1. Program to create a class Employee with empno, name, depname, designation, age and salary and perform the following function.
i) Accept details of N employees.
ii) Search given employee using empno.
iii) Display employee details in neat format.

Code :

```
class Employee:
    def __init__(self, empno, name, depname, designation, age, salary):
        self.empno = empno
        self.name = name
        self.depname = depname
        self.designation = designation
        self.age = age
        self.salary = salary

    def display_employee(self):
        print("Employee no.:", self.empno)
        print("Employee name:", self.name)
        print("Department name:", self.depname)
        print("Designation:", self.designation)
        print("Employee's age:", self.age)
        print("Employee's salary:", self.salary)

def main():
    n = int(input("Enter the number of employees: "))

    if n <= 0:
        print("At least one employee is needed. Exiting...!")
        return

    employees = []
    for i in range(n):
        print(f"\nEnter the details of employee {i+1}:")
        empno = int(input("Enter employee number: "))
        name = input("Enter employee name: ")
        depname = input("Enter department name: ")
        designation = input("Enter designation: ")
        age = int(input("Enter employee age: "))
        salary = float(input("Enter employee's salary: "))

        employee_obj = Employee(empno, name, depname, designation, age, salary)
        employees.append(employee_obj)

    while True:
        empno = int(input("\nEnter the employee number you want to search: "))
        found = False
        for employee_obj in employees:
            if employee_obj.empno == empno:
                found = True
                employee_obj.display_employee()
```

-ABR

```
                break

        if not found:
            print("Employee not found...")

        opt = input("\nDo you want to continue searching (Y/N)? ")
        if opt.lower() == "n":
            print("Exiting the program...")
            break

if __name__ == '__main__':
    main()
```

-ABR

2. Write a program menu driven to create a BankAccount class. class should support the following methods for i) Deposit ii) Withdraw iii) GetBalanace . Create a subclass SavingsAccount class that behaves just like a BankAccount, but also has an interest rate and a method that increases the balance by the appropriate amount of interest.

Code :

```
class BankAccount:
    def __init__(self, account_num, bal=0):
        self.account_num = account_num
        self.bal = bal

    def deposit(self, amt):
        self.bal += amt
        print(f"Deposit {amt}. New balance is {self.bal}.")

    def withdraw(self, amt):
        if amt > self.bal or amt < 0:
            print("Insufficient funds.")
        else:
            self.bal -= amt
            print(f"Withdraw {amt}. New balance is {self.bal}.")

    def get_balance(self):
        print(f"Balance for account {self.account_num} is {self.bal}.")

class SavingsAccount(BankAccount):
    def __init__(self, account_num, bal=0, interest_rate=0.03):
        super().__init__(account_num, bal)
        self.interest_rate = interest_rate

    def add_interest(self):
        interest = self.bal * self.interest_rate
        self.bal += interest
        print(f"Added {interest} in interest. New balance is {self.bal}.")

def main():
    account_num = input("Enter account number : ")
    account_type = input("Enter account type [b=BankAccount, s=SavingsAccount] : ")

    if account_type == 'b':
        account = BankAccount(account_num)
    elif account_type == 's':
        account = SavingsAccount(account_num)
    else:
        print("Invalid account type.")
        return

    while True:
        print("\n---Menu---")
        print("1. Deposit")
```

-ABR

```
        print("2. Withdraw")
        print("3. Get balance")
        if account_type == 's':
            print("4. Add interest")

        print("0. Exit")
        choice = int(input("Enter choice: "))

        if choice == 1:
            amt = float(input("Enter amount to deposit: "))
            account.deposit(amt)
        elif choice == 2:
            amt = float(input("Enter amount to withdraw: "))
            account.withdraw(amt)
        elif choice == 3:
            account.get_balance()
        elif choice == 4 and account_type == 's':
            account.add_interest()
        elif choice == 0:
            break
        else:
            print("Invalid choice!")

if __name__ == '__main__':
    main()
```

-ABR

3. Create a GUI to input Principal amount, rate of interest and number of years, Calculate Compound interest. When button submit is pressed Compound interest should be displayed in a textbox. When clear button is pressed all contents should be cleared.

Code :

```
import tkinter as tk

def calculate_ci():
    principal = float(principal_entry.get())
    rate = float(rate_entry.get()) / 100
    years = float(years_entry.get())

    amount = principal * (1 + rate) ** years
    ci = amount - principal
    ci_text.set(round(ci, 2))

def clear_entries():
    principal_entry.delete(0, tk.END)
    rate_entry.delete(0, tk.END)
    years_entry.delete(0, tk.END)
    ci_text.set("")

window = tk.Tk()
window.title("Compound Interest Calculator")

# Labels
principal_label = tk.Label(window, text="Principal amount")
principal_label.grid(row=0, column=0, padx=10, pady=5)

rate_label = tk.Label(window, text="Rate of interest")
rate_label.grid(row=1, column=0, padx=10, pady=5)

years_label = tk.Label(window, text="Number of years")
years_label.grid(row=2, column=0, padx=10, pady=5)

ci_label = tk.Label(window, text="Compound interest")
ci_label.grid(row=3, column=0, padx=10, pady=5)

# Entry fields
principal_entry = tk.Entry(window)
principal_entry.grid(row=0, column=1)

rate_entry = tk.Entry(window)
rate_entry.grid(row=1, column=1)

years_entry = tk.Entry(window)
years_entry.grid(row=2, column=1)

ci_text = tk.StringVar()
ci_text.set("")
```

-ABR

```
ci_entry = tk.Entry(window, textvariable=ci_text, state="readonly")
ci_entry.grid(row=3, column=1)
# Buttons
calculate_button = tk.Button(window, text="Calculate", command=calculate_ci)
calculate_button.grid(row=4, column=0, padx=10, pady=5)

clear_button = tk.Button(window, text="Clear", command=clear_entries)
clear_button.grid(row=4, column=1, padx=10, pady=5)

window.mainloop()
```

-ABR

4. Write a GUI program to implement Simple Calculator.

Code :

```
import tkinter as tk

def btn_click(value):
    current = entry.get()
    entry.delete(0, tk.END)
    entry.insert(tk.END, current + value)

def clear():
    entry.delete(0, tk.END)

def equal():
    try:
        result = eval(entry.get())
        entry.delete(0, tk.END)
        entry.insert(tk.END, str(result))
    except Exception as e:
        entry.delete(0, tk.END)
        entry.insert(tk.END, "ERROR")

window = tk.Tk()
window.title("SIMPLE CALCULATOR")

entry = tk.Entry(window, width=40, justify=tk.RIGHT)
entry.grid(row=0, column=0, columnspan=4)

btns = [
    ("7", 1, 0), ("8", 1, 1), ("9", 1, 2), ("/", 1, 3),
    ("4", 2, 0), ("5", 2, 1), ("6", 2, 2), ("*", 2, 3),
    ("1", 3, 0), ("2", 3, 1), ("3", 3, 2), ("-", 3, 3),
    ("%", 4, 0), ("0", 4, 1), (".", 4, 2), ("+", 4, 3)
]

for btn in btns:
    text, row, col = btn
    btn = tk.Button(window, text=text, width=7, height=1, command=lambda text=text:
btn_click(text)) #type in same line.
    btn.grid(row=row, column=col)

clear_btn = tk.Button(window, text="C", width=25, height=1, command=clear)
clear_btn.grid(row=5, column=0, columnspan=3)

equal_btn = tk.Button(window, text="=", width=7, height=1, command=equal)
equal_btn.grid(row=5, column=3)

window.mainloop()
```

-ABR

5. Create a table student table (regno, name and marks in 3 subjects) using MySQL and perform the followings,

a. To accept the details of students and store it in database.

b. To display the details of all the students.

c. Delete particular student record using regno.

Code :

```
import sqlite3

conn = sqlite3.connect('students.db')
cursor = conn.cursor()

cursor.execute('''
    CREATE TABLE IF NOT EXISTS student
    (
        REGNO INT PRIMARY KEY NOT NULL,
        NAME TEXT NOT NULL,
        SUB1_MARKS INT NOT NULL,
        SUB2_MARKS INT NOT NULL,
        SUB3_MARKS INT NOT NULL
    );
    ''')

def insert_student():
    regno = int(input("Enter regno : "))
    name = input("Enter name : ")
    sub1 = int(input("Enter marks in subject 1: "))
    sub2 = int(input("Enter marks in subject 2: "))
    sub3 = int(input("Enter marks in subject 3: "))

    try:
        cursor.execute('INSERT INTO student (REGNO, NAME, SUB1_MARKS, SUB2_MARKS,
SUB3_MARKS) VALUES (?,?,?,?,?)',(regno, name, sub1, sub2, sub3)) #type in same line.
        conn.commit()
        print('Record inserted successfully!')
    except Exception as e:  #sqlite3.IntegrityError alternative
        print('Record with same regno already exists!')

def display_student():
    cursor.execute('SELECT * FROM student')
    rows = cursor.fetchall()

    if len(rows) == 0:
        print('\nNo Record Found!')
    else:
        for row in rows:
            print('REGNO : ', row[0])
            print('NAME : ', row[1])
            print('Sub 1 : ', row[2])
            print('Sub 2 : ', row[3])
```

-ABR

```
            print('Sub 3 : ', row[4])
            print('------------------')

def delete_student():
    regno = int(input("Enter regno of student to delete : "))
    cursor.execute('SELECT * FROM student WHERE REGNO=?', (regno,))
    row = cursor.fetchone()
    if row is None:
        print('No record found.')
    else:
        cursor.execute('DELETE FROM student WHERE REGNO=?', (regno,))
        conn.commit()
        print('Record deleted successfully!')

while True:
    print('\n---------MENU---------')
    print('1. Insert student record')
    print('2. Display records')
    print('3. Delete record')
    print('4. Exit')
    choice = int(input("Enter your choice [1-4] : "))
    if choice == 1:
        insert_student()
    elif choice == 2:
        display_student()
    elif choice == 3:
        delete_student()
    elif choice == 4:
        print('\nExiting...')
        break
    else:
        print('\nInvalid choice! try again...')

conn.close()
```

-ABR

6. Create a table employee (empno, name and salary) using MySQL and perform the followings,
a. To accept the details of employees and store it in database.
b. To display the details of a specific employee.
c. To display employee details whose salary lies within a certain range.

Code :

```
import sqlite3

conn = sqlite3.connect('employees.db')
cursor = conn.cursor()

cursor.execute('''
    CREATE TABLE IF NOT EXISTS employee
    (
        EMPNO INT PRIMARY KEY NOT NULL,
        NAME TEXT NOT NULL,
        SALARY INT NOT NULL
    );
    ''')

def insert_employee():
    empno = int(input('\nEnter employee number: '))
    name = input('Enter name: ')
    salary = int(input('Enter salary: '))

    try:
        cursor.execute('INSERT INTO employee(EMPNO, NAME, SALARY) VALUES (?, ?, ?)',
(empno, name, salary)) #type in same line.
        conn.commit()
        print('Record inserted successfully!')
    except Exception as e:  #sqlite3.IntegrityError alternative
        print('Record with same empno already exists!')


def display_employee():
    empno = int(input('\nEnter empno of the employee: '))

    cursor.execute('SELECT * FROM employee WHERE EMPNO = ?', (empno,))
    row = cursor.fetchone()

    if row is None:
        print('\nNo Record Found!')
    else:
        print('\nEmpno:', row[0])
        print('Name:', row[1])
        print('Salary:', row[2])

def salary_range():
    min_sal = int(input('\nEnter minimum salary: '))
    max_sal = int(input('Enter maximum salary: '))
```

-ABR

```
    cursor.execute('SELECT * FROM employee WHERE SALARY BETWEEN ? AND ?', (min_sal,
max_sal)) #type in same line.
    rows = cursor.fetchall()

    if len(rows) == 0:
        print('No records within salary range.')
    else:
        for row in rows:
            print('Empno:', row[0])
            print('Name:', row[1])
            print('Salary:', row[2])
            print('--------------')

while True:
    print('\n------Menu------')
    print('1. Insert employee record')
    print('2. Display employee record')
    print('3. Display employees within salary range')
    print('4. Exit')

    choice = int(input('Enter your choice [1-4] : '))

    if choice == 1:
        insert_employee()
    elif choice == 2:
        display_employee()
    elif choice == 3:
        salary_range()
    elif choice == 4:
        print('\nExiting.....')
        break
    else:
        print('\nInvalid choice. Please try again.')

conn.close()
```

-ABR

Source Code :

https://drive.google.com/drive/folders/1NqV702WgEiiJA5faXAOfqw-FrHtHZXWD?usp=sharing

-ABR