Python Programming Tasks

1. Factorial of a Number

Write a method/function factorial(n) that returns the factorial of n.

Accept n as user input and print the result. Use memoization to make the program faster and more efficient.

Input:

N = 5

Output:

120

2. Implementation of List

Create a list of n natural numbers of your choice. Using list comprehension, create another list that returns the square of the numbers.

Input:

[1, 3, 4, 6, 7]

Output:

[1, 9, 16, 36, 49]

3. 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).

Accept a number from the user and print "Yes" or "No".

Input / Output:

Enter a number: 12321

Output: Yes

Enter a number: 12345

Output: No

Enter a number: 54321

Output: No

4. Anagram Checker

Write a Python function is_anagram(word1, word2) that checks if two words are anagrams.

An anagram is a word or phrase formed by rearranging the letters of another word, using all the original letters exactly once.

Input / Output:

Enter first word: listen

Enter second word: silent

Output: Yes

Enter first word: apple

Enter second word: pale

Output: No

5. Dictionary

Write a Python program to create a dictionary where the keys are student names and the values are lists of their marks in 3 subjects.

Calculate and print the total marks and average marks of each student. Finally, find and display the name of the topper along with their total marks.

Input:

Enter number of students: 3

Enter name of student 1: Aditi

Enter 3 marks for Aditi separated by space: 85 90 88

Enter name of student 2: Rahul

Enter 3 marks for Rahul separated by space: 92 78 84

Enter name of student 3: Meera

Enter 3 marks for Meera separated by space: 75 80 70

Output:

Aditi: Total = 263, Average = 87.67

Rahul: Total = 254, Average = 84.67

Meera: Total = 225, Average = 75.00

Topper: Aditi with 263 marks

6. Fibonacci Series (Recursion)

Write a recursive function fibonacci(n) that returns the nth Fibonacci number.

Accept a number n from the user and print the Fibonacci sequence up to n terms.

Input:

N = 5

Output:

0 1 1 2 3

7. Prime Number Checker

Write a function is_prime(n) to check if a number is prime.

Accept a number from the user and print "Prime" or "Not Prime".

Input: 59

Output: Prime

Input: 80

Output: Not Prime

8. Count Repeating Letters

Accept a word from the user. Create a dictionary with key as the letter and value as the number of occurrences.

Input:

Enter a word: banana

Output:

```
{'b': 1, 'a': 3, 'n': 2}
```

9. Matrix Multiplication

Write a function multiply_matrices(A, B) that multiplies two matrices.

Accept two matrices from the user (as lists of lists) and print the resulting matrix.

Input:

```
Enter number of rows of matrix A: 2
Enter number of columns of matrix A: 2
Enter elements of matrix A row-wise:
1 2
3 4
Enter number of rows of matrix B: 2
Enter number of columns of matrix B: 2
Enter elements of matrix B row-wise:
5 6
7 8
```

Output:

```
Resultant Matrix after multiplication:
[19, 22]
[43, 50]
```

10. String Encoding (Run-Length Encoding)

Write a function encode_string(s) that compresses a string (e.g., "aaabbcddd" → "a3b2c1d3").

Accept a string from the user and print the encoded version.

11. Set Operations

Accept two sets of numbers from the user.

Print their union, intersection, and difference.

Input:

Set1: {1,2,3,4}

Set2: {3,4,5,6}

Output:

Union: {1,2,3,4,5,6}

Intersection: {3,4}

Difference (Set1-Set2): {1,2}

12. Sort Words Alphabetically

Accept a sentence from the user, split it into words, and print the words sorted alphabetically.

Input:

"banana apple cherry"

Output:

['apple', 'banana', 'cherry']

13. Stack Implementation

Implement a class Stack with methods:

push(x) → adds an element

pop() → removes the top element

peek() → shows the top element

Accept user input to perform stack operations and print results.

Do not use the inbuilt functions .pop and .push

14. Singly Linked List Implementation

Implement a class LinkedList with methods:

insert_end(x) → inserts a node at the end

delete_end() → deletes a node from the end

display() → prints the list

Accept user input to insert and delete values, and display the linked list.