**1. Number Guessing Game**

Write a program that:

1. Picks a random number between 1 and 100.
2. Lets the user guess the number.
3. After each guess, the program should say:
   * "Too high!" if the guess is greater than the number.
   * "Too low!" if the guess is less than the number.
4. If the user guesses correctly, print how many attempts it took.  
   **Concepts:** Random numbers, loops, conditional statements.

**2. Student Grading System**

Create a program to handle student scores:

1. Ask the user to enter scores for 5 subjects.
2. Calculate the total and average score.
3. Assign grades based on the average:
   * A: 90-100
   * B: 80-89
   * C: 70-79
   * D: 60-69
   * F: Below 60
4. Display the total, average, and grade.  
   **Concepts:** Input, loops, conditionals, operators.

**3. ATM Simulation**

Simulate an ATM system with these options:

1. Display the user’s balance.
2. Let the user deposit money.
3. Let the user withdraw money (ensure sufficient balance).
4. Exit the program.  
   Use a loop to keep showing the menu until the user exits.  
   **Concepts:** Loops, conditionals, arithmetic operators.

**4. Rock-Paper-Scissors Game**

Create a rock-paper-scissors game where:

1. The user plays against the computer.
2. The program randomly chooses between rock, paper, and scissors for the computer.
3. Display the result of each round (win, lose, draw).
4. Let the user play multiple rounds and display the total score at the end.  
   **Concepts:** Loops, conditionals, random numbers.

**5. Prime Numbers in a Range**

Write a program that:

1. Takes two numbers as input to define a range.
2. Finds and prints all prime numbers in that range.
3. At the end, prints how many primes were found.  
   **Concepts:** Nested loops, conditionals, operators.

**6. Password Validator**

Create a password validation program that:

1. Asks the user to input a password.
2. Validates the password based on these rules:
   * At least 8 characters long.
   * Contains at least one uppercase letter, one lowercase letter, and one number.
3. Keep prompting until the user enters a valid password.
4. Print "Password is valid" when the criteria are met.  
   **Concepts:** Loops, conditionals, string operations.

**7. Shopping Cart System**

Build a simple shopping cart program that:

1. Displays a menu of items with prices.
2. Lets the user add items to the cart by choosing item numbers.
3. Calculates the total price after the user is done shopping.
4. If the user’s total exceeds 500, apply a 10% discount.  
   **Concepts:** Loops, conditionals, arithmetic operations.

**8. Fibonacci Finder**

Write a program that:

1. Takes an integer input n from the user.
2. Generates the first n terms of the Fibonacci sequence.
3. Checks if each term in the sequence is even or odd.
4. Prints the sequence along with whether each number is "even" or "odd."  
   **Concepts:** Loops, conditionals, arithmetic operators.

**9. Matrix Multiplication**

Write a program that:

1. Takes two 2x2 matrices as input from the user.
2. Multiplies them and prints the resulting matrix.
3. Checks if the resulting matrix is symmetric.  
   **Concepts:** Nested loops, operators, multi-dimensional arrays.

**10. Number Analyzer**

Write a program that:

1. Takes a list of numbers as input from the user.
2. Prints the following:
   * Maximum and minimum numbers in the list.
   * The sum of all even numbers in the list.
   * The count of odd numbers in the list.
3. Keep repeating the process until the user chooses to exit.  
   **Concepts:** Lists, loops, conditionals.

**11. FizzBuzz with a Twist**

Print numbers from 1 to 50. For multiples of 3, print "Fizz"; for multiples of 5, print "Buzz"; and for multiples of both, print "FizzBuzz". Instead of just printing, store the results in a list.

**Hints**:

* Use a **for loop** to iterate through the range.
* Use **if-elif-else conditions** to check divisibility.
* Append the results (numbers or strings like "Fizz") to a list.

**Topics**: Loops, conditionals, lists.

**12. Analyzing a List of Numbers**

Write a program to take a list of numbers from the user and:

1. Calculate the sum and average.
2. Find the smallest and largest numbers.
3. Count how many numbers are even and odd.

**Hints**:

* Use a **list** to store numbers.
* Use **for loops** to iterate over the list.
* Use conditions to separate even and odd numbers.

**Topics**: Lists, loops, conditionals, basic math operations.

**13. Palindrome Checker**

Write a program that checks whether a given string is a palindrome (reads the same forwards and backward).

**Hints**:

* Use **string slicing** to reverse the string (string[::-1]).
* Compare the original string with its reverse.
* Use a **function** to perform the check and return True or False.

**Topics**: Strings, slicing, functions.

**14. Grade Book System**

Create a grade book where:

1. The user enters student names and scores (e.g., "Alice: 85").
2. Store the data in a dictionary.
3. Allow the user to retrieve a student’s score or calculate the class average.

**Hints**:

* Use a **dictionary** to store student-score pairs.
* Use loops and conditions to allow retrieval and average calculation.

**Topics**: Dictionaries, loops, conditionals, input handling.

**15. Currency Converter**

Write a program to simulate a **currency converter**:

1. Let the user choose the currency they want to convert from (USD, EUR, INR, etc.).
2. Let the user enter the amount they want to convert.
3. Convert the amount into another currency using fixed exchange rates.
4. Display the result and ask if the user wants to make another conversion.  
   **Concepts:** Loops, conditionals, arithmetic operations, input validation.

**16. Quiz Application**

Create a **quiz application**:

1. Ask the user a series of multiple-choice questions.
2. Track their answers and score.
3. At the end of the quiz, display the total score and percentage.
4. Allow the user to take the quiz again if they wish.  
   **Concepts:** Loops, conditionals, input handling, arrays.

**17. Weather System Simulation**

Create a **weather system simulation**:

1. The program should generate a random temperature (between -10°C and 40°C).
2. Depending on the temperature, display the appropriate weather condition:
   * Below 0°C: Freezing
   * 0°C to 10°C: Cold
   * 11°C to 20°C: Mild
   * 21°C to 30°C: Warm
   * Above 30°C: Hot
3. Allow the user to request multiple random temperatures.  
   **Concepts:** Random numbers, loops, conditionals.

**18. Grocery Store Billing System**

Write a **grocery store billing system**:

1. Display a list of items available for purchase, each with a price.
2. Let the user add items to the shopping cart by entering item numbers.
3. Calculate the total bill and apply a discount (e.g., 5% off if total is above a certain amount).
4. Ask if the user wants to continue shopping.  
   **Concepts:** Loops, conditionals, arrays, arithmetic operations.

**19. Contact Book**

Build a **contact book** system:

1. Allow the user to add contacts with names, phone numbers, and email addresses.
2. Let the user view the list of contacts.
3. Let the user search for contacts by name.
4. Allow editing and deleting of contacts.  
   **Concepts:** Arrays, loops, strings, input handling.

**20. Leap Year Checker**

Write a program to check if a year is a **leap year**:

1. The year is a leap year if:
   * It is divisible by 4.
   * If divisible by 100, it must also be divisible by 400.
2. Ask the user for a year and print whether it’s a leap year or not.  
   **Concepts:** Conditional statements, logical operators.

**21. Sum of Digits**

Write a program that:

1. Takes an integer input from the user.
2. Calculates and prints the sum of its digits.  
   For example, if the user enters 123, the program should output 6 (1 + 2 + 3).  
   **Concepts:** Loops, arithmetic operations.

**22. Odd or Even Numbers in a Range**

Write a program that:

1. Takes two numbers as input to define a range (e.g., 1 to 10).
2. Prints all the **odd numbers** and **even numbers** in that range.
3. At the end, display how many odd and even numbers were found.  
   **Concepts:** Loops, conditionals, arithmetic operations.

**23. Character Frequency Counter**

Write a program that:

1. Takes a string as input.
2. Counts how many times each character appears in the string.
3. Display the character and its frequency.  
   **Concepts:** Strings, loops, conditionals.

**24. Letter Count in a String**

Write a program that:

1. Takes a string as input.
2. Counts how many times the letter **"a"** appears in the string.
3. Print the count of the letter "a".  
   **Concepts:** Strings, loops, conditionals.

**25. Exception Handling in Division**

Write a program that:

1. Takes two numbers as input from the user and divides the first number by the second.
2. Implement exception handling to catch the error when the user tries to divide by zero (zero division error).
3. Display a message like "Cannot divide by zero" in case of an error.  
   **Concepts:** Exception handling (try, except), arithmetic operations.

**26. Rock-Paper-Scissors Game**

Create a rock-paper-scissors game where:

1. The user plays against the computer.
2. The program randomly chooses between rock, paper, and scissors for the computer.
3. Display the result of each round (win, lose, draw).
4. Let the user play multiple rounds and display the total score at the end.  
   **Concepts:** Loops, conditionals, random numbers.

**27. Simple To-Do List**

Write a program that:

1. Lets the user create a to-do list.
2. The user can add, remove, and view tasks.
3. Store the tasks in a file, and each time the program runs, load the tasks from the file.
4. Implement exception handling for cases where the file might be missing.  
   **Concepts:** File handling, exception handling, loops.

**28. File Reader and Writer**

Write a program that:

1. Creates a text file and writes a user-defined string to it.
2. Then, reads the file and prints the content to the console.
3. Implement exception handling for cases where the file may not exist or cannot be opened.  
   **Concepts:** File handling (open, read, write), exception handling.

Challenging question:

**1. Simple Encryption**

Write a program that encrypts a string by shifting each character by a fixed number (shift) in the alphabet. For example, with a shift of 2, "abc" becomes "cde".

**Hints**:

* Use **ASCII values** (ord() and chr()) to shift characters.
* Handle wrap-around for letters near the end of the alphabet (z to a).
* Use a **function** to handle encryption.

**Topics**: Strings, functions, ASCII operations.

**2. Name Formatter**

Create a program that:

1. Takes a full name as input from the user (e.g., "john doe").
2. Format the name into "John Doe" by capitalizing the first letter of each word.
3. Handle cases where the user may have extra spaces between words.  
   **Concepts:** Strings, string methods (strip(), title(), split(), join()).