NAME Soham Desai
BATCH-B
ROLL NO-11
XIE ID - 202003021

Department of Information Technology

Sem: IV Python Lab 2021-22

Assignment

Q.1) Write a python program to input a string, remove punctuation from a string and then sort words in alphabetic order. (LO1)

Code:

```
punctuations = '''!()-[]{};:'"\,<>./?@#$%^&*_~'''

my_str = input("Enter a string: ")

no_punct = ""

for char in my_str:

   if char not in punctuations:

       no_punct = no_punct + char

print(no_punct)

words = [word.lower() for word in no_punct.split()]

words.sort()

print("The sorted words are:")

for word in words:

   print(word)
```

Output:

```
Enter a string: Hello!!! My Name#& is Soham,, Desai..
Hello My Name is Soham Desai
The sorted words are:
desai
hello
is
my
name
soham
```

Q.2) Write the following programs (LO2)

i) Write a python program to count tuples occurrences in given list of tuples and then remove duplicate tuples from list of tuples

Code:

```
import collections

x = [[('Mon', 'Wed')], [('Mon')], [('Tue')],[('Mon', 'Wed')] ]

a = collections.defaultdict(int)

for elem in x:

  a[elem[0]] += 1

print("Count of tuples present in the list:\n",a)

def removeDuplicates(Tuple):

   return [t for t in (set(tuple(i) for i in Tuple))]

print(removeDuplicates(x))
```

Output:

```
Count of tuples present in the list:
 defaultdict(<class 'int'>, {('Mon', 'Wed'): 2, 'Mon': 1, 'Tue': 1})
[('Tue',), (('Mon', 'Wed'),), ('Mon',)]
```

ii) Write a python program to create a sub-dictionary containing all keys from dictionary list

Code:

```
from itertools import chain

a = [{'soham' : 3, 'is' : 7},

                {'soham' : 3, 'is' : 1, 'best' : 5},

                {'soham' : 8}]

print("The original list is : " + str(a))

all_keys = set(chain.from_iterable(a))

res = [dict((key, sub.get(key, None)) for key in all_keys) for sub in a]

print("Reformed dictionaries list : " + str(res))
```

Output:

```
The original list is : [{'soham': 3, 'is': 7}, {'soham': 3, 'is': 1, 'best': 5}, {'soham': 8}]
Reformed dictionaries list : [{'is': 7, 'soham': 3, 'best': None}, {'is': 1, 'soham': 3, 'best': 5}, {'is': None, 'soham': 8, 'best': None}]
```

Q.3) Create a Vehicle class with max_speed and mileage instance attributes. Create a Bus and Taxi classes that inherit the Vehicle class. Give the capacity argument of Bus. The seating_capacity() for bus and Taxi a default value of 50 and 3 respectively. The default fare charge of any vehicle is seating capacity * 100 per 5km. If Vehicle is Bus instance, we need to add an extra 10% on full fare as a maintenance charge. So total fare for bus instance will become the final amount = total fare + 10% of the total fare. Calculate total fare charges spent by group for picnic if both taxi and bus is used for travelling 100km distance one way. (LO3)

Code:

```
class Vehicle:
    def __init__(self, name, mileage, capacity):
        self.name = name
        self.mileage = mileage
        self.capacity = capacity
    def show(self):
        print("Name:", self.name, "\nMileage:", self.mileage, "\nCapacity:", self.capacity)
class Bus(Vehicle):
    def fare(self, distance):
        def_fare = 0
        def_fare = self.capacity * 20 * distance
        print("Fare:", def_fare)
        total_bus_fare = def_fare + 0.1 * def_fare
        print("Total fare:", total_bus_fare)
class Taxi(Vehicle):
    def fare(self, distance):
        def_fare = 0
        def_fare = self.capacity * 10 * distance
        print("Fare:", def_fare)
        total_bus_fare = def_fare + 0.05 * def_fare
        print("Total fare:", total_bus_fare)
School_bus = Bus("School Volvo", 12, 50)
School_bus.show()
```

```
School_bus.fare(100)

taxi = Taxi("Taxi", 10, 10)

taxi.show()

taxi.fare(100)
```

Output:

```
Name: School Volvo
Mileage: 12
Capacity: 50
Fare: 100000
Total fare: 110000.0
Name: Taxi
Mileage: 10
Capacity: 10
Fare: 10000
Total fare: 10500.0
```

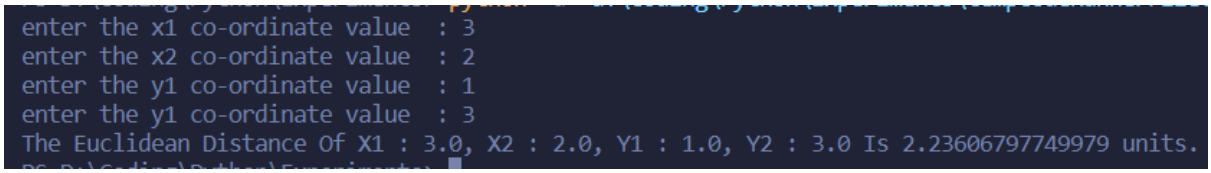
Q.4) Create module for performing mathematical function and import it to calculate Euclidean distance. Show exception handling to handle the runtime mistake done by user. (LO4)

Code:

```
import math

class MyMathLibrary:

    @staticmethod

    def calculateEuclideanDistance(x1, x2, y1, y2):

        xMinus = x2 - x1

        yMinus = y2 - y1

        internalCalc = xMinus**2 + yMinus**2

        euclidDistance = math.sqrt(internalCalc)

        return euclidDistance
```

```
import mathmodule

try:
    x1 = float(input("enter the x1 co-ordinate value  : "))
    x2 = float(input("enter the x2 co-ordinate value  : "))
    y1 = float(input("enter the y1 co-ordinate value  : "))
    y2 = float(input("enter the y1 co-ordinate value  : "))
    myResult = mathmodule.MyMathLibrary.calculateEuclideanDistance(x1, x2, y1, y2)
    print(f"The Euclidean Distance Of X1 : {x1}, X2 : {x2}, Y1 : {y1}, Y2 : {y2} Is {myResult} units.")
except:
    print("kindly enter valid co-ordinates value.")
```

Output:



Q.5) Develop GUI Application for E-commerce application use (LO5)

1) file, pickle, dictionary to show add, delete, update operations.

Code:

```
import sys, pickle

from PyQt5.QtWidgets import QApplication, QWidget, QGridLayout, QLabel, QLineEdit, QPushButton, QTableWidget,  QTableWidgetItem, QMessageBox

class dictionary(dict):
    def init(self):
        self = dict()
    def add(self, key, value):
        self[key] = value
    def delete(self, key):
        self.pop(key)
    def main(self):
        s = dictionary()
        def add():
```

```
        name = nameLine.text()
        price = priceLine.text()
        s.add(name, price)
        print(s)
        row = table.rowCount()
        table.setRowCount(row + 1)
        namecell = QTableWidgetItem(name)
        pricecell = QTableWidgetItem(price)
        table.setItem(row, 0, namecell)
        table.setItem(row, 1, pricecell)
    def delete(self):
        selected = table.selectedItems()
        name = selected[0].text()
        selectedIndex = table.selectedIndexes()
        rowNo = selectedIndex[0].row()
        table.removeRow(rowNo)
        s.delete(name)
        print(s)
    def save(self):
        file = open("shopping.pickle", "wb")
        pickle.dump(s, file)
        file.close()
        print("Data saved!")
    def upload(self):
        file = open("shopping.pickle", "rb")
        temp = pickle.load(file)
        for name, price in temp.items():
            print(name, price)
        s.add(name, price)
        row = table.rowCount()
```

```
        table.setRowCount(row + 1)
        namecell = QTableWidgetItem(name)
        pricecell = QTableWidgetItem(price)
        table.setItem(row, 0, namecell)
        table.setItem(row, 1, pricecell)
        file.close()
    def bill():
        file = open("shopping.pickle", "rb")
        temp = pickle.load(file)
        sum = 0
        for Name, price in temp.items():
            print(Name, price)
            sum += float(price)
        msg = QMessageBox()
        msg.setWindowTitle("Cash Invoice")
        msg.setText("Your bill amount is " + str(sum))
        x = msg.exec_()
        file.close()
    app = QApplication(sys.argv)
    w = QWidget()
    layout = QGridLayout()
    nameLabel = QLabel()
    nameLabel.setText("Name of Product :")
    nameLine = QLineEdit()
    priceLabel = QLabel()
    priceLabel.setText("Price :")
    priceLine = QLineEdit()
    emptyLabel = QLabel()
    emptyLabel.setText("***Product Names And Prices****")
    addButton = QPushButton()
```

```
addButton.setText("Add Record")
delButton = QPushButton()
delButton.setText("Delete Record")
saveButton = QPushButton()
saveButton.setText("Save Record")
uploadButton = QPushButton()
uploadButton.setText("Upload Record")
billButton = QPushButton()
billButton.setText("View bill")
billButton.clicked.connect()
table = QTableWidget()
table.setColumnCount(2)
table.setHorizontalHeaderLabels(["Name of Product", "Price"])
table.resizeColumnToContents(0)
table.resizeColumnToContents(1)
table.setWordWrap(True)
addButton.clicked.connect(add)
delButton.clicked.connect(delete)
saveButton.clicked.connect(save)
uploadButton.clicked.connect(upload)
w.resize(550, 350)
w.setWindowTitle("Shopping List")
layout.addWidget(nameLabel, 1, 1)
layout.addWidget(nameLine, 1, 2)
layout.addWidget(priceLabel, 2, 1)
layout.addWidget(priceLine, 2, 2)
layout.addWidget(addButton, 3, 1)
layout.addWidget(delButton, 3, 2)
layout.addWidget(saveButton, 3, 3)
layout.addWidget(uploadButton, 3, 4)
```

```
        layout.addWidget(billButton, 3, 5)
        layout.addWidget(emptyLabel, 4, 2)
        layout.addWidget(table, 5, 2)
        w.setLayout(layout)
        w.show()
        sys.exit(app.exec_())
if __name__ == " main ":
    d = dictionary()
    d.main()
```

2) sqlite3 dictionary to show add, delete, update operations.

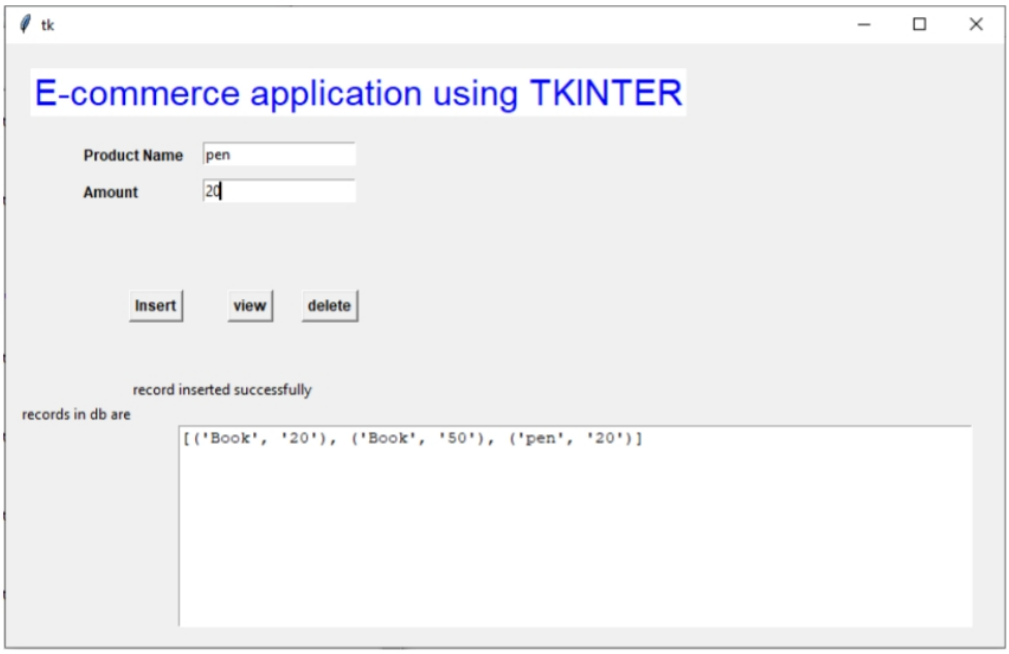
Code:

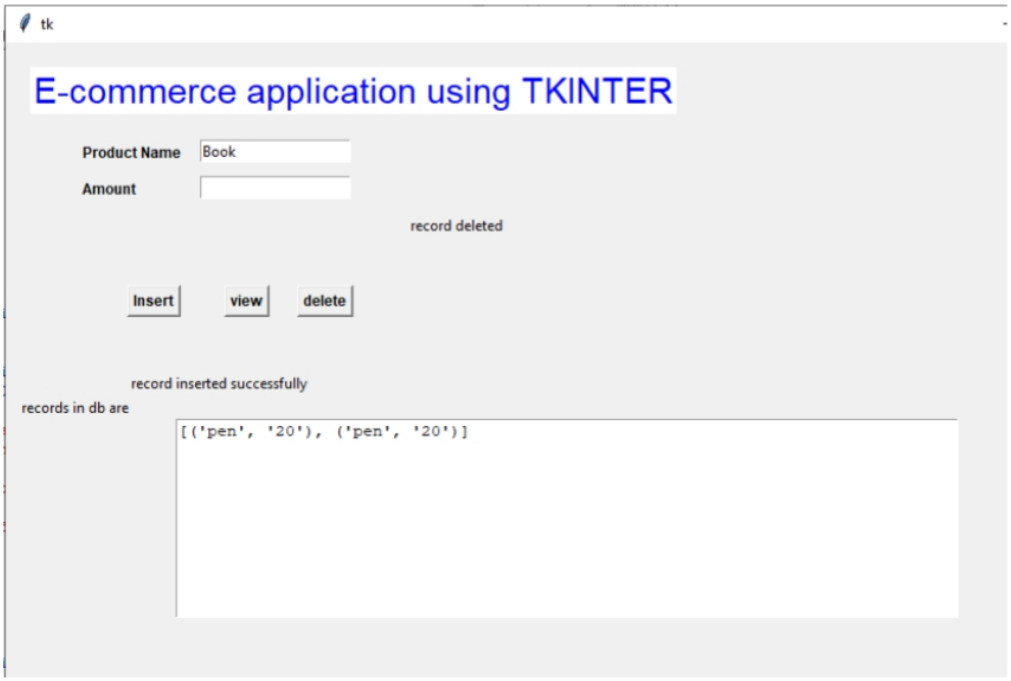
```
from tkinter import *
import sqlite3
top = Tk()
top.geometry("750x700")
conn = sqlite3.connect('products.db')
print("Database established succesfully!")
cur = conn.cursor()
cur.execute( """CREATE TABLE IF NOT EXISTS PRODUCTS(NAME TEXT,PRICE
TEXT,QUANTITY TEXT)""")  # write SQL queries in () print("Products table created
succesfully")
conn.commit()
conn.close()
s1 = e1.get()
s2 = e2.get()
s3 = e3.get()
print(s1, s2, s3)
conn = sqlite3.connect('products.db')
print("Attempting to open the database")
cur = conn.cursor()
```

```
val = (s1, s2, s3)
cur.execute("'INSERT INTO PRODUCTS(NAME,PRICE,QUANTITY) VALUES (?,?,?)"', val)
conn.commit()
print("Values fed into database: ", val)
l_add = Label(top, text="Record inserted successfully")
l_add.place(x=200, y=300)
conn.close()
def view():
    conn = sqlite3.connect('products.db')
    print("Attempting to open the database")
    cur = conn.cursor()
cur.execute("'SELECT * FROM PRODUCTS"')
records = cur.fetchall()  # record = str(records)
print("Records in the database are: ", str(records))
l_view = Label(top, text="Records in the Table products are: ")
l_view.place(x=200, y=380)
T = Text(top, height=10, width=50)
T.place(x=200, y=440)
T.insert(INSERT, str(records))
conn.commit()
conn.close()
def delete():
    conn = sqlite3.connect('products.db')
    print("Attempting to open the database")
cur = conn.cursor()
n1 = e4.get()
print(n1)
cur.execute("'DELETE FROM PRODUCTS WHERE NAME = ?"', (n1,))
l_del = Label(top, text="Record deleted")
l_del.place(x=450, y=150)
```

```
conn.commit()

conn.close()

l_main = Label(top, text="Available Products", font=("Helvetica 25 bold "), fg="violet",
bg="orange")

l_main.place(x=20, y=20)

l_name = Label(top, text="Name of product", font=('Helvetica 12 bold'))

l_name.place(x=20, y=80)

e1 = Entry(top)

e1.place(x=200, y=80)

l_price = Label(top, text="Price", font=('Helvetica 12 bold'))

l_price.place(x=20, y=120)

e2 = Entry(top)

e2.place(x=200, y=120)

l_quantity = Label(top, text="Quantity", font=('Helvetica 12 bold'))

l_quantity.place(x=20, y=160)

e3 = Entry(top)

e3.place(x=200, y=160)

l_n = Label(top, text="Product Name to be deleted", font=('Helvetica 12 bold'))

l_n.place(x=350, y=80)

e4 = Entry(top)

e4.place(x=600, y=80)

b1 = Button(top, text="ADD", font=('Helvetica 12 bold'), command=insert)

b1.place(x=20, y=240)

b2 = Button(top, text="VIEW", font=('Helvetica 12 bold'), command=view)

b2.place(x=100, y=240)

top.mainloop()
```

Output:


tk
E-commerce application using TKINTER
Product Name
pen
Amount
20
Insert
view
delete
record inserted successfully
records in db are
[('Book', '20'), ('Book', '50'), ('pen', '20')]


tk
E-commerce application using TKINTER
Product Name
Book
Amount
record deleted
Insert
view
delete
record inserted successfully
records in db are
[('pen', '20'), ('pen', '20')]

NAME Soham Desai
BATCH-B
ROLL NO-11
XIE ID - 202003021

Q.6) Prepare a graph showing attendance analysis of SE IT students (attendance sheet is uploaded on Google classroom) (LO6)

Code:

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
df = pd.read_csv('atten.csv')
rollNo = df['Roll_no'].values
a = np.arange(len(rollNo))
w = 0.5
plt.bar(a, df['Sub1'].values, width=w, color='r',label='Sub1')
plt.bar(a+w, df['Sub2'].values, width=w, color='g',label='Sub2')
plt.bar(a+2*w, df['Sub3'].values, width=w, color='b',label='Sub3')
plt.bar(a+3*w, df['Sub4'].values, width=w, color='y',label='Sub4')
plt.bar(a+4*w, df['Sub5'].values, width=w, color='c',label='Sub5')
plt.xticks(a+w, rollNo)
plt.legend()
plt.show()
```