Easy:

. **Character Counter:** Write a program that takes a string as input and counts the frequency of each character in the string.

. **Number Patterns:** Create a program that prints a pattern of numbers in a triangle shape.

. **Palindrome Check:** Write a function that determines if a given string is a palindrome (reads the same forwards and backwards).

. **Simple Encryption:** Implement a basic encryption algorithm that shifts the characters of a string by a fixed number of positions.

. **Character Frequency Counter:** Write a program that takes a string as input and prints the frequency of each character in the string.

. **Pattern Printing:** Create a program that prints a pyramid pattern of stars.

. **Reverse List:** Write a function to reverse a given list without using the built-in `reverse()` function.

. **Simple Encryption:** Implement a basic encryption algorithm that shifts the letters of a message by a certain number of positions.

Intermediate:

. **Matrix Multiplication:** Write a program that performs matrix multiplication for two given matrices.

. **Text Analyzer:** Create a program that reads a text file,

counts the frequency of each word, and displays the most common words.

. **Binary Search:** Implement a function that performs binary search to find a target element in a sorted list.

. **File Handling - Copy and Merge:** Write a program that copies the contents of one text file into another and then merges the content of two text files into a new file.

. **Matrix Multiplication:** Write a function to multiply two matrices without using any external libraries.

. **Word Palindromes:** Implement a function that takes a list of words as input and returns a list of words that are palindromes.

. **File Word Counter:** Create a program that reads a text file and counts the occurrences of each word without using external libraries.

. **Duplicate Elements:** Write a function that finds and removes duplicates from a given list without using built-in functions like `set()`.

Challenging:

. **Web Scraping Simulation:** Create a program that simulates web scraping by extracting specific information from a provided HTML string.

. **Graph Representation and Traversal:** Implement a program that represents a graph using an adjacency matrix or list and performs depth-first search (DFS) traversal.

. **Object-Oriented Design - Game Simulation:** Design a text-based game using object-oriented programming

principles, with multiple classes representing characters, items, and interactions.

. **Data Encryption:** Develop a program that encrypts and decrypts a message using a more advanced encryption algorithm, such as the Caesar cipher with a randomized key.

. **Simulation - Traffic Lights:** Develop a program that simulates a basic traffic light system with red, yellow, and green lights.

. **Data Encryption:** Implement a more advanced encryption algorithm, such as the Caesar cipher with variable shift.

. **Graph Connectivity:** Write a function that determines if a given undirected graph is connected, without using external libraries.

. **Game of Life:** Implement Conway's Game of Life, a cellular automaton simulation, using a custom implementation of the grid and rules.