Download all programs: https://github.com/LonleyC/labMSE/archive/master.zip

q1.py

```
"""
The finance department of a company wants to calculate the monthly pay of one of
its employee. Monthly pay should be calculated as mentioned in the formula below
and display all the employee details.
Monthly Pay= No. of hours worked in a week * Pay rate per hour * No .of weeks in a
month
Write a Python Program to implement the problem.
"""

week_hours = float(input("Enter weekly working hours : "))
hourly_pay = float(input("Enter pay rate per hour : "))
working_weeks = float(input("Enter working weeks in a month : "))

monthly_pay = week_hours * hourly_pay * working_weeks

print("Calculated monthly pay : Rs", monthly_pay)
```

q2.py

```
"""
The finance department of a company wants to calculate the monthly net pay of one
of its employee by finding the income tax to be paid (in Indian Rupees) and the net
salary after the income tax deduction. The employee should pay the income tax
based on the following table:
Gross Salary(In Rs) | Tax Percentage
--------------------|---------------
Below 5,000         | Nil
5,001 to 10,000     | 10%
10,001 to 20,000    | 20%
More than 20,000    | 30%

Display the employee id, basic salary, allowances, gross pay, income tax and net pay
using Python Programming.
"""

emp_id = int(input("Enter Employee ID : "))
basic_pay = float(input("Enter Basic Pay : "))
allowances = float(input("Enter Allowances : "))

gross_pay = basic_pay + allowances

if gross_pay > 20000:
    income_tax = gross_pay * 0.3
elif gross_pay > 10000:
    income_tax = gross_pay * 0.2
elif gross_pay > 5000:
    income_tax = gross_pay * 0.1
else:
    income_tax = 0

net_pay = gross_pay - income_tax

print("Employee ID : ", emp_id)
print("Basic Pay   : Rs", basic_pay)
print("Allowances  : Rs", allowances)
print("Gross Pay   : Rs", gross_pay)
print("Income Tax  : Rs", income_tax)
print("Net Pay     : Rs", net_pay)
```

q3.py

```
"""
In the retail application, display the details of the customer like bill id, customer id,
bill amount and customer name. But the retail shop wants the customer name to be
between 3 to 20 characters. Write a Python Program to implement the case study.
"""
bill_id = input("Enter the Bill ID : ")
cust_id = input("Enter the Customer ID : ")
bill_amt = float(input("Enter the bill amount : "))
cust_name = input("Enter the Customer name :")
flag = True
while flag:
    if(len(cust_name)<3 or len(cust_name)>20):
        print("The name is invalid!")
        cust_name=input("Enter the Customer Name : ")
    else:
        flag = False

print("BILL ID : ",bill_id)
print("CUSTOMER ID : ",cust_id)
print("BILL AMOUNT : ",bill_amt)
print("CUSTOMER NAME : ",cust_name)
```

q4.py

```
#Write a Python program to check whether the given string is palindrome or not.

def easy_pal(string):
    if string == string[::-1]:
        print(string + " is a palindrome")
    else:
        print(string + " is not a palindrome")

def harder_pal(string):
    LENGTH = len(string)

    for i in range(LENGTH//2):
        if string[i] != string[LENGTH - i - 1]:
            print(string + " is not a palindrome")
            return

    print(string + " is a palindrome")

inp = input("Enter a string: ")

easy_pal(inp)

harder_pal(inp)
```

q5.py

```python
#Write a Python program to generate first ‘n’ Fibonacci numbers.

#takes an integer argument for number of terms required
#Pronunciation of fibonacci : fee-bo-na-chee (chi like cheetos) LOOK IT UP
def gen_fibonacci(terms):
    if terms <= 0:
        return
    if terms is 1:
        print('0')
        return

    a, b, c = 0, 1, 0
    for i in range(terms):
        print(c, ' ', end='')
        c = a + b
        a, b = b, c


terms = int(input("enter number of terms required : "))
gen_fibonacci(terms)
```

q6.py

```
"""
Consider the scenario of processing marks of students for a course in student
management system. Given below is the list of marks scored by students. Find top
three scorers for the course and also display the average marks scored by all
students. Implement the solution using Python Programming.
Student Name Marks Scored
John    |86.5
Jack    |91.2
Jill    |84.5
Harry   |72.1
Joe     |80.5

"""

marks_list = {
    'John': 86.5,
    'Jack': 91.2,
    'Jill': 84.5,
    'Harry': 72.1,
    'Joe': 80.5
}

sorted_keys = sorted(marks_list, key=marks_list.get, reverse=True)

print("Top 3 students: ")
for student in sorted_keys[:3]:
    print(student, " : ", marks_list[student])

sum = 0
for student in marks_list:
    sum += marks_list[student]

print("Average mark : ", sum/len(marks_list))
```

q7.py

```python
#Write a program to count and display the number of capital letters in a given string.

def count_capitals(string):
    count = 0
    for char in string:
        if char.isupper():
            count += 1
    return count

string = input("Enter a string : ")
print("Number of upper case letter in ", string, " is ", count_capitals(string))
```

q8.py

```python
#Write a program to count the number of each vowel in a given string.

string = input("Enter a string : ")
count = {
    'a': 0,
    'e': 0,
    'i': 0,
    'o': 0,
    'u': 0,
    'A': 0,
    'E': 0,
    'I': 0,
    'O': 0,
    'U': 0
}
for char in string:
    if char in "AEIOUaeiou":
        count[char] += 1

for vow in count:
    if count[vow] is not 0:
        print(vow, " : ", count[vow]);
```

q9.py

```python
#Write a program to remove all punctuations like “’!()-[]{};:’’’,\,<>,/,?,@,#,$, %^&*_~” from the string provided by the user.

def remove_punctuations(string):
    res = ""
    for char in string:
        if char not in "’!()-[]{};:’’’,\<>/?@#$%^&*_~":
            res += char

    return res
```

```
string = input("Enter a string : ")
print("your string without punctuations : ", remove_punctuations(string))

q10.py
'''
Consider two strings, String1 and String2 and display the merged_string as
output. The merged_string should be the capital letters from both the strings in
the order they appear.
Sample Input: String1: I Like C String2: Mary Likes Python
Merged_string should be ILCMLP
'''

def get_caps_only(string1, string2):
    res = ""
    merge = string1 + string2

    for char in merge:
        if char.isupper():
            res += char

    return res

string1 = input("Enter first string : ")
string2 = input("Enter second string : ")

print("Merged result : ", get_caps_only(string1, string2))

q11.py
'''
With a given integral number n, write a program to generate a dictionary that
contains (i, i*i) such that is an integral number between 1 and n (both included)
and then the program should print the dictionary. Suppose the following input
is supplied to the program: 8 Then, the output should be:
{1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64}
'''

def gen_squares(upper_limit):
    res = dict()

    for i in range(1, upper_limit + 1):
        res[i] = i*i

    return res

upper_limit = int(input("Enter upper limit for dictonary : "))
print(gen_squares(upper_limit))

q12.py
'''
Write a binary search function which searches an item in a sorted list.
The function should return the index of element to be searched in the list.
'''

def binary_search(sorted_list, search_item):

    lower, upper = 0, len(sorted_list) - 1

    while lower <= upper:
        mid_index = (lower + upper) // 2 # // will only retuen int after division
        mid_item = sorted_list[mid_index]

        if search_item == mid_item:
            return mid_index + 1

        if mid_item > search_item:
            upper = mid_index - 1

        else:
            lower = mid_index + 1

    return -1


sorted_list = input("Enter list items [in sorted order] : ")
search_item = int(input("Enter the element to search : "))

sorted_list = sorted_list.split(' ')
sorted_list = list(map(int, sorted_list))   #converts elements from string to integers

result = binary_search(sorted_list, search_item)

if result is -1:
    print("Element not found")
else:
    print(search_item, "found at position ", result)

q13.py
'''
Design a class named Rectangle to represent a rectangle. class contains:
* Two data fields named width and height .
* A constructor that creates a rectangle with the specified width and height .
* The default values are 1 and 2 for the width and height, respectively.
* A method named getArea() that returns the area of this rectangle.
* A method named getPerimeter() that returns the perimeter.
Write a test program that creates two Rectangle objects—one with width 4 and
height 40 and the other with width 3.5 and height 35.7. Display the width,
height, area, and perimeter of each rectangle in this order.
'''

class Rectangle:

    def __init__(self, width = 1, height = 2):
        self.width = width
        self.height = height

    def get_area(self):
        return self.height * self.width;

    def get_perimeter(self):
```

```
        return 2 * (self.height + self.width)

    def display(self):
        print("Width     : ", self.width)
        print("Height    : ", self.height)
        print("Area      : ", self.get_area())
        print("Perimeter : ", self.get_perimeter())



#test
rect1 = Rectangle(4, 40)
rect2 = Rectangle(3.5, 35.7)

print("Rectangle 1 ")
rect1.display()

print("Rectangle 2 ")
rect2.display()

q14.py
"""
Design a class named Account that contains:
* A private int data field named id for the account.
* A private float data field named balance for the account.
* A private float data field named annualInterestRate that stores the
  current interest rate.
* A constructor that creates an account with the specified id (default 0),
  initial balance (default 100), and annual interest rate (default 0).
* The accessor and mutator methods for id , balance , and
  annualInterestRate .
* A method named getMonthlyInterestRate() that returns the monthly
  interest rate.
* A method named getMonthlyInterest() that returns the monthly
  interest.
* A method named withdraws that withdraws a specified amount from
  the account.
* A method named deposit that deposits a specified amount to the
  account.
(Hint: The method getMonthlyInterest() is to return the monthly interest
amount, not the interest rate. Use this formula to calculate the monthly
interest: balance*monthlyInterestRate. monthlyInterestRate is
annualInterestRate / 12 . Note that annualInterestRate is a percent (like 4.5%).
You need to divide it by 100 .)
Write a test program that creates an Account object with an account id of
1122, a balance of $20,000, and an annual interest rate of 4.5%. Use the
withdraw method to withdraw $2,500, use the deposit method to deposit
$3,000, and print the id, balance, monthly interest rate, and monthly interest.
"""


class Account:
    def __init__(self, id = 0, balance = 100, annualIR = 0):
        self.__id = int(id)
        self.__balance = float(balance)
        self.__annualIR = float(annualIR)

    # accessor methods - getters
    def get_id(self):
        return self.__id

    def get_balance(self):
        return  self.__balance

    def get_annual_ir(self):
        return self.__annualIR

    # accessor methods - setters
    def set_id(self, id):
        self.__id = id

    def set_balance(self, balance):
        self.__balance = balance

    def set_annual_ir(self, anualIR):
        self.__annualIR = anualIR

    def get_monthly_ir(self):
        return self.__annualIR / 12

    def get_monthly_interest(self):
        return self.__balance * (self.get_monthly_ir() / 100)

    def withdraw(self, amount):
        if amount > self.__balance:
            return False
        else:
            self.__balance -= amount
            return True

    def deposit(self, amount):
        if amount < 0:
            return False

        self.__balance += amount
        return True

    def info(self):
        print(" ID                       : ", self.__id)
        print(" Balance                  : $", self.__balance)
        print(" Annual Interest Rate     : ", self.__annualIR, "%")
        print(" Monthly Interest Rate    : ", self.get_monthly_ir(), "%")
        print(" Monthly Interest         : $", self.get_monthly_interest())


# test

new_account = Account(1122, 20000, 4.5)
print("New account created!")
```

```
new_account.info()

print("Withdrawing $2500")
new_account.withdraw(2500)
new_account.info()

print("depositing $3000")
new_account.deposit(3000)
new_account.info()
```

q15.py

```
'''
Write a function that returns the number of days in a year using the following
header:
def numberOfDaysInAYear(year):
Write a test program that displays the number of days in the years from 2010 to
2020.
'''

def is_leap(year):
    if year % 4 != 0:
        return False

    if year % 100 == 0:
        if year % 400 == 0:
            return True
        else:
            return False
    else:
        return True

def numberOfDaysInAYear(year):
    if is_leap(year):
        return 366
    else:
        return 365

#test

for i in range(2010, 2021):
    print("There are ", numberOfDaysInAYear(i), " days in ", i)
```

q16.py

```
'''
(Display matrix of 0s and 1s) Write a function that displays an n-by-n matrix
using the following header:
def printMatrix(n):
Each element is 0 or 1, which is generated randomly. Write a test program that
prompts the user to enter n and displays an n-by-n matrix.
Sample run:
Enter n: 3
0 1 0
0 0 0
1 1 1
'''

import random

def gen_random_matrix(order):
    for i in range(order):
        for j in range(order):
            print(random.randint(0, 1), end=' ')
        print('')


#test
order = int(input("Enter order of matrix : "))
gen_random_matrix(order)
```

q17.py

```
key = ['D',   'B',   'D',   'C',   'C',   'D',   'A',   'E',   'A',   'D']

student_answers = [
    ['A'   ,'B',  'A',   'C',   'C',   'D',   'E',   'E',   'A',   'D'],
    ['D',  'B',   'A',   'B',   'C',   'A',   'E',   'E',   'A',   'D'],
    ['E',  'D',   'D',   'A',   'C',   'B',   'E',   'E',   'A',   'D'],
    ['C',  'B',   'A',   'E',   'D',   'C',   'E',   'E',   'A',   'D'],
    ['A',  'B',   'D',   'C',   'C',   'D',   'E',   'E',   'A',   'D'],
    ['B',  'B',   'E',   'C',   'C',   'D',   'E',   'E',   'A',   'D'],
    ['B',  'B',   'A',   'C',   'C',   'D',   'E',   'E',   'A',   'D'],
    ['E',  'B',   'E',   'C',   'C',   'D',   'E',   'E',   'A',   'D']
]

i = 0

for ans in student_answers:
    score=0
    for q in range(10):
        if key[q]==ans[q]:
            score=score+1
    i += 1
    print("Student ", i, " scored: ", score)
```

q18.py

```
'''
(Find the index of the smallest element) Write a function that returns the index of
the smallest element in a list of integers. If the number of such elements is greater
than 1, return the smallest index. Use the following header:
def indexOfSmallestElement(lst):
Write a test program that prompts the user to enter a list of numbers, invokes this
function to return the index of the smallest element, and displays the index.
'''
```

```
def indexOfSmallestElement(lst):
    LENGTH = len(lst)
    small_element, small_index = lst[0], 0

    for i in range(LENGTH):
        if lst[i] < small_element:
            small_element = lst[i]
            small_index = i

    return small_index + 1

#test

element_list = input("Enter list items : ")

#convert list elements to int
element_list = element_list.split(' ')
print(element_list)
element_list = list(map(int, element_list))

print("Index of smallest element : ", indexOfSmallestElement(element_list))
```

q19.py

```
'''
Write the following function that tests whether the list has four consecutive numbers
with the same value:
def isConsecutiveFour(values):
Write a test program that prompts the user to enter a series of integers and reports
whether the series contains four consecutive numbers with the same value.
'''

def isConsecutiveFour(element_list):
    LENGTH = len(element_list)
    for i in range(LENGTH - 3):
        cur_ele = element_list[i]

        if element_list[i : i + 4] == [cur_ele, cur_ele, cur_ele, cur_ele]:
            return True

    return False


element_list = input("Enter list elements : ")
element_list = element_list.split(' ')
print("occurance of consecutive four in list : ", isConsecutiveFour(element_list))

#valid input : 1 2 2 2 2 3 4 (four consecutive 2s)
#invalid input : 1 2 3 4 5
```

q20.py

```
'''
Write a program that will count the number of characters, words, and lines in a file.
Words are separated by a white-space character. Your program should prompt the
user to enter a filename.
'''

file_name = input("Enter file name : ")

try:
    file = open(file_name)
    word_count = 0
    line_count = 0
    char_count = 0
    for line in file:
        line_count += 1
        char_count += len(line)
        word_count += line.count(' ') + 1    #number of spaces + 1

    print("Number of Lines : ", line_count)
    print("Number of Words : ", word_count)
    print("Number of Chars : ", char_count)
except FileNotFoundError:
    print("file not found")
```

q21.py

```
'''
Suppose that a text file contains an unspecified number of scores. Write a program
that reads the scores from the file and displays their total and average. Scores are
separated by blanks. Your program should prompt the user to enter a filename.
'''

file_name = input("Enter file name : ")

try:
    file = open(file_name)
    scores = []

    for line in file:
        line = line.split(' ')
        line = list(map(int, line))

        scores += line

    total = 0

    for score in scores:
        total += score

    print("Total score : ", total)
    print("Average score : ", total / len(scores))

except FileNotFoundError:
    print("file not found")
```

q22.py

```
'''
```

```
Design a class named Triangle that extends the GeometricObject class. The Triangle
class contains:
* Three float data fields named side1 , side2 , and side3 to denote the three
  sides of the triangle.
* A constructor that creates a triangle with the specified side1 , side2 , and
  side3 with default values 1.0 .
* The accessor methods for all three data fields.
* A method named getArea() that returns the area of this triangle.
* A method named getPerimeter() that returns the perimeter of this triangle.
* A method named _ _str_ _() that returns a string description for the triangle.
Write a test program that prompts the user to enter the three sides of the triangle, a
color, and 1 or 0 to indicate whether the triangle is filled. The program should create
a Triangle object with these sides and set the color and filled properties using the
input. The program should display the triangle's area, perimeter, color, and True or
False to indicate whether the triangle is filled or not.
'''

import math

class GeometricObject:
    def __init__(self, color, filled):
        self.color = color
        self.filled = filled

    def getPerimeter(self):
        pass

    def getArea(self):
        pass

class Triangle(GeometricObject):
    def __init__(self, color, filled, side1 = 1.0, side2 = 1.0, side3 = 1.0):
        GeometricObject.__init__(self, color, filled)
        self.side1 = side1
        self.side2 = side2
        self.side3 = side3

    #accessor methods - getters
    def getSide1(self):
        return self.side1

    def getSide2(self):
        return self.side2

    def getSide3(self):
        return self.side3

    #accessor methods - setters
    def setSide1(self, side):
        self.side1 = side

    def setSide2(self, side):
        self.side2 = side

    def setSide3(self, side):
        self.side3 = side

    def getPerimeter(self):
        return self.side1 + self.side2 + self.side3

    def getArea(self):
        # Heron's formula
        p = self.getPerimeter() / 2;

        area = math.sqrt(p * (p - self.side1) * (p - self.side2) * (p - self.side3))
        return area

    def __str__(self):
        return  "Side 1    : " + str(self.side1) + "\n" + \
                "Side 2    : " + str(self.side2) + "\n" + \
                "Side 3    : " + str(self.side3) + "\n" + \
                "Perimeter : " + str(self.getPerimeter()) + "\n" + \
                "Area      : " + str(self.getArea()) + "\n" + \
                "Filled    : " + str(self.filled) + "\n" + \
                "Color     : " + str(self.color)

#test

triangle1 = Triangle("RED", True)
triangle2 = Triangle("GREEN", False, 3, 4, 5)

print("Triangle 1 : ")
print(triangle1)

print("Triangle 2 : ")
print(triangle2)
```

q23.py

```
'''
Write a program that draws a rectangle or an oval, as shown in Figure below: The
user selects a figure from a radio button and specifies whether it is filled by selecting
a check button.
'''

from tkinter import *

class DrawArea(Frame):
    def __init__(self):
        super().__init__()

        self.canvas = Canvas(self, width=200, height=200)
        self.canvas.pack()
```

```
    def draw_rect(self, filled):
        if filled:
            self.canvas.create_rectangle(10, 10, 180, 180, fill="cyan")
        else:
            self.canvas.create_rectangle(10, 10, 180, 180)

    def draw_oval(self, filled):
        if filled:
            self.canvas.create_oval(10, 10, 200, 100, fill="cyan")
        else:
            self.canvas.create_oval(10, 10, 200, 100)


class MainWindow:
    def  __init__(self):
        self.root = Tk()

        self.draw = DrawArea()
        self.draw.pack()

        self.shape = IntVar()
        self.filled = BooleanVar()

        Radiobutton(
            self.root,
            text="Rectangle",
            variable=self.shape,
            value=1,
            command=self.shape_select
        ).pack()

        Radiobutton(
            self.root,
            text="Oval",
            variable=self.shape,
            value=2,
            command=self.shape_select,
        ).pack()

        Checkbutton(
            self.root,
            text="filled",
            variable=self.filled,
            onvalue=True,
            offvalue=False,
            command=self.shape_select
        ).pack()

        self.root.mainloop()

    def shape_select(self):
        self.draw.canvas.delete(ALL)

        if self.shape.get() == 1:
            self.draw.draw_rect(self.filled.get())
        else:
            self.draw.draw_oval(self.filled.get())


MainWindow()
```

q24.py

```
'''
Write a program that calculates the future value of an investment at a given interest
rate for a specified number of years. The formula for the calculation is as follows:
futureValue = investmentAmount * (1 + monthlyInterestRate) years* 12
Use text fields for users to enter the investment amount, years, and interest rate.
Display the future amount in a label field when the user clicks the Calculate button,
as shown.
'''

from tkinter import *
import math
```

```
class MainWindow:
    def __init__(self):
        self.root = Tk()
        self.root.title("account")

        self.invest_amnt = DoubleVar()
        self.intrest_rate = DoubleVar()
        self.years = DoubleVar()

        Label(
            self.root,
            text="investment ammount"
        ).grid(row=0, column=0)
        Entry(
            self.root,
            text=self.invest_amnt
        ).grid(row=0, column=1)

        Label(
            self.root,
            text="years"
        ).grid(row=1, column=0)
        Entry(
            self.root,
            text=self.years
        ).grid(row=1, column=1)

        Label(
            self.root,
            text="annual interest rate"
        ).grid(row=2, column=0)
        Entry(
            self.root,
            text=self.intrest_rate
        ).grid(row=2, column=1)
```

```
        Label(
            self.root,
            text="future investment"
        ).grid(row=3, column=0)

        Button(
            self.root,
            text="calculate",
            fg="blue",
            command=self.onclick
        ).grid(row=4, column=1)

        self.root.mainloop()

    def onclick(self):

        ans = self.invest_amnt.get() * (1 + self.intrest_rate.get()) ** (self.years.get() * 12)

        Label(
            self.root,
            text=str(ans)
        ).grid(row=3, column=1)


MainWindow()
```

q25.py

```
#Write a program that displays a still fan.


from tkinter import *

root = Tk()
canvas = Canvas(root, width=400, height=400)
canvas.pack()

#encloses circle in bound rectangle
bound = 50, 50, 300, 300

#draws 30degree arcs from angles 0, 90, 180, 270
for i in [0, 90, 180, 270]:
    canvas.create_arc(bound, start=i, extent=30, fill="red")


canvas.pack()
root.mainloop()
```

q26.py

```
"""
Design a Tkinter interface to perform the following operations on a database by
considering the table Student (USN: String, Name: String, Age: Int, Branch: String).
Display the success and failure message using MessageBox
a. Insert student details
b. Search the student details with USN=”4NM06CS001”
"""

import pymysql.cursors
from tkinter import *
from tkinter import simpledialog
from  tkinter import messagebox


class DBUI:
    def __init__(self, db):
        self.db = db
        self.root = Tk()

        self.name = StringVar()
        self.usn = StringVar()
        self.age = IntVar()
        self.branch = StringVar()

        Label(text="Enter Name : ").grid(row=0, column=0)
        Entry(text=self.name).grid(row=0, column=1)

        Label(text="Enter USN : ").grid(row=1, column=0)
        Entry(text=self.usn).grid(row=1, column=1)

        Label(text="Enter Age : ").grid(row=2, column=0)
        Entry(text=self.age).grid(row=2, column=1)

        Label(text="Enter Branch : ").grid(row=3, column=0)
        Entry(text=self.branch).grid(row=3, column=1)

        Button(text="Insert", command=self.insert).grid(row=4, column=0)
        Button(text="Search", command=self.search).grid(row=4, column=1)

        self.root.mainloop()

    def insert(self):
        try:
            cursor = self.db.cursor()

            query = "INSERT INTO student values('%s', '%s', %d, '%s');" % (self.name.get(), self.usn.get(), self.age.get(), self.branch.get())
            print(query)
            cursor.execute(query)
            self.db.commit()
            messagebox.showinfo("Success", "Inserted!")

        except Exception as ex:
            messagebox.showerror("Insert Error", ex)

    def search(self):
        try:

            usn_inp = simpledialog.askstring("Search", "Enter USN to search")
```

```
            cursor = self.db.cursor()
            query = "select * from student where usn = '%s';" % (usn_inp)
            print(query)

            cursor.execute(query)

            if cursor.rowcount == 0:
                raise Exception("No student of that USN found")
            else:
                result = cursor.fetchone()
                show_string  = "Name    : " + result[0] + "\n"
                show_string += "USN     : " + result[1] + "\n"
                show_string += "Age     : " + str(result[2]) + "\n"
                show_string += "Branch  : " + result[3] + "\n"
                messagebox.showinfo(usn_inp,  show_string)

        except Exception as ex:
            messagebox.showerror("Not found", ex)


# Connect to the database
connection = pymysql.connect(host='localhost',
                             user='root',
                             password='admin',
                             db='pythonlab')

DBUI(connection)
```

```
q27.py
"""
Design a Tkinter interface to perform the following operations on a database by
considering the table Employee (SSN:Int, Fname: String, LName: String, Age: Int,
Place: String, Salary: Int). Display the success and failure message using MessageBox
a. Insert employee details
b. Delete the details of employee
c. Update the employee details.
"""
```

```
"""
Design a Tkinter interface to perform the following operations on a database by
considering the table Employee (SSN:Int, Fname: String, LName: String, Age: Int,
Place: String, Salary: Int). Display the success and failure message using MessageBox
a. Insert employee details
b. Delete the details of employee
c. Update the employee details.
"""

import pymysql.cursors
from tkinter import *
from tkinter import simpledialog
from  tkinter import messagebox


class DBUI:
    def __init__(self, db):
        self.db = db
        self.root = Tk()

        self.ssn = StringVar()
        self.lname = StringVar()
        self.age = IntVar()
        self.place = StringVar()
        self.sal = IntVar()

        Label(text="Enter SSN : ").grid(row=0, column=0)
        Entry(text=self.ssn).grid(row=0, column=1)

        Label(text="Enter Last Name : ").grid(row=1, column=0)
        Entry(text=self.lname).grid(row=1, column=1)

        Label(text="Enter Age : ").grid(row=2, column=0)
        Entry(text=self.age).grid(row=2, column=1)

        Label(text="Enter Place : ").grid(row=3, column=0)
        Entry(text=self.place).grid(row=3, column=1)

        Label(text="Enter Salary : ").grid(row=4, column=0)
        Entry(text=self.sal).grid(row=4, column=1)

        Button(text="Insert", command=self.insert).grid(row=5, column=0)
        Button(text="Delete", command=self.delete).grid(row=5, column=1)
        Button(text="Update", command=self.update).grid(row=6, column=0)

        self.root.mainloop()

    def insert(self):
        try:
            cursor = self.db.cursor()

            query = "INSERT INTO employee values('%s', '%s', %d, '%s', %d);" % (self.ssn.get(), self.lname.get(), self.age.get(),
self.place.get(), self.sal.get())
            print(query)
            cursor.execute(query)
            self.db.commit()
            messagebox.showinfo("Success", "Inserted!")

        except Exception as ex:
            messagebox.showerror("Insert Error", ex)

    def delete(self):
        try:

            usn_inp = simpledialog.askstring("Search", "Enter SSN to delete")
```

```
            cursor = self.db.cursor()
            query = "delete from employee where SSN = '%s';" % (usn_inp)
            cursor.execute(query)
            self.db.commit()

            messagebox.showinfo("Success", "Employee deleted!")

        except Exception as ex:
            messagebox.showerror("Error", ex)


    def update(self):
        try:
            cursor = self.db.cursor()

            query = "UPDATE employee Set LName = '%s', Age = %d, Place = '%s', Salary = %d WHERE SSN = '%s';" % (self.lname.get(),
self.age.get(), self.place.get(), self.sal.get(), self.ssn.get())
            print(query)
            cursor.execute(query)
            self.db.commit()
            messagebox.showinfo("Success", "Updated!!")

        except Exception as ex:
            messagebox.showerror("Update Error", ex)

# Connect to the database
connection = pymysql.connect(host='localhost',
                             user='root',
                             password='admin',
                             db='pythonlab')

DBUI(connection)
```

q28.py

```
"""
Write a Client/Server Socket program to demonstrate the file transfer operation
using Python Programming.
"""

# client

import socket
import sys


HOST = socket.gethostname()
PORT = 9999

soc = socket.socket(socket.AF_INET,   socket.SOCK_STREAM)
soc.connect((HOST, PORT))
print("[+] Connected with Server")

# get file name to send
file_name = input("Enter file name to send : ")
# open file
try:
    file = open(file_name, "rb")
    # send file
    print("[+] Sending file...")
    data = file.read()
    soc.sendall(data)

    # close connection
    soc.close()
    file.close()
    print("[-] Disconnected")

except FileNotFoundError:
    print("Error : file not found")
```

```
# server

import socket
import sys


HOST = socket.gethostname()
PORT = 9999

soc = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
soc.bind((HOST, PORT))
soc.listen(5)

print("Listening ...")

while True:
    conn, addr = soc.accept()
    print("[+] Client connected: ", addr)

    # content will be put in recv.txt
    file = open("recv.txt", "wb")
    while True:
        # get file bytes
        data = conn.recv(4096)
        if not data:
            break
        # write bytes on file
        file.write(data)
    file.close()
    print("[+] Download complete!")

    # close connection
    conn.close()
    print("[-] Client disconnected")
```

q29.py

```
"""
Write a CGI script to demonstrate the concept of check boxes.
"""

<html>

    <body>
        <h1>Select your some subject you hate : </h1>
        <form action = "/cgi-bin/test.py" method = "post" target = "_blank">
            <input type = "checkbox" value = "on" name = "SE" /> Software Engineering
            <input type = "checkbox" value = "on" name = "LD" /> Logic Design
            <input type = "checkbox" value = "on" name = "SA" /> Software Architecture
            <input type = "submit" value = "Select Subject" />
            </form>
    </body>
</html>
```

```
#!/usr/bin/python

# Import modules for CGI handling
import cgi, cgitb

# Create instance of FieldStorage
form = cgi.FieldStorage()
subjects = "<br/>"

# Get data from fields
if form.getvalue('SE'):
   subjects += "Software Engineering <br/>"

if form.getvalue('SA'):
   subjects += "Software Architecture <br/>"

if form.getvalue('LD'):
   subjects += "Logic Design <br/>"


print ("Content-type:text/html")
print ('')
print ("<html>")
print ("<head>")
print ("<title>Radio - Fourth CGI Program</title>")
print ("</head>")
print ("<body>")
print ("<h2> Subjects you hate: <h2>")
print ("<h3>" + subjects + " </h3>")
print ("</body>")
print ("</html>")
```

q30.py

```
"""
Write a CGI script to demonstrate the concept of radio buttons.
"""
<html>

    <body>
        <h1>Select your a subject you hate : </h1>
        <form action = "/cgi-bin/test.py" method = "post" target = "_blank">
            <input type = "radio" name = "subject" value = "Math" /> Maths
            <input type = "radio" name = "subject" value = "Physics" /> Physics
            <input type = "radio" name = "subject" value = "Software Engineering" /> Software Engineering
            <input type = "radio" name = "subject" value = "Logic Design" /> Logic Design
            <input type = "radio" name = "subject" value = "Software Testing" /> Software Testing
            <input type = "radio" name = "subject" value = "Object Oriented Design" /> Object Oriented Design
            <input type = "submit" value = "Select Subject" />
            </form>
    </body>
</html>
```

```
#!/usr/bin/python

# Import modules for CGI handling
import cgi, cgitb

# Create instance of FieldStorage
form = cgi.FieldStorage()

# Get data from fields
if form.getvalue('subject'):
   subject = form.getvalue('subject')
else:
   subject = "Not set"


print ("Content-type:text/html")
print ('')
print ("<html>")
print ("<head>")
print ("<title>Radio - Fourth CGI Program</title>")
print ("</head>")
```

```
print ("<body>")
print ("<h2> The subject you hate is " + subject + " </h2>")
print ("</body>")
print ("</html>")
```