1) Return Prime Numbers

Write a Python program that accepts a list of integers and returns a new list containing only the prime numbers from the original list. Use list comprehension to achieve this.

Input: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]

Output: [2, 3, 5, 7, 11]

2) Palindrome Checker

Write a Python program to check whether a given tuple is a palindrome.

Input: (1, 2, 3, 2, 1)

Output: The tuple is a palindrome.

Input: (1, 2, 3, 4, 5)

Output: The tuple is not a palindrome.

3) Least Required Coins

Write a Python program that accepts an amount in coins and calculates the minimum number of coins needed to make up that amount using the following denominations:

coins = [25, 10, 5, 2, 1]

Enter the amount: 87

Output: THE MINIMUM COINS NEEDED ARE: 6

4) Hill Number Check

Write a function is_hill_number(n) that checks if a number is a hill number (digits first strictly increase, then strictly decrease).

Input: Enter a number: 12321

Output: Yes

Input: Enter a number: 12345

Output: No

5) Student Grades Analyzer

Design a Python program to manage student grades using lists, tuples, and dictionaries.

Requirements:

- Accept information for multiple students.
- Each student: (name, [marks])
- Store all students in a list of tuples.
- Dictionary mapping student → average score.

Functions:

a. add_student(name, marks_list)

b. calculate_averages()

c. top_student()

d. display_student_info(name)

Program Behavior:

- Add students, calculate averages, display info, find top student.
- Handle errors if student not found.

6) Library Management System Using Classes and Objects

Class Library:

Class Attributes:

- library_name
- total_books

Instance Attributes:

- branch_name
- books

Methods:

- add_book(book_title)
- display_books()

- display_total_books() (class method)

Demonstration:

- Create two branches
- Add books
- Display books of each branch
- Display total books across branches

7) Temperature Class

Implement a class Temperature:

Methods:

a) set_celsius(c)

b) to_fahrenheit()

c) to_kelvin()

d) display()

8) Word Frequency Analyzer with Custom Error Handling

Functions:

- frequency(para) → word count dictionary (raise AttributeError if input not string)
- dict_to_tup() → dictionary → list of tuples
- sort_list() → sort tuples by frequency desc
- retrieve_word(word) → frequency of a word (raise KeyError if missing)

Menu Options:

1. Enter paragraph → compute frequency
2. Convert dict → tuple list & sort
3. Search for frequency of word
4. Exit

Error handling for invalid input and missing words.

9) String Encoding (Run-Length Encoding)

Write encode_string(s) that compresses a string:

Example: "aaabbcddd" → "a3b2c1d3"

10) Recursive Math Operations – Menu Driven

Operations with recursion:

- factorial(n)
- fibonacci(n)
- power(x, y)

Menu Driven:

- Perform operation until user exits.
- Handle invalid inputs.

11) Stack Implementation

Class Stack with methods:

- push(x)
- pop()
- peek()

Menu-driven stack operations.

Don't use built-in .pop() or .append().

12) Singly Linked List Implementation

Class LinkedList with methods:

- insert_end(x)

- delete_end()

- display()

Menu-driven program for insert, delete, display.