Nirvan Sharma

Class: XII-A

Board Roll No. \_\_\_\_\_\_\_\_

DPS Indirapuram

Computer Science Practical File

Session: 2021-2022

**INDEX**

|  |  |  |
| --- | --- | --- |
| **TERM 1** | | |
|  | **PROGRAMS BASED ON CONTROL STRUCTURES** | **SIG.** |
| 1. | Write a program to implement a card game called “DRAGONS AND WIZARDS”. Make two teams Dragons and Wizards. The rules of the game are as follows   * If the card drawn is a diamond or a club, Team Dragons gets a point. * If the card drawn is a heart which is a number, team Wizard gets a point * If the card drawn is a heart that is not a number, team dragon gets a point * For any other card, team wizard gets a point * The team with highest point is the winner.   Input: shape, value  Process: Increment in respective team scores by one based on the outcome of the card drawn, as defined in the rules.  Output: Winning Team |  |
| 2. | Write a Python program that requests five integer values from the user. It then prints the maximum and minimum values entered. If the user enters the values 3, 2, 5, 0, and 1, the program would indicate that 5 is the maximum and 0 is the minimum. Your program should handle ties properly; for example, if the user enters 2, 4, 2, 3, and 3, the program should report 2 as the minimum and 4 as maximum. |  |
| 3. | Write a program that simulates a traffic light. The program should consist of the following:   * A user defined function trafficLight() that accepts input from the user and displays an error message if user enters anything other than RED, YELLOW or GREEN. Function LIGHT() is called and following is displayed depending upon return value from LIGHT()   + “STOP, your life is precious” if the value returned by LIGHT() is 0.   + “Please WAIT, untill the light is green”, if the value returned by LIGHT() is 1   + “GO! Thankyou for being patient”, if the value returned by LIGHT() is 2. * A user defined function LIGHT() that accepts a string as input and returns 0 when the input is RED, 1 when the input is YELLOW and 2 when the input is GREEN. The input should be passed as an argument.   Display “SPEED THRILLS BUT KILLS” after the function trafficLight() is executed. |  |
| 4. | To secure your account, whether it be an email, online bank account or any other account, it is important that we use authentication. Use your programming expertise to create a program using user defined function named login that accepts userid and password as parameters (login(uid,pwd)) that displays a message “account blocked” in case of three wrong attempts. The login is successful if the user enters user ID as “ADMIN” and password as “St0rE@1”. On successful login, display a message, “Login Successful” |  |
|  | **PROGRAMS BASED ON STRINGS/LISTS/TUPLES/DICTIONARY** |  |
| 5. | Write a program to convert a given number into equivalent Roman number (store its value as a string). You can use following guidelines to develop solution for it:   * From the given number, pick successive digits, using %10 and /10 to gather the digits from right to left. * The rules for Roman Numerals involve using four pairs of symbols for ones and fives, tens and fifties, hundreds and five hundreds. An additional symbol for thousands covers all the relevant bases. * When a number is followed by the same or smaller number, it means addition. “II” is two 1’s =2. “VI” is 5+1=6. * When one number is followed by a large number, it means subtraction. “IX” is 1 before 10=9. “IIX” isn’t allowed, this would be “VIII”. For numbers from 1 to 9, the symbols are “I” and “V”, and the coding works like this. “I”, “II”, “III”, “IV”, “V”, “VI”, “VII”, “VIII”, “IX”. * The same rules work for numbers from 10 to 99, using “X” and “L”. For numbers from 100 to 900, using the symbols “C” and “D”. For numbers from 1000 to 4000, using “M”.   Here are some examples. 1994=MCMXCIV, 1956=MCMLVI, 3888=MMMDCCCLXXXVIII |  |
| 6. | ROT13 is a weak form of encryption that involves “rotating” each letter in a word by 13 places. To rotate a letter means to shift it through the alphabet, wrapping around to the beginning if necessary, so ’A’ shifted by 3 is ’D’ and ’Z’ shifted by 1 is ’A’.  Write a function called rotate\_word that takes a string and an integer as parameters, and that returns a new string that contains the letters from the original string “rotated” by the given amount.  For example, “cheer” rotated by 7 is “jolly” and “melon” rotated by -10 is “cubed”.  You might want to use the built-in functions ord, which converts a character to a numeric code, and chr, which converts numeric codes to characters. |  |
| 7. | This question is based on a Puzzler that was broadcast on the radio program Car Talk. Give me a word with three consecutive double letters. I’ll give you a couple of words that almost qualify, but don’t. For example, the word committee, c-o-m-m-i-t-t-e-e. It would be great except for the ‘i’ that sneaks in there. Or Mississippi: M-i-s-s-i-s-s-ip- p-i. If you could take out those i’s it would work. But there is a word that has three consecutive pairs of letters and to the best of my knowledge this may be the only word. Of course there are probably 500 more but I can only think of one. What is the word? Write a program to find it/them out of the given string. |  |
| 8. | Two words are anagrams if you can rearrange the letters from one to spell the other.  Write a function called is\_anagram that takes two strings and returns True if they are anagrams. |  |
| 9. | Complete the following function that determines if the number of even and odd values in an integer list is the same. The function would return true if the list contains 5, 1, 0, 2 (two evens and two odds), but it would return false for the list containing 5, 1, 0, 2, 11 (too many odds). The function should return true if the list is empty, since an empty list contains the same number of evens and odds (0 for both). The function does not affect the contents of the list.  **def balanced(a):**  **# Add your code...** |  |
| 10. | Write a program that takes a list of words and creates a dictionary with frequency (number of occurences) of word as key and list of words for that frequency as value.  For example, if list is given as [‘the’,’of’, ‘an’, ‘is’, ‘an’, ‘the’]  Then dictionary should be {2:[‘the’,’an’],1:[‘of’,’is’]} |  |
| 11. | Write a program to find the frequency of values in the given dictionary. For example, if the dictionary is given as D={‘P1’:60,’P2’:30,’P3’:50,’P4’:60,’P5’:30,’P6’:10}  Then output should be {10:1,30:2,50:1,60:2} |  |
|  | **PROGRAMS BASED ON TEXT FILES** |  |
| 12 | Read a text file line by line and display each word separated by a #. |  |
| 13 | Read a text file and display the number of vowels/consonants/uppercase/lowercase characters in the file. |  |
| 14. | Write a menu driven program to   * to display last three characters of all the lines available in the text File ‘abcd.txt’ * to display the content of a text file ‘abcd.txt’ file in uppercase. * to find and display the count of all the uppercase characters available in text file ‘abcd.txt’ to count and display total number of vowels available in a text File ‘abcd.txt’ |  |
| 15. | Write a menu driven program to   * to read the content of a text file ‘abcd.txt’ file and copy the same content in another file ‘Copy.txt’ file * to read a word from keyboard and find out the frequency of this word in a text file ‘abcd.txt’ * To read a text file ‘abcd.txt’ and copy all those lines that start with ‘the’ into another text file ‘copy.txt’ * to read the text file ‘abcd.txt’ and replace the word ‘This’ with ‘That’ in this file. * to read the text file ‘abcd.txt’ and display the content after removing the word ‘This’ from the file |  |
|  | **PROGRAMS BASED ON BINARY FILES** |  |
| 16. | Following is the structure of each record in a data file named “PRODUCT.DAT”.  **{"prod\_code":value, "prod\_desc":value, "stock":value}**  The values for prod\_code and prod\_desc are strings, and the value for stock is an integer.  Write a menu driven program using functions   * to enter records * to display all records * to update the file with a new value of stock. The stock and the product\_code, whose stock is to be updated, are to be input during the execution of the function. |  |
| 17. | Given a binary file “STUQ2.DAT”, containing records of the following type:  **[S\_Admno, S\_Name, Percentage]**  Where these three values are:  **S\_Admno – Admission Number of student (string)**  **S\_Name – Name of student (string)**  **Percentage – Marks percentage of student (float)**  Write a menu driven program using functions   * to enter records * to display all records   to read contents of the file “STUDENT.DAT” and display the details of those students whose percentage is above 75. |  |
| 18. | Assuming the tuple Vehicle as follows:  **( vehicletype, no\_of\_wheels)**  where vehicletype is a string and no\_of\_wheels is an integer.  Write a menu driven program to perform the following in binary file **vehicle.dat** using functions   * to enter records * to display all records * to count and display the number of records present in the file. |  |
|  | **PROGRAMS BASED ON CSV FILES** |  |
| 19. | Create a file PRODUCT.CSV. Sample data of the file is as follows:    Write a menu driven program using functions   * to add sample data to the file * to display all the records * to copy/transfer only those records from the file PRODUCT.CSV to another file “PRO1.CSV” whose quantity is more than 150. Also include the first row with headings. * To display the total cost of all the products of the file PRODUCT.CSV * To search and display the record of that product from the file product.csv which has maximum cost. |  |
| 20. | Create a file Tour.csv having headings as follows TID,DESTINATION,DAYS,FARE. Sample data of file is as follows:    Write a menu driven program using functions   * to add sample data to the file * to display all the records * to read the file tour.csv and display the records where fare is between 500 and 750. If no such record is found in the file then display an appropriate message on the screen. |  |
| **TERM 2** | | |
|  | **PROGRAMS BASED ON STACK** |  |
| 21. | Write a function in python, **MakePush(Package)** and **MakePop(Package)** to add a new Package and delete a Package from a List of Package Description, considering them to act as push and pop operations of the Stack data structure. Implement the complete menu driven program |  |
| 22. | Write the functions in Python push (stk, item ) and pop(stk) to check whether the stack is empty, to add a new item, to delete an item and display the stack respectively. Implement the menu driven program. |  |
| 23. | Write a program to perform push operation on a stack to push all prime numbers from a list entered by a user. |  |
|  | **SQL QUERIES BASED ON 2 TABLES** |  |
| 24. | Write SQL queries for (i) to (v), which are based on the tables CUSTOMERS and PURCHASES given below: |  |
|  | (i) To display details of all CUSTOMERS whose CITIES are neither Delhi nor Mumbai.  (ii) To display the CNAME and CITIES of all CUSTOMERS in ascending order of their CNAME.  (iii) To display the number of CUSTOMERS along with their respective CITIES in each of the CITIES.  (iv) To display details of all PURCHASES whose Quantity is more than 15.  v) To display customer name, quantity and purchase date for only those purchases whose quantity is either 10 or 20.  vi) To display the maximum purchase date |  |
|  | **PROGRAMS BASED ON PYTHON MySQL CONNECTIVITY** |  |
| 25. | **Consider the following tables product and client.** |  |
|  | Develop a complete menu driven application using python mySQL connectivity based on following parameters. Create functions wherever required |  |
| a) | Database name:STORE |  |
| b) | Table names: PRODUCT and CLIENT (as shown above) |  |
| c) | Primary keys: pid, cid |  |
| d) | Write suitable code in python to create database, tables and to insert the records. |  |
| e) | Create a function to display the client name and the product purchased by the client in descending order of client names |  |
| f) | Create a function to increase the price of all the products by 5%. Now display all the records of product table. |  |
| g) | Create a function to remove the records of clients who are from Bangalore. Now display all the records of Client table. |  |
| h) | Create a function to display number of clients from each city. |  |
| i) | Create a function to increase the width of column city to 50. |  |
|  |  |  |
|  | …………………………………………………………………………… |  |

Answers

Q1. Write a program to implement a card game called “DRAGONS AND WIZARDS”. Make two teams Dragons and Wizards. The rules of the game are as follows

* If the card drawn is a diamond or a club, Team Dragons gets a point.
* If the card drawn is a heart which is a number, team Wizard gets a point
* If the card drawn is a heart that is not a number, team dragon gets a point
* For any other card, team wizard gets a point
* The team with highest point is the winner.

Input: shape, value

Process: Increment in respective team scores by one based on the outcome of the card drawn, as defined in the rules.

Output: Winning Team

Code:

# Dragons and Wizards

print("Welcome to Dragons and Wizards.")

# Number of times the card is drawn

n = int(input("How many times would you like to draw the cards? "))

# Declaring scores

d\_point = 0

w\_point = 0

# Drawing Cards

for i in range(n):

    shape = str(input("\nEnter Shape: (Hearts, Spades, Diamonds, Clubs)\n"))

    # Hearts and its number

    if shape == 'Hearts':

        heartval = 0

        inttry = 0

        while inttry == 0:

            enterint = input("Do you want to draw a Heart with a number? (y/n) ")

            if enterint == 'y':

                heartval = int(input("Enter number for Hearts: "))

                break

            elif enterint == 'n':

                break

            else:

                print('\nPlease enter a valid answer!')

    if shape == 'Hearts' and heartval > 0:

        w\_point += 1

        print('\nOne point to the Wizards!', d\_point, '-', w\_point)

    elif shape == 'Spades':

        w\_point += 1

        print('\nOne point to the Wizards!', d\_point, '-', w\_point)

    elif shape == 'Hearts':

        d\_point += 1

        print('\nOne point to the Dragons!', d\_point, '-', w\_point)

    elif shape == 'Diamonds' or 'Clubs':

        d\_point += 1

        print('\nOne point to the Dragons!', d\_point, '-', w\_point)

# Determining the winning team

if d\_point > w\_point :

    print("\nTeam Dragons win!", d\_point, "-", w\_point)

elif d\_point == w\_point:

    print("\nIt's a Tie!", w\_point, "-", d\_point)

else:

    print("\nTeam Wizards win!", w\_point, "-", d\_point)

Output:

(next page)

Text

Description automatically generated

Q2. Write a Python program that requests five integer values from the user. It then prints the maximum and minimum values entered. If the user enters the values 3, 2, 5, 0, and 1, the program would indicate that 5 is the maximum and 0 is the minimum. Your program should handle ties properly; for example, if the user enters 2, 4, 2, 3, and 3, the program should report 2 as the minimum and 4 as maximum.

Code:

# Maximum and Minimum values from a list of 5 integers

# Inputting Numbers

num\_list = []

i = 0

while i < 5:

    try:

        num\_inp = int(input(str(i+1)+". Enter an integer: "))

        num\_list.append(num\_inp)

        i += 1

    except:

        print('Please enter an integer.\n')

        continue

# Finding the Max and Min values from the list

max\_num = max(num\_list)

min\_num = min(num\_list)

# Result

print("\nThe greatest number is", str(max\_num), "and the smallest number is", str(min\_num)+'.')

Output:

Graphical user interface, text, application, email

Description automatically generated

Q3. Write a program that simulates a traffic light. The program should consist of the following:

* A user defined function trafficLight() that accepts input from the user and displays an error message if user enters anything other than RED, YELLOW or GREEN. Function LIGHT() is called and following is displayed depending upon return value from LIGHT()
  + “STOP, your life is precious” if the value returned by LIGHT() is 0.
  + “Please WAIT, until the light is green”, if the value returned by LIGHT() is 1
  + “GO! Thank you for being patient”, if the value returned by LIGHT() is 2.
* A user defined function LIGHT() that accepts a string as input and returns 0 when the input is RED, 1 when the input is YELLOW and 2 when the input is GREEN. The input should be passed as an argument.

Display “SPEED THRILLS BUT KILLS” after the function trafficLight() is executed.

Code:

# Program to simulate a Traffic Light

def LIGHT():

    # Declaring list of colours

    cols = ['RED', 'YELLOW', 'GREEN']

    # Letting the user select the colour of the light

    colchosen = False

    while colchosen == False:

        light = str(input("Enter the color of the traffic light: (RED, YELLOW, GREEN)\n"))

        if light.upper() not in cols:

            print(("Please choose a colour from the options given.\n"))

            continue

        else:

            colchosen = True

            break

    # Determining value from user's selection

    global val

    if light.upper() == 'RED':

        val = 0

    elif light.upper() == 'YELLOW':

        val = 1

    elif light.upper() == 'GREEN':

        val = 2

def trafficLight():

    # Calling LIGHT() function

    LIGHT()

    # Printing message based on value

    if val == 0:

        print("\nSTOP! Your life is precious!")

    elif val == 1:

        print("\nPlease WAIT until the light is GREEN!")

    elif val == 2:

        print("\nGO! Thank you for being patient!")

    # Printing message after trafficLight() has finished executing

    print("SPEED THRILLS BUT KILLS\n")

# Calling trafficLight() to begin program

trafficLight()

Output:

Graphical user interface, text, application, email

Description automatically generated

Q4. To secure your account, whether it be an email, online bank account or any other account, it is important that we use authentication. Use your programming expertise to create a program using user defined function named login that accepts userid and password as parameters (login(uid,pwd)) that displays a message “account blocked” in case of three wrong attempts. The login is successful if the user enters user ID as “ADMIN” and password as “St0rE@1”. On successful login, display a message, “Login Successful”

Code:

# Authentication Program

# Initialize number of attempts

tries = 0

# Checking credentials

def login(uid, pwd):

    if uid == "ADMIN" and pwd == "St0rE@1" and tries < 3:

        print("Login Successful\n")

        return True

    elif tries < 2:

        print("Try again\n")

        return False

    else:

        print('Account Blocked\n')

        return False

print("Please enter your username and password(You have 3 attempts).\n")

while tries < 3:

    if login(str(input("Username: ")), str(input("Password: "))) is False:

        tries += 1

        continue

    else:

        break

Output:

(next page)

Graphical user interface, text, application, email

Description automatically generated

Q5. Write a program to convert a given number into equivalent Roman number (store its value as a string). You can use following guidelines to develop solution for it:

* From the given number, pick successive digits, using %10 and /10 to gather the digits from right to left.
* The rules for Roman Numerals involve using four pairs of symbols for ones and fives, tens and fifties, hundreds and five hundreds. An additional symbol for thousands covers all the relevant bases.
* When a number is followed by the same or smaller number, it means addition. “II” is two 1’s =2. “VI” is 5+1=6.
* When one number is followed by a large number, it means subtraction. “IX” is 1 before 10=9. “IIX” isn’t allowed, this would be “VIII”. For numbers from 1 to 9, the symbols are “I” and “V”, and the coding works like this. “I”, “II”, “III”, “IV”, “V”, “VI”, “VII”, “VIII”, “IX”.
* The same rules work for numbers from 10 to 99, using “X” and “L”. For numbers from 100 to 900, using the symbols “C” and “D”. For numbers from 1000 to 4000, using “M”.

Here are some examples. 1994=MCMXCIV, 1956=MCMLVI, 3888=MMMDCCCLXXXVIII

Code:

# Integer to Roman

# Declare fundamental roman numbers

int\_list = (1000, 900, 500, 400, 100, 90, 50, 40, 10, 9, 5, 4, 1)

roman\_list = ('M', 'CM', 'D', 'CD', 'C', 'XC', 'L', 'XL', 'X', 'IX', 'V', 'IV', 'I')

# Take input from user

inp\_num = int(input("Please enter the number that you want to convert to Roman Numerals: "))

romantext = ''

# Conversion to Roman

for i in range(len(int\_list)):

    quot = int((inp\_num)/(int\_list[i]))

    romantext += str((roman\_list[i])\*(quot))

    inp\_num -= (int\_list[i])\*(quot)

print(romantext)

Output:

(next page)

Text, timeline

Description automatically generated

Q6. ROT13 is a weak form of encryption that involves “rotating” each letter in a word by 13 places. To rotate a letter means to shift it through the alphabet, wrapping around to the beginning if necessary, so ’A’ shifted by 3 is ’D’ and ’Z’ shifted by 1 is ’A’.

Write a function called rotate\_word that takes a string and an integer as parameters, and that returns a new string that contains the letters from the original string “rotated” by the given amount.

For example, “cheer” rotated by 7 is “jolly” and “melon” rotated by -10 is “cubed”.

You might want to use the built-in functions ord, which converts a character to a numeric code, and chr, which converts numeric codes to characters.

Code:

# String Rotation

def rotate\_word():

    alphabegin = ord("a")

    alphaend = ord("z")

    inp\_str = str(input("Enter the word you want to encrypt: "))

    num = int(input("Number of rotations: "))

    rot\_num = num%26

    new\_str = ''

    for i in inp\_str.lower():

        new\_num = ord(i) + rot\_num

        while new\_num > alphaend:

            new\_num -= 26

        while new\_num < alphabegin:

            new\_num += 26

        new\_str += chr(new\_num)

    print(new\_str)

rotate\_word()

Output:Graphical user interface, text, application, email

Description automatically generated

Q7. This question is based on a Puzzler that was broadcast on the radio program Car Talk. Give me a word with three consecutive double letters. I’ll give you a couple of words that almost qualify, but don’t. For example, the word committee, c-o-m-m-i-t-t-e-e. It would be great except for the ‘i’ that sneaks in there. Or Mississippi: M-i-s-s-i-s-s-ip- p-i. If you could take out those i’s it would work. But there is a word that has three consecutive pairs of letters and to the best of my knowledge this may be the only word. Of course there are probably 500 more but I can only think of one. What is the word? Write a program to find it/them out of the given string.

Code:

# Program to find out words with three consecutive pairs of letters

user\_input = str(input("Enter your sentence: "))

word\_list = [(i.strip(',.')).lower() for i in user\_input.split()]

c\_count = 0

c\_list = []

for word in word\_list:

    for i in range(len(word)):

        if len(word) > 6 and i != len(word)-5 and word[i] == word[i+1] and word[i+2] == word[i+3] and word[i+4] == word[i+5]:

            c\_list.append(word)

        elif len(word) > 6 and i == len(word)-5:

            break

        elif len(word) < 6:

            break

        else:

            continue

if len(c\_list) == 0:

    print('There are no words with three consecutive pairs of letters in the sentence.')

elif len(c\_list) == 1:

    print('The word with three consecutive letters is: '+str(c\_list)[1:-1]+'.')

else:

    print('The words with three consecutive letters are: '+str(c\_list)[1:-1]+'.')

Output:

Text

Description automatically generated

Q8. Two words are anagrams if you can rearrange the letters from one to spell the other.

Write a function called is\_anagram that takes two strings and returns True if they are anagrams.

Code:

# Anagram Checker

def is\_anagram():

    wrd1 = str(input("Enter the first word: "))

    wrd2 = str(input("Enter the second word: "))

    if sorted(wrd1.lower()) == sorted(wrd2.lower()):

        print("The words are anagrams of each other.")

    else:

        print("They are not anagrams.")

is\_anagram()

Output:

Graphical user interface, text, application, email

Description automatically generated

Q9. Complete the following function that determines if the number of even and odd values in an integer list is the same. The function would return true if the list contains 5, 1, 0, 2 (two evens and two odds), but it would return false for the list containing 5, 1, 0, 2, 11 (too many odds). The function should return true if the list is empty, since an empty list contains the same number of evens and odds (0 for both). The function does not affect the contents of the list.

**def balanced(a):**

**# Add your code...**

Code:

# Program to determine whether the number of odd and even digits in a list are equal

def balanced(a):

    if a == '':

        print("True")

    else:

        int\_list = []

        for i in str(a):

            int\_list.append(int(i))

        even\_count = 0

        odd\_count = 0

        for i in int\_list:

            if i % 2 == 0:

                even\_count += 1

            else:

                odd\_count += 1

        print(odd\_count == even\_count)

balanced(input("Enter the integers you want to compare: "))

Output:

(next page)

Graphical user interface, text

Description automatically generated

Q10. Write a program that takes a list of words and creates a dictionary with frequency (number of occurences) of word as key and list of words for that frequency as value.

For example, if list is given as [‘the’,’of’, ‘an’, ‘is’, ‘an’, ‘the’]

Then dictionary should be {2:[‘the’,’an’],1:[‘of’,’is’]}

Code:

# Program to find the frequency of words appearing in a string

words = str(input("Enter a string: "))

words\_list = words.split()

word\_freq\_dict = {}

for i in words\_list:

    if i in word\_freq\_dict:

        word\_freq\_dict[i] += 1

    else:

        word\_freq\_dict[i] = 1

print(word\_freq\_dict)

final\_dict = {}

for key, value in word\_freq\_dict.items():

    if value not in final\_dict:

        final\_dict[value] = [key]

    else:

        final\_dict[value].append(key)

print(final\_dict)

Output:

Graphical user interface, text, application

Description automatically generated

Q11. Write a program to find the frequency of values in the given dictionary. For example, if the dictionary is given as D={‘P1’:60,’P2’:30,’P3’:50,’P4’:60,’P5’:30,’P6’:10}

Then output should be {10:1,30:2,50:1,60:2}

Code:

# Program to find the frequency of  values in a given dictionary

D = {'P1':60,'P2':30,'P3':50,'P4':60,'P5':30,'P6':10}

newdict = {}

for i in D.values():

    if i in newdict:

        newdict[i] += 1

    else:

        newdict[i] = 1

print(newdict)

Output:

Graphical user interface, text, application

Description automatically generated

Q12. Read a text file line by line and display each word separated by a #.

Code:

# Program to read a text file line by line and display each word separated by a '#'

with open('Files/text.txt', 'r') as f:

    for line in f:

        print(line.replace(' ','#'))

Output:

Graphical user interface, text

Description automatically generated

Q13. Read a text file and display the number of vowels/consonants/uppercase/lowercase characters in the file.

Code:

# Program to read a text file and display the number of vowels/consonants/uppercase/lowercase characters in the file

f = open('Files/text.txt', 'r')

volist = []

colist = []

uplist = []

lolist = []

for i in f.read():

    if i in 'aeiouAEIOU':

        volist.append(i)

        if i.isupper():

            uplist.append(i)

        else:

            lolist.append(i)

    elif i not in 'aeiouAEIOU':

        colist.append(i)

        if i.isupper():

            uplist.append(i)

        else:

            lolist.append(i)

    else:

        continue

f.seek(0)

print(f.read())

print('\nNumber of vowels is',len(volist),

        '\nNumber of consonants is',len(colist),

        '\nNumber of uppercase characters is',len(uplist),

        '\nNumber of lowercase characters is',len(lolist))

f.close()

Output:Graphical user interface, text, application, email

Description automatically generated

Q14. Write a menu driven program to

* to display last three characters of all the lines available in the text File ‘abcd.txt’
* to display the content of a text file ‘abcd.txt’ file in uppercase.
* to find and display the count of all the uppercase characters available in text file ‘abcd.txt’
* to count and display total number of vowels available in a text File ‘abcd.txt’

Code:

# Text File Menu-driven Program 1

while True:

    f = open('Files/abcd.txt', 'r')

    print('\n[1] Display last three characters of each line\n[2] Display text in uppercase\n[3] Display uppercase characters and their number\n[4] Display number of vowels\n[5] Quit')

    try:

        useropt = int(input('\nSelect [1], [2], [3], [4], or [5]: '))

    except:

        print('\nPlease enter a number.')

        continue

    if useropt == 1:

        for i in f.readlines():

            print(i[-4:-1])

    elif useropt == 2:

        print('\n'+(f.read()).upper())

    elif useropt == 3:

        unum = 0

        ulist = []

        for i in f.read():

            if i.isupper():

                unum += 1

                ulist.append(i)

            else:

                continue

        print('Uppercase characters are:',\*ulist,'\nNumber of uppercase characters is',unum)

    elif useropt == 4:

        vnum = 0

        for i in f.read():

            if i in 'AEIOUaeiou':

                vnum += 1

            else:

                continue

        print('\nNumber of vowels is',vnum)

    elif useropt == 5:

        f.close()

        break

    else:

        print('\nPlease choose a number between 1 and 4')

        continue

Output:

First 3 options:

Text

Description automatically generated

Last 2 options:

Text

Description automatically generated

Q15. Write a menu driven program to

* to read the content of a text file ‘abcd.txt’ file and copy the same content in another file ‘Copy.txt’ file
* to read a word from keyboard and find out the frequency of this word in a text file ‘abcd.txt’
* to read a text file ‘abcd.txt’ and copy all those lines that start with ‘the’ into another text file ‘copy.txt’
* to read the text file ‘abcd.txt’ and replace the word ‘This’ with ‘That’ in this file.
* to read the text file ‘abcd.txt’ and display the content after removing the word ‘This’ from the file

Code:

# Text File Menu-driven Program 2

while True:

    f1 = open('Files/abcd.txt', 'r+')

    print('\n[1] Copy content to Copy.txt\n[2] Find frequency of a word\n[3] Copy lines starting with "the" to copy2.txt\n[4] Replace "This" with "That"\n[5] Remove "This" and display file\n[6] Quit')

    try:

        useropt = int(input("\nSelect [1], [2], [3], [4], [5], or [6]: "))

    except:

        print('\nPlease enter a number.')

        continue

    if useropt == 1:

        with open('Files/Copy.txt', 'w+') as f2:

            f2.write(f1.read())

            f2.seek(0)

            print("\nText in",f2.name+':\n\n'+f2.read())

        continue

    elif useropt == 2:

        wordopt = input('\nPlease type the word you want to find the frequency of: ')

        freq = 0

        for i in (f1.read()).split():

            if wordopt == i.strip(',.\n'):

                freq += 1

            else:

                continue

        print('\nThe word appears',freq,'times.')

        continue

    elif useropt == 3:

        thelist = []

        for i in f1.readlines():

            if (i.split())[0] == 'the':

                thelist.append(i)

            else:

                continue

        with open('Files/copy2.txt', 'w+') as f3:

            f3.writelines(thelist)

            f3.seek(0)

            print('\nText in',f3.name+':\n\n'+f3.read())

        continue

    elif useropt == 4:

        newcon = (f1.read()).replace('This','That')

        f1.seek(0)

        f1.truncate()

        f1.write(newcon)

        f1.seek(0)

        print('Text after replacing:\n\n'+f1.read())

        continue

    elif useropt == 5:

        print('\n'+(f1.read()).replace('This',''))

        continue

    elif useropt == 6:

        f1.close()

        break

    else:

        print('\nPlease choose a number between 1 and 6')

        continue

Output:

(next page)

First 2 options:Text

Description automatically generated

(next page)

Text

Description automatically generatedNext 2 options:

(next page)

Text

Description automatically generatedLast 2 options:

Q16. Following is the structure of each record in a data file named “PRODUCT.DAT”.

**{"prod\_code":value, "prod\_desc":value, "stock":value}**

The values for prod\_code and prod\_desc are strings, and the value for stock is an integer.

Write a menu driven program using functions

* to enter records
* to display all records
* to update the file with a new value of stock. The stock and the product\_code, whose stock is to be updated, are to be input during the execution of the function.

Code:

# Binary File Menu-driven Program 1

import pickle

def entrec(pcode,pdesc,stock):

    record = {'prod\_code':pcode, 'prod\_desc':pdesc, 'stock':stock}

    with open('Files/PRODUCT.DAT', 'ab') as f:

        pickle.dump(record,f)

def disrec():

    print('')

    with open('Files/PRODUCT.DAT', 'rb') as f:

        while True:

            try:

                print(pickle.load(f))

            except EOFError:

                break

def updstock(pcode):

    datalist = []

    with open('Files/PRODUCT.DAT', 'rb+') as f:

        while True:

            try:

                rec = pickle.load(f)

                datalist.append(rec)

            except EOFError:

                break

        for i in datalist:

            if i['prod\_code'] == pcode:

                try:

                    i['stock'] = int(input('Enter new value of stock: '))

                    break

                except:

                    print('\nPlease enter an integer value for stock.')

                    break

            else:

                continue

        f.seek(0)

        for i in datalist:

            pickle.dump(i,f)

while True:

    print('\n[1] Enter records\n[2] Display records\n[3] Update records\n[4] Quit')

    try:

        useropt = int(input("\nSelect [1], [2], [3], or [4]: "))

    except:

        print('\nPlease enter a number.')

        continue

    if useropt == 1:

        try:

            entrec(input('\nEnter Product Code: '),

            input('Enter Product Description: '),

            int(input('Enter value of Stock: ')))

        except:

            print('\nPlease enter an integer value for Stock')

    elif useropt == 2:

        disrec()

    elif useropt == 3:

        updstock(input('\nEnter the product code of the item that you want to update: '))

    elif useropt == 4:

        break

    else:

        print('\nPlease choose a number between 1 and 4')

Output:

(next page)

Entering records:Text

Description automatically generated

Text

Description automatically generatedDisplaying records:

Updating record:

Text, timeline

Description automatically generated

Q17. Given a binary file “STUQ2.DAT”, containing records of the following type:

**[S\_Admno, S\_Name, Percentage]**

Where these three values are:

**S\_Admno – Admission Number of student (string)**

**S\_Name – Name of student (string)**

**Percentage – Marks percentage of student (float)**

Write a menu driven program using functions

* to enter records
* to display all records
* to read contents of the file “STUDENT.DAT” and display the details of those students whose percentage is above 75.

Code:

# Binary File Menu-driven Program 2

import pickle

def entrec(admno,name,perc):

    record = [admno,name,perc]

    with open('Files/STUQ2.DAT','ab') as f:

        pickle.dump(record,f)

def disrec():

    print('')

    with open('Files/STUQ2.DAT', 'rb') as f:

        while True:

            try:

                print(pickle.load(f))

            except EOFError:

                break

def condrec():

    print('')

    with open('Files/STUQ2.DAT', 'rb') as f:

        while True:

            try:

                currec = pickle.load(f)

                if (currec)[2] > 75:

                    print(currec)

            except EOFError:

                break

while True:

    print('\n[1] Enter records\n[2] Display records\n[3] Display records with percentage above 75\n[4] Quit')

    try:

        useropt = int(input("\nSelect [1], [2], [3], or [4]: "))

    except:

        print('\nPlease enter a number.')

        continue

    if useropt == 1:

        try:

            entrec(input('\nEnter Admission Number: '),

            input('Enter Name of Student: '),

            float(input('Enter Marks Percentage of Student: ')))

        except:

            print('\nPlease enter a float value for percentage.')

    elif useropt == 2:

        disrec()

    elif useropt == 3:

        condrec()

    elif useropt == 4:

        break

    else:

        print('\nPlease enter a value between 1 and 4')

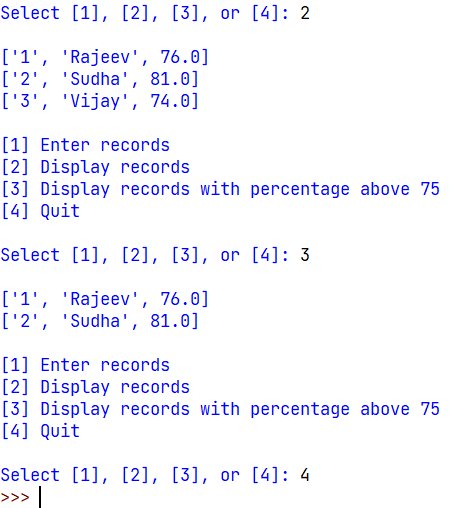
Output:

(next page)

Entering records: Text

Description automatically generated

Displaying records:



Q18: Assuming the tuple Vehicle as follows:

**(vehicletype, no\_of\_wheels)**

where vehicletype is a string and no\_of\_wheels is an integer.

Write a menu driven program to perform the following in binary file **vehicle.dat** using functions

* to enter records
* to display all records
* to count and display the number of records present in the file.

Code:

# Binary File Menu-driven Program 3

import pickle

def entrec(vtype,wheelno):

    vrec = (vtype,wheelno)

    with open('Files/vehicle.dat', 'ab') as f:

        pickle.dump(vrec,f)

def disrec():

    print('')

    with open('Files/vehicle.dat', 'rb') as f:

        while True:

            try:

                print(pickle.load(f))

            except EOFError:

                break

def countrec():

    reccount = 0

    with open('Files/vehicle.dat', 'rb') as f:

        try:

            for i in pickle.load(f):

                reccount += 1

        except:

            pass

    print('\nNumber of records is:',reccount)

while True:

    print('\n[1] Enter records\n[2] Display records\n[3] Display number of records\n[4] Quit')

    try:

        useropt = int(input("\nSelect [1], [2], [3], or [4]: "))

    except:

        print('\nPlease enter a number.')

        continue

    if useropt == 1:

        try:

            entrec(input('\nEnter Vehicle Type: '),

            int(input('Enter Number of Wheels: ')))

        except:

            print('\nPlease enter an integer value for number of wheels.')

    elif useropt == 2:

        disrec()

    elif useropt == 3:

        countrec()

    elif useropt == 4:

        break

    else:

        print('\nPlease enter a value between 1 and 4')

Output:

Entering records:

Graphical user interface, text, application

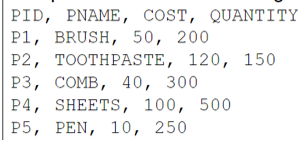
Description automatically generated

Displaying records:

Text

Description automatically generated

Q19. Create a file PRODUCT.CSV. Sample data of the file is as follows:



Write a menu driven program using functions

* to add sample data to the file
* to display all the records
* to copy/transfer only those records from the file PRODUCT.CSV to another file “PRO1.CSV” whose quantity is more than 150. Also include the first row with headings.
* To display the total cost of all the products of the file PRODUCT.CSV
* To search and display the record of that product from the file product.csv which has maximum cost.

Code:

# CSV Menu-driven Program 1

import csv

def adddata():

    sampledata = [

        ['PID','PNAME','COST','QUANTITY'],

        ['P1','BRUSH',50,200],

        ['P2','TOOTHPASTE',120,150],

        ['P3','COMB',40,300],

        ['P4','SHEETS',100,500],

        ['P5','PEN',10,250]

    ]

    with open('Files/PRODUCT.CSV', 'w', newline='') as f:

        (csv.writer(f)).writerows(sampledata)

def disrec():

    print('')

    with open('Files/PRODUCT.CSV', 'r') as f:

        csvread = csv.reader(f)

        next(csvread)

        for i in csvread:

            print(i)

def copyrec():

    newrec = []

    with open('Files/PRODUCT.CSV', 'r') as f1:

        csvread = csv.reader(f1)

        newrec.append(next(csvread))

        for i in csvread:

            if int(i[3]) > 150:

                newrec.append(i)

    with open('Files/PRO1.CSV', 'w', newline='') as f2:

        (csv.writer(f2)).writerows(newrec)

def totalcost():

    costsum = 0

    with open('Files/PRODUCT.CSV', 'r') as f:

        csvread = csv.reader(f)

        next(csvread)

        for i in csvread:

            costsum += (int(i[2])\*int(i[3]))

    print('\nTotal cost of all products is:',costsum)

def maxcost():

    highestcost = 0

    with open('Files/PRODUCT.CSV', 'r') as f:

        csvread = csv.reader(f)

        next(csvread)

        for i in csvread:

            if int(i[2]) > highestcost:

                highestcost = int(i[2])

        f.seek(0)

        next(csvread)

        for i in csvread:

            if int(i[2]) == highestcost:

                print('\nProduct with highest cost is:\n\n',i)

while True:

    print('\n[1] Add Sample Data\n[2] Display records\n[3] Copy records whose quantity is more than 150 to "PRO1.CSV"\n[4] Display total cost of all products\n[5] Show record with highest cost\n[6] Quit')

    try:

        useropt = int(input("\nSelect [1], [2], [3], [4], [5], or [6]: "))

    except:

        print('\nPlease enter a number.')

        continue

    if useropt == 1:

        adddata()

    elif useropt == 2:

        disrec()

    elif useropt == 3:

        copyrec()

    elif useropt == 4:

        totalcost()

    elif useropt == 5:

        maxcost()

    elif useropt == 6:

        break

    else:

        print('\nPlease enter a value between 1 and 4')

Output:

Text

Description automatically generated

Text

Description automatically generated

Q20. Create a file Tour.csv having headings as follows TID,DESTINATION,DAYS,FARE. Sample data of file is as follows:



Write a menu driven program using functions

* to add sample data to the file
* to display all the records
* to read the file tour.csv and display the records where fare is between 500 and 750. If no such record is found in the file then display an appropriate message on the screen.

Code:

# CSV Menu-driven Program 2

import csv

def adddata():

    sampledata = [

        ['TID','DESTINATION','DAYS','FARE'],

        ['T10','AUSTRALIA',10,300],

        ['T11','AUSTRIA',15,750],

        ['T12','RAJASTHAN',10,700],

        ['T13','FRANCE',12,650]

    ]

    with open('Files/Tour.csv', 'w', newline='') as f:

        (csv.writer(f)).writerows(sampledata)

def disrec():

    print('')

    with open('Files/Tour.csv', 'r') as f:

        csvread = csv.reader(f)

        next(csvread)

        for i in csvread:

            print(i)

def findrec():

    print('')

    found = False

    with open('Files/Tour.csv', 'r') as f:

        csvread = csv.reader(f)

        next(csvread)

        for i in csvread:

            if int(i[3]) > 500 and int(i[3]) < 750:

                print(i)

                found = True

    if found is False:

        print('\nNo such record found.')

while True:

    print('\n[1] Add Sample Data\n[2] Display records\n[3] Display records where fare is between 500 and 750\n[4] Quit')

    try:

        useropt = int(input("\nSelect [1], [2], [3], or [4]: "))

    except:

        print('\nPlease enter a number.')

        continue

    if useropt == 1:

        adddata()

    elif useropt == 2:

        disrec()

    elif useropt == 3:

        findrec()

    elif useropt == 4:

        break

    else:

        print('\nPlease enter a value between 1 and 4')

Output:

(next page)

Text

Description automatically generated

Q21. Write a function in python, **MakePush(Package)** and **MakePop(Package)** to add a new Package and delete a Package from a List of Package Description, considering them to act as push and pop operations of the Stack data structure. Implement the complete menu driven program

Code:

# Stacks Menu-driven Program 1

def MakePush(Package):

    PackageList.append(Package)

def MakePop():

    try:

        print('\nSuccessfully popped',PackageList.pop())

    except:

        print('\nStack is empty!')

def ShowList():

    print('\n\_\_\_\_\_\_\_\_\_\_\n')

    print(\*PackageList[::-1], sep='\n')

    print('\_\_\_\_\_\_\_\_\_\_\n')

PackageList = []

while True:

    print('\n[1] Show stack\n[2] Push item to stack\n[3] Pop item from stack\n[4] Quit')

    try:

        useropt = int(input("\nSelect [1], [2], [3], or [4]: "))

    except:

        print('\nPlease enter a number.')

        continue

    if useropt == 1:

        ShowList()

    elif useropt == 2:

        MakePush(input("\nEnter package to push: "))

    elif useropt == 3:

        MakePop()

    elif useropt == 4:

        break

    else:

        print('\nPlease enter a value between 1 and 4')

Output:

Text

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Q22. Write the functions in Python push (stk, item ) and pop(stk) to check whether the stack is empty, to add a new item, to delete an item and display the stack respectively. Implement the menu driven program.

Code:

# Stacks Menu-driven Program 2

stack1 = []

def push(stk,item):

    stk.append(item)

def pop(stk):

    try:

        print('\nSuccessfully popped',stk.pop())

    except:

        print('\nStack is empty!')

def check(stk):

    if len(stk) != 0:

        print('\nStack is not empty.')

    else:

        print('\nStack is empty.')

def show(stk):

    print('\n\_\_\_\_\_\_\_\_\_\_\n')

    print(\*stk[::-1], sep='\n')

    print('\_\_\_\_\_\_\_\_\_\_\n')

while True:

    print('\n[1] Show stack\n[2] Push item to stack\n[3] Pop item from stack\n[4] Check stack\n[5] Quit')

    try:

        useropt = int(input("\nSelect [1], [2], [3], [4], or [5]: "))

    except:

        print('\nPlease enter a number.')

        continue

    if useropt == 1:

        show(stack1)

    elif useropt == 2:

        push(stack1, input("\nEnter item to push to stack: "))

    elif useropt == 3:

        pop(stack1)

    elif useropt == 4:

        check(stack1)

    elif useropt == 5:

        break

    else:

        print('\nPlease enter a value between 1 and 5')

Output:

Text

Description automatically generated

Text

Description automatically generatedText

Description automatically generated

Q23. Write a program to perform push operation on a stack to push all prime numbers from a list entered by a user.

Code:

# Check for prime numbers and push into stack

def pushprimes(inlist):

    outlist = []

    isprime = False

    if 2 in inlist:

        outlist.append(2)

    for i in inlist:

        for dnm in range(2,int(i)):

            if int(i)%dnm != 0:

                isprime = True

            else:

                isprime = False

                break

        else:

            if isprime is True:

                outlist.append(i)

    global primestack

    primestack = outlist

primestack = []

numlist = []

while True:

    print('Current list:',numlist)

    useropt = input("Enter numbers to add to list (type a string to stop adding): ")

    try:

        num = int(useropt)

        numlist.append(num)

        continue

    except:

        break

pushprimes(numlist)

print('\nThe final stack is:')

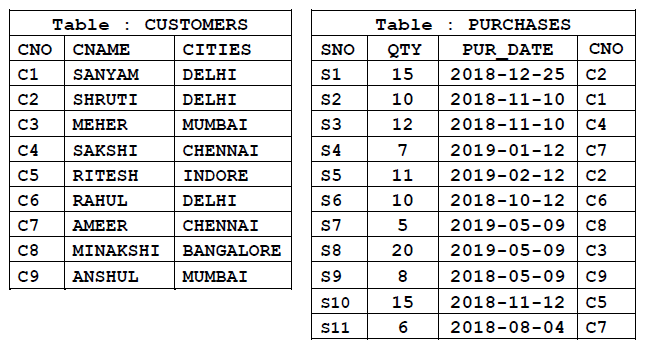
print(\*primestack, sep='\n')

Output:

Text

Description automatically generated

Q24. Write SQL queries for (i) to (vi), which are based on the tables CUSTOMERS and PURCHASES given below:



(i) To display details of all CUSTOMERS whose CITIES are neither Delhi nor Mumbai.

(ii) To display the CNAME and CITIES of all CUSTOMERS in ascending order of their CNAME.

(iii) To display the number of CUSTOMERS along with their respective CITIES in each of the CITIES.

(iv) To display details of all PURCHASES whose Quantity is more than 15.

(v) To display customer name, quantity and purchase date for only those purchases whose quantity is either 10 or 20.

(vi) To display the maximum purchase date

Code:

(i) SELECT \* FROM CUSTOMERS WHERE CITIES NOT IN ('DELHI','MUMBAI');

(ii) SELECT CNAME, CITIES FROM CUSTOMERS ORDER BY CNAME ASC;

(iii) SELECT COUNT(\*) AS 'Number of Customers', CITIES FROM CUSTOMERS GROUP BY CITIES;

(iv) SELECT \* FROM PURCHASES WHERE QTY > 15;

(v) SELECT CUSTOMERS.CNAME, PURCHASES.QTY, PURCHASES.PUR\_DATE FROM CUSTOMERS INNER JOIN PURCHASES ON CUSTOMERS.CNO = PURCHASES.CNO WHERE QTY = 10 OR QTY = 20;

(vi) SELECT MAX(PUR\_DATE) FROM PURCHASES;

**Q25. Consider the following tables product and client.**

Table

Description automatically generated

Table

Description automatically generated

Develop a complete menu driven application using python mySQL connectivity based on following parameters. Create functions wherever required.

1. Database name:STORE
2. Table names: PRODUCT and CLIENT (as shown above)
3. Primary keys: pid, cid
4. Write suitable code in python to create database, tables and to insert the records.
5. Create a function to display the client name and the product purchased by the client in descending order of client names
6. Create a function to increase the price of all the products by 5%. Now display all the records of product table.
7. Create a function to remove the records of clients who are from Bangalore. Now display all the records of Client table.
8. Create a function to display number of clients from each city.
9. Create a function to increase the width of column city to 50.

Code:

# MySQL Menu-driven Program

import mysql.connector

import getpass

def cnameprod():

    cur.execute('SELECT CLIENT.CLIENTNAME, PRODUCT.PRODUCTNAME FROM CLIENT\

        INNER JOIN PRODUCT ON CLIENT.P\_ID = PRODUCT.P\_ID\

        ORDER BY CLIENT.CLIENTNAME DESC;')

    print('')

    for i in cur.fetchall():

        print(i)

def priceinc():

    cur.execute('UPDATE PRODUCT SET PRICE = PRICE + (PRICE\*0.05);')

    cur.execute('SELECT \* FROM PRODUCT;')

    print('')

    for i in cur.fetchall():

        print(i)

def removebangalore():

    cur.execute("DELETE FROM CLIENT WHERE City = 'Bangalore'")

    cur.execute('SELECT \* FROM CLIENT;')

    print('')

    for i in cur.fetchall():

        print(i)

def clientno():

    cur.execute("SELECT COUNT(\*) AS 'Number of Clients', City FROM CLIENT GROUP BY City;")

    print('')

    for i in cur.fetchall():

        print(i)

def incwidth():

    cur.execute("ALTER TABLE CLIENT MODIFY COLUMN City varchar(50);")

while True:

    rootpwd = getpass.getpass('Please enter the root password: ')

    try:

        db = mysql.connector.connect(host='localhost', user='root', password=rootpwd, autocommit=True)

        break

    except:

        print('\nTry again.')

cur = db.cursor(buffered=True)

cur.execute('CREATE DATABASE STORE;')

cur.execute('USE STORE;')

cur.execute('CREATE TABLE PRODUCT(\

    P\_ID varchar(255) NOT NULL,\

    PRODUCTNAME varchar(255),\

    MANUFACTURER varchar(255),\

    PRICE int,\

    PRIMARY KEY (P\_ID)\

    );')

cur.execute('CREATE TABLE CLIENT(\

    C\_ID int NOT NULL,\

    CLIENTNAME varchar(255),\

    City varchar(255),\

    P\_ID varchar(255),\

    PRIMARY KEY (C\_ID)\

    );')

cur.execute("INSERT INTO PRODUCT (P\_ID, PRODUCTNAME, MANUFACTURER, PRICE)\

    VALUES \

        ('TP01','TALCOM POWDER','LAK',40),\

        ('FW05','FACE WASH','ABC',45),\

        ('BS01','BATH SOAP','ABC',55),\

        ('SH06','SHAMPOO','XYZ',120),\

        ('FW12','FACE WASH','XYZ',95);")

cur.execute("INSERT INTO CLIENT (C\_ID, CLIENTNAME, CITY, P\_ID)\

    VALUES \

        (01,'COSMETIC SHOP','Delhi','FW05'),\

        (06,'TOTAL HEALTH','Mumbai','BS01'),\

        (12,'LIVE LIFE','Delhi','SH06'),\

        (15,'PRETTY WOMAN','Delhi','FW12'),\

        (16,'DREAMS','Bangalore','TP01');")

while True:

    print('\n[1] Display Client Name and Product Purchased\

        \n[2] Increase price by 5%\

        \n[3] Remove clients from Bangalore\

        \n[4] Display number of clients from each city\

        \n[5] Change width of City column to 50\

        \n[6] Quit')

    try:

        useropt = int(input("\nSelect [1], [2], [3], [4], [5], or [6]: "))

    except:

        print('\nPlease enter a number.')

        continue

    if useropt == 1:

        cnameprod()

    elif useropt == 2:

        priceinc()

    elif useropt == 3:

        removebangalore()

    elif useropt == 4:

        clientno()

    elif useropt == 5:

        incwidth()

    elif useropt == 6:

        break

    else:

        print('\nPlease enter a value between 1 and 6')

Output:

SQL Output:

Table

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated with low confidence

Python Output:

Text

Description automatically generated

Text

Description automatically generatedTable

Description automatically generated

Text

Description automatically generatedText

Description automatically generatedText

Description automatically generated with low confidence

Text

Description automatically generatedTable

Description automatically generated