%s, %s,
%s)"
    val = (email_info, username_info, password_info)
```

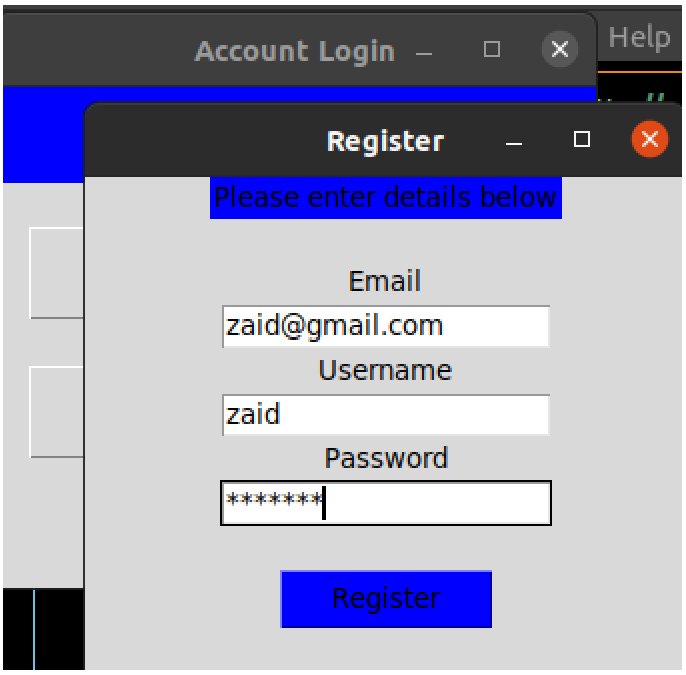
```python
    mycursor.execute(query, val)

    mydb.commit()

    email_entry.delete(0, END)
    username_entry.delete(0, END)
    password_entry.delete(0, END)

    Label(register_screen, text="Registration Success", fg="green",
font=("calibri", 11)).pack()

# Implementing event on login button

def login_verify():
    username1 = username_verify.get()
    password1 = password_verify.get()
    username_login_entry.delete(0, END)
    password_login_entry.delete(0, END)

    try:
        query = "SELECT * FROM Admin"
        mycursor.execute(query)

        result = mycursor.fetchall()
        for x in result:
            if username1 and password1 in x:
                home_page()
            elif username1 not in x:
                user_not_found()
            elif password1 not in x:
                password_not_recognised()
    except Exception as e:
        print("Something went wrong")
        print(e)

# Designing popup for login success

def login_sucess():
    global login_success_screen
    login_success_screen = Toplevel(login_screen)
    login_success_screen.title("Success")
    login_success_screen.geometry("150x100")
    Label(login_success_screen, text="Login Success").pack()
    Button(login_success_screen, text="OK",
command=delete_login_success).pack()

# Designing popup for login invalid password
```

```
def password_not_recognised():
    global password_not_recog_screen
    password_not_recog_screen = Toplevel(login_screen)
    password_not_recog_screen.title("Success")
    password_not_recog_screen.geometry("150x100")
    Label(password_not_recog_screen, text="Invalid Password ").pack()
    Button(password_not_recog_screen, text="OK",
command=delete_password_not_recognised).pack()

# Designing popup for user not found

def user_not_found():
    global user_not_found_screen
    user_not_found_screen = Toplevel(login_screen)
    user_not_found_screen.title("Success")
    user_not_found_screen.geometry("150x100")
    Label(user_not_found_screen, text="User Not Found").pack()
    Button(user_not_found_screen, text="OK",
command=delete_user_not_found_screen).pack()

# Deleting popups

def delete_login_success():
    login_success_screen.destroy()


def delete_password_not_recognised():
    password_not_recog_screen.destroy()


def delete_user_not_found_screen():
    user_not_found_screen.destroy()

# home page shows user details can only be accessed using admin login

def home_page():
    global home_screen
    main_screen.destroy()
    home_screen = Tk()

    home_screen.geometry("600x300")

    sql = "SELECT * FROM Register"
    mycursor.execute(sql)
    result =  mycursor.fetchall()
    i = 0
    items = ['Email', 'Username', 'Password']
    for x in range(1):
        for j in range(len(items)):
            e = Entry(home_screen, width=25, fg='red')
            e.grid(row=i, column=j)
            e.insert(END, items[j])
        i=i+1
    for user in result:
        for j in range(len(user)):
            e = Entry(home_screen, width=25, fg='blue')
            e.grid(row=i, column=j)
            e.insert(END, user[j])
        i=i+1

    home_screen.mainloop()

# Designing Main(first) window

def main_account_screen():
    global main_screen
    main_screen = Tk()
    main_screen.geometry("300x250")
    main_screen.title("Account Login")
    Label(text="Select Your Choice", bg="blue", width="300", height="2",
font=("Calibri", 13)).pack()
    Label(text="").pack()
    Button(text="Admin Login", height="2", width="30", command = login).pack()
    Label(text="").pack()
    Button(text="Register", height="2", width="30", command=register).pack()

    main_screen.mainloop()


main_account_screen()
```

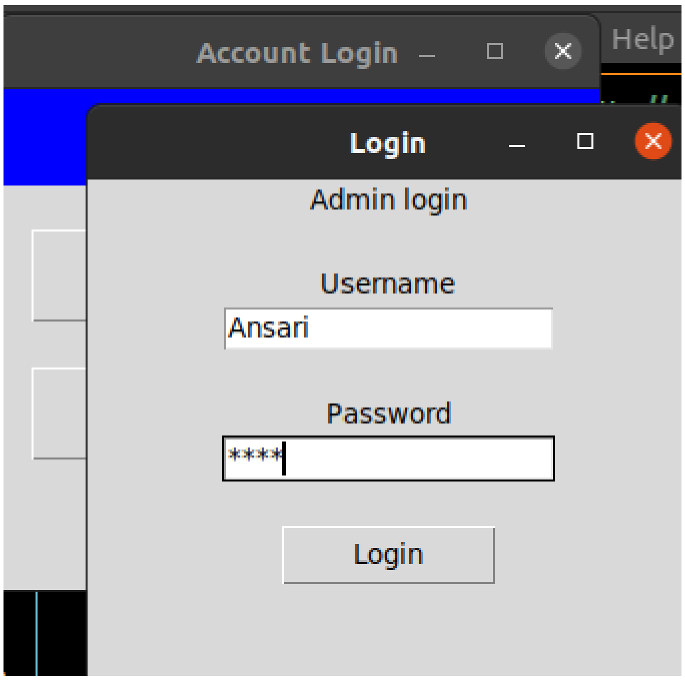
OUTPUT:



--



Email
zaid@gmail.com
Username
zaid
Password
****



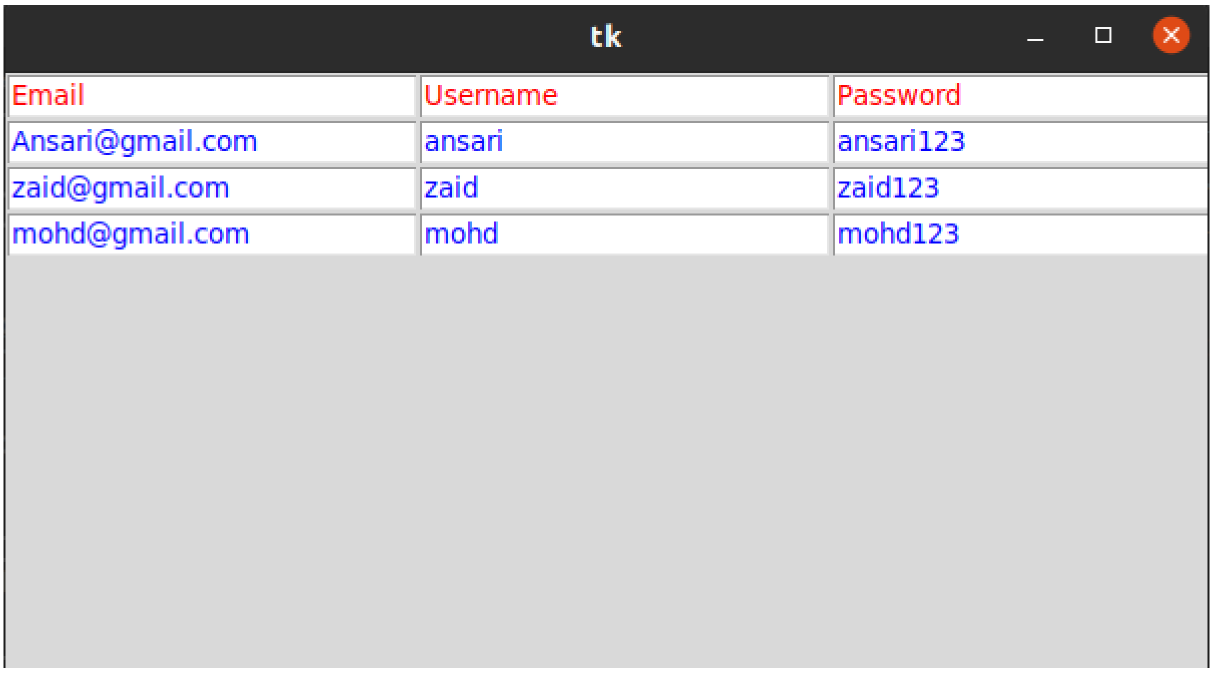
Admin login

Username

Ansari

Password

****



Email	Username	Password
Ansari@gmail.com	ansari	ansari123
zaid@gmail.com	zaid	zaid123
mohd@gmail.com	mohd	mohd123

[7] Python – Exploratory Data Analysis Select an appropriate dataset and perform EDA to answer the following questions. Create a Jupyter Notebook and start with installing required python libraries and loading the dataset into the data frame.

Number of independent variables

In any given data set, labeling variables as dependent or independent is arbitrary -- there is no fundamental reason that one column should be independent and another should be dependent. That said, typically it's conventional to say that "causes" are independent variables and "effects" are dependent variables.

In our data set, we have 5 independent variable

1. *Brand*
2. *Processor*
3. *RAM*
4. *Hard Disk Capacity*
5. *Rating*

What is the dependent variable in the selected dataset

In any given data set, labeling variables as dependent or independent is arbitrary -- there is no fundamental reason that one column should be independent and another should be dependent. That said, typically it's conventional to say that "causes" are independent variables and "effects" are dependent variables.

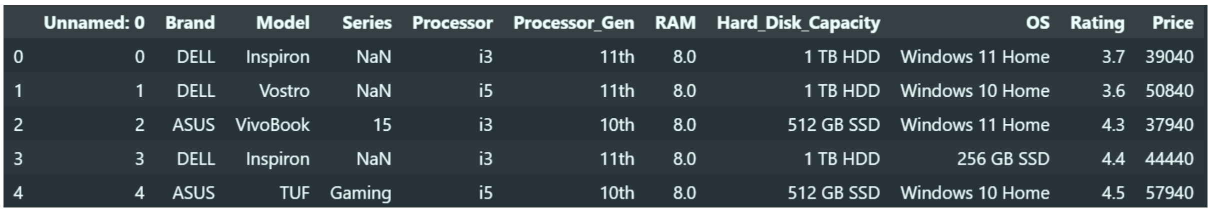
In our data set, we have 4 dependent variable

1. *Model*
2. *Series*
3. *Processor Generation*
4. *Price*

Display top 5 rows from the selected dataset

```
import pandas as pd

data = pd.read_csv("Laptop.csv")
data.head()
```

OUTPUT:



```
   Unnamed: 0  Brand     Model  Series  Processor  Processor_Gen  RAM  Hard_Disk_Capacity                OS  Rating  Price
0           0   DELL  Inspiron     NaN         i3           11th  8.0             1 TB HDD  Windows 11 Home     3.7  39040
1           1   DELL    Vostro     NaN         i5           11th  8.0             1 TB HDD  Windows 10 Home     3.6  50840
2           2   ASUS  VivoBook      15         i3           10th  8.0           512 GB SSD  Windows 11 Home     4.3  37940
3           3   DELL  Inspiron     NaN         i3           11th  8.0             1 TB HDD       256 GB SSD     4.4  44440
4           4   ASUS       TUF  Gaming         i5           10th  8.0           512 GB SSD  Windows 10 Home     4.5  57940
```

Name of the independent variable having minimum average value

```
data.mean()
```

OUTPUT

```
RAM 8.622951 Rating 4.194309
```

```
Rating has the minmun average value
```

Name of the independent variable having high standard deviation

```
data.std()
```

OUTPUT

```
RAM 3.253296 Rating 0.364960

RAM has the highest standard deviation
```

Find the total count of missing values in each column(independent variables). Visualize all missing values, and identify the independent variable having maximum number of missing values?

```
Brand             0
Model             14
Series            50
Processor         7
Processor_Gen     7
RAM               8
Hard_Disk_Capacity 8
OS                8
Rating            7
Price             0
```

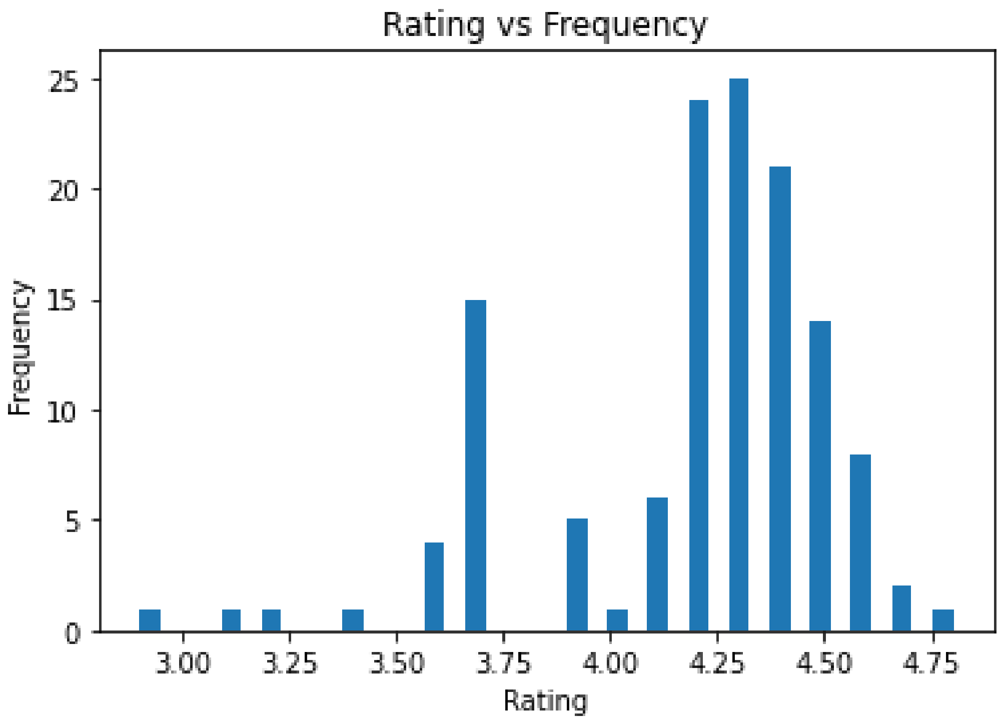
Choose one numeric independent variable having missing values and replace the missing values with the average value of the column.

```
data["Rating"].fillna(data["Rating"].mean(), inplace = True)
data.isnull().sum()
```

```
Brand              0                   RAM               8
Model              14                  Hard_Disk_Capacity 8
Series             50                  OS                8
Processor          7                   Rating            0
Processor_Gen      7                   Price             0
```

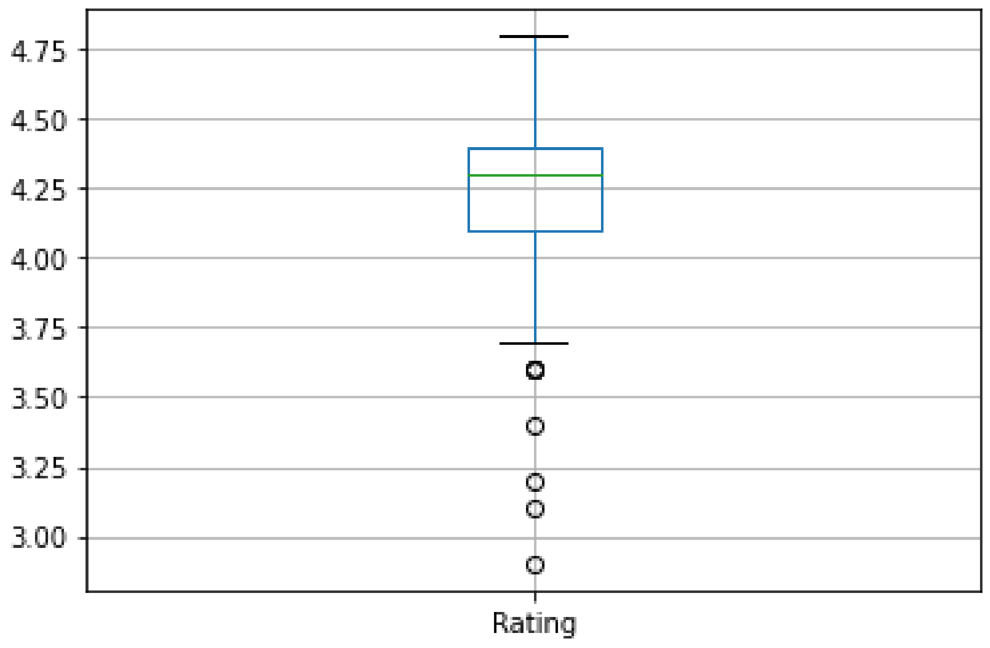
Choose one independent variable and show frequency distribution using histogram

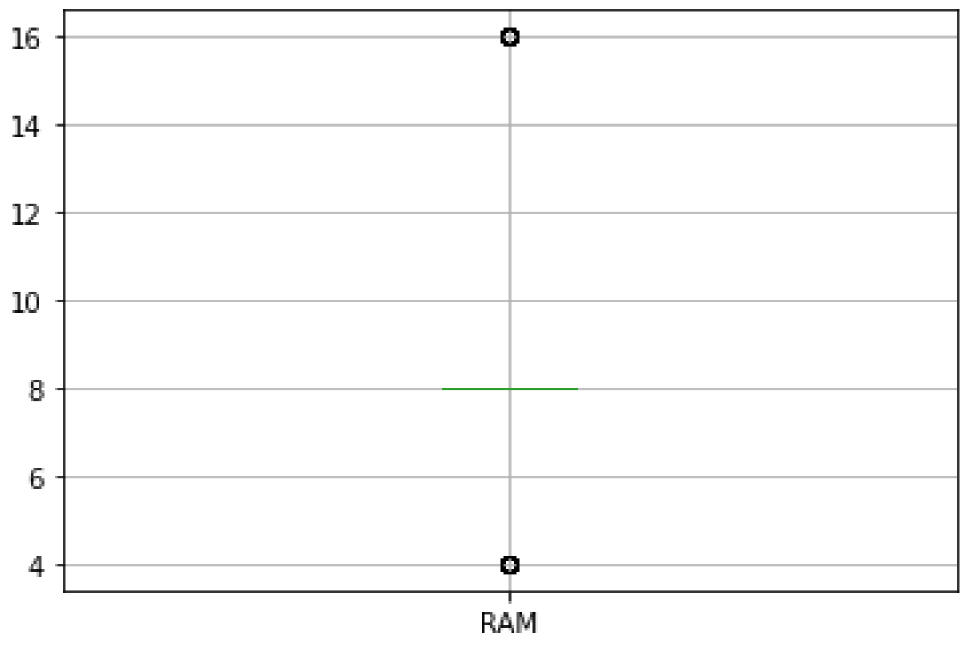
```
import matplotlib.pyplot as plt

plt.hist(data["Rating"],bins=40)
plt.title("Rating vs Frequency")
plt.xlabel("Rating")
plt.ylabel("Frequency")

plt.show()
```



Name of the independent variable having outliers? Use Box-plot to visualize.

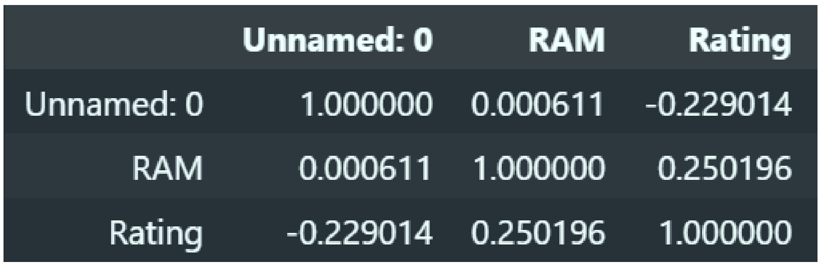
Rating and RAM





Display Correlation Matrix and find two pairs of independent variables, one having strong positive correlation and the other pair having strong negative correlation

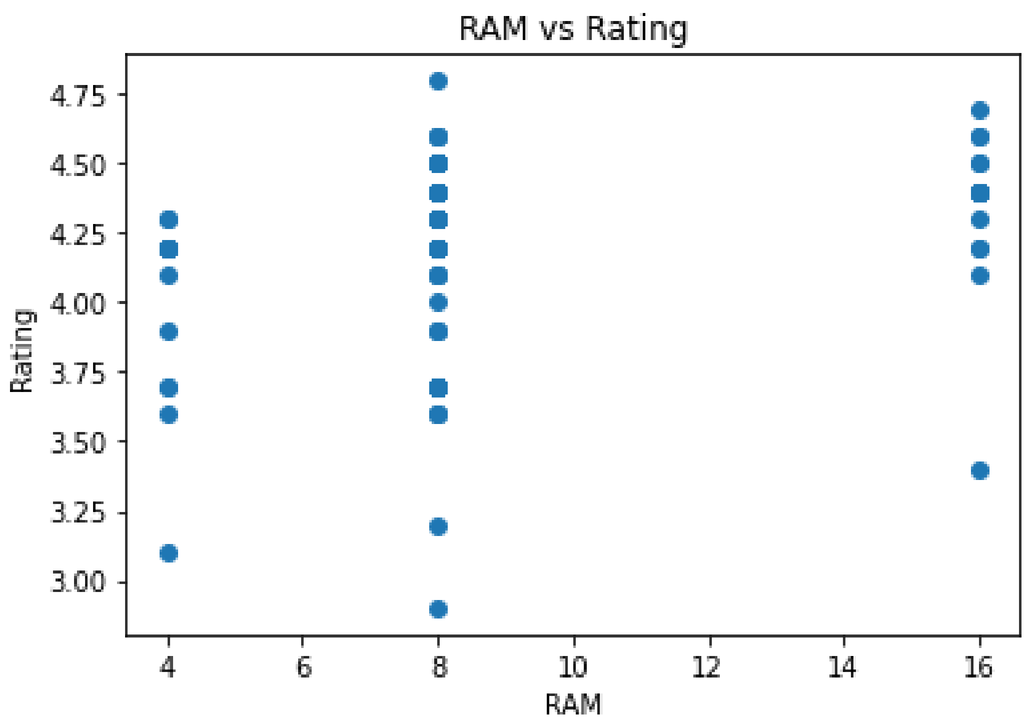
```
data.corr()
```



	Unnamed: 0	RAM	Rating
Unnamed: 0	1.000000	0.000611	-0.229014
RAM	0.000611	1.000000	0.250196
Rating	-0.229014	0.250196	1.000000

This data set has only one correlation between RAM and Rating which is weak positive

Use scatter plots to visualize the correlation between independent and dependent variables (use same two pairs identified in previous questions)


RAM vs Rating
Rating
RAM