Create a Python program that accepts two numbers from the user and performs basic arithmetic operations (addition, subtraction, multiplication, division). Display the results with appropriate labels.

1. Use input() to read numbers.
2. Use variables and expressions.
3. Handle division carefully (no division by zero).

Write a Python program that prints a famous quote, ensuring the quote and the author's name are properly formatted with the use of quotation marks and escape characters.

Albert Einstein once said, "Imagination is more important than knowledge."

1. Use proper use of **quotes** and **escape sequences** (\").
2. Use string variables to store the quote and author separately.

Ask the user for a number and determine whether it is **even** or **odd**. Display an appropriate message.

1. Use if-else decision-making statements.
2. Use modulus operator (%) for the check.
3. Print clear output.

Create a Python program that accepts a student's score and prints their grade based on the following:

90 and above: A

80–89: B

70–79: C

60–69: D

Below 60: F

Use if-elif-else statements.

Validate that the score entered is between 0 and 100.

Write a Python program that accepts a string from the user and prints the reversed version of the string.

1. Use slicing (`[::-1]`).

2. Also display the original string length using `len()`.

Ask the user to input a string and check whether the string is a palindrome (reads the same forwards and backwards).

1. Ignore case (use .lower()).

2. Use decision-making (if-else) based on comparison.

Write a Python program that counts the number of words in a user-entered sentence.

1. Use the split() method.

2. Print the total word count.

Create a program that asks the user to input a sentence and a character, then finds and prints:

1. How many times that character appears (count() method).

2. The first position where the character occurs (find() method).

Create a list of 5 numbers entered by the user. Perform the following:

1. Display the list.

2. Add a new number to the list.

3. Sort the list in ascending order.

4. Remove a number entered by the user.

Use append(), sort(), and remove() methods.

Create a tuple with 5 user-entered elements and perform:

1. Find the maximum and minimum values.

2. Find the sum of all elements.

3. Print the second and fourth elements separately.

Use built-in functions like max(), min(), and sum().

Ask the user for 3 keys and their corresponding values to create a dictionary. Then:

1. Display all keys and values separately.
2. Access and print the value for a specific key entered by the user.

Use keys(), values(), and indexing methods.

Create a dictionary to store the student's name, roll number, and marks. Allow the user to:

1. Update marks.
2. Add a new field like 'grade'.
3. Delete the roll number field.

Use dictionary update(), del, and standard dictionary methods.

Create a 1D array of 5 integers entered by the user. Perform the following:

1. Display the array elements.
2. Find and display the sum of all elements.
3. Find the maximum element.

Use array module or lists.

Use simple loops (for).

Create a 2D array (matrix) of size 3x3 where the user inputs the elements. Perform the following:

1. Display the matrix.
2. Calculate the sum of each row and each column separately.

*Use **nested lists**.*

*Use **nested loops**.*

Write a Python program with a function calculate_average(numbers) that accepts a list of numbers and returns the average.

1. Define and call user-defined function.
2. Use input from the user.

Create a recursive function to print the Fibonacci series up to n terms.

1. Define a recursive function fibonacci(n).
2. Use if-else for recursion base condition.

Write a Python program to input a list of numbers from the user and use map() with a lambda function to create a new list containing the squares of the numbers.

1. Use map() and lambda.
2. Print the original and squared lists.

Create a list of numbers. Use the filter() function and a lambda function to extract only the even numbers into a new list.

1. Use filter() and lambda.
2. Print both the original and filtered lists.

Ask the user for a list of numbers. Use the reduce() function along with a lambda function to compute the **sum** of all numbers.

1. Import reduce from functools.
2. Use lambda inside reduce().

Write a program to find the maximum number in a list using reduce() and a lambda function (without using built-in max()).

1. Use reduce() and lambda.
2. Print the maximum number found.

Write a Python class Book with attributes title, author, and price.

1. Create a constructor to initialize these values.
2. Write a method display_details() to display book information.

Create at least two book objects and display their details.

Write a class Rectangle with attributes length and breadth.

1. Write a constructor to initialize dimensions.
2. Write methods to calculate and return the **area** and **perimeter** of the rectangle.

Create an object and display area and perimeter.

Create a class BankAccount with attributes account_holder, account_number, and balance.

1. Use a constructor to initialize them.
2. Write methods deposit(amount) and withdraw(amount) to update the balance.
3. Also add a method display_balance() to show current balance.

Create one object and perform deposit and withdrawal operations.

Create a class Employee with attributes name, employee_id, and salary.

1. Write a constructor to initialize these.
2. Add a method show_details() to display employee details.
3. Add a method increment_salary() to increase the salary by a given percentage.

Create at least two employee objects and apply salary increment.

Create a base class *Vehicle* with attributes *brand* and *year*.

Create derived classes *Car* and *Bike* which add specific attributes like *model* for Car and *type* for Bike.

Write methods to display all details.

Use constructors and method overriding.

Create a base class *Shape* with a method *area()*.

Create derived classes *Rectangle* and *Circle* that override the *area()* method to calculate area appropriately.

Use method overriding (polymorphism).

Create a class *Employee* with attributes like *name* and *salary*.

Create two subclasses *Manager* and *Developer* that inherit from *Employee* and have their own extra method *show_role()* that displays their roles.

Demonstrate polymorphism by calling the same method (show_role()) from different objects.

Create a base class *BankAccount* with basic attributes like *account_holder*, *balance*.
Create two subclasses *SavingsAccount* and *CurrentAccount* with methods specific to their type (e.g., interest calculation for savings, overdraft limit for current).
Use inheritance and additional methods in subclasses.

Write a Python program that:
1. Accepts a string input from the user.
2. Writes the string into a text file.
3. Then reads and displays the contents of the file.
4. Handle exceptions like *FileNotFoundError* and *IOError*.

Use *open()*, *write()*, and *read()* functions.
Include *try-except-finally* blocks for error handling.

Write a Python program that:
1. Opens the file in read mode.
2. Counts and prints the total number of words in the file.
3. Handle any file errors using exception handling.

Use file reading methods (`read()` or `readlines()`).

Use `split()` to count words.

Use `try-except` to manage errors properly.

Design a GUI application using Tkinter that:
Accepts two numbers as input.
Displays their sum when a button is clicked.

Use *Entry*, *Label*, and *Button* widgets.
Add basic input validation (e.g., both fields must be filled).

Create a basic login form GUI with:

1. Username and password entry fields.
2. A "Login" button that shows a message like “Login Successful” on clicking.

Use *Label*, *Entry*, and *Button* widgets.

Display feedback using *messagebox*.

Create a Python program to:

1. Connect to an SQLite database.
2. Create a *students* table (if it doesn’t exist).
3. Insert at least 3 student records (name, age, grade).
4. Fetch and display all student records.

Use DDL *(CREATE TABLE)* and DML *(INSERT, SELECT)* commands.

Use sqlite3 module in Python.

Write a Python program that:

1. Creates an employees table with fields: id, name, salary.
2. Inserts at least two employee records.
3. Updates the salary of one employee.
4. Deletes one employee record by name.
5. Displays all remaining records.

Use DDL (CREATE TABLE) and DML (INSERT, UPDATE, DELETE, SELECT).

Use sqlite3 with commit and close statements properly.

Write a program that:

1. Creates a Pandas DataFrame with names and marks of 5 students.
2. Uses NumPy to calculate the average, maximum, and minimum marks.
3. Uses Matplotlib to display a bar chart of student names vs. marks.

Use pandas.DataFrame, numpy.mean(), and matplotlib.pyplot.bar().

Create a program that:

1. Loads monthly sales data into a Pandas DataFrame (e.g., 6 months).

2. Calculates the total and average sales using NumPy.

3. Plots a line graph of month vs. sales using Matplotlib.

Use pandas, numpy, and matplotlib.pyplot.plot().