🧰 Module 4: Functions and Functional Programming – Practice Problems

---

🧱 Section 1: Defining and Calling Functions

1. Write a function to print "Hello, World!".
2. Write a function that returns the square of a number.
3. Create a function that adds two numbers.
4. Write a function to check if a number is even.
5. Function to return the absolute value of a number.
6. Function to return the maximum of two numbers.
7. Function to return the sum of elements in a list.
8. Function to calculate the average of a list.
9. Function that accepts a string and returns it reversed.
10. Function that counts vowels in a string.

🎯 Section 2: Function Parameters & Return Values

11. Function with default parameters (e.g., greet user).
12. Function with keyword arguments.
13. Write a function that takes any number of arguments using *args and returns their sum.
14. Function using **kwargs to print key-value pairs.
15. Function that accepts both *args and **kwargs.
16. Create a function that returns the factorial of a number.
17. Create a function to check if a string is a palindrome.
18. Function to calculate the compound interest.
19. Create a function to calculate BMI and return category.
20. Function to convert Celsius to Fahrenheit.

🔁 Section 3: Looping Inside Functions

21. Function that returns all prime numbers up to n.
22. Function that finds all even numbers in a list.
23. Function that removes all duplicates from a list.
24. Write a function that returns a list of squares from 1 to n.

25. Function that returns the first n Fibonacci numbers.
26. Function that counts occurrences of each character in a string.
27. Function that finds common elements between two lists.
28. Function that returns the longest word in a sentence.
29. Function that checks if two strings are anagrams.
30. Function that returns the sum of digits of a number.

🧮 Section 4: Functional Programming with lambda, map, filter, reduce

🔹 lambda

31. Write a lambda function to multiply two numbers.
32. Use lambda to get square of a number.
33. Use lambda to check if a number is even.
34. Create a list of cubes using map and lambda.
35. Sort a list of tuples by second element using lambda.

🔹 map()

36. Use map() to convert a list of strings to uppercase.
37. Use map() to round off a list of floats.
38. Use map() to add two lists element-wise.
39. Use map() to convert a list of temperatures from C to F.
40. Use map() with a user-defined function.

🔹 filter()

41. Use filter() to get even numbers from a list.
42. Use filter() to remove empty strings from a list.
43. Use filter() to get elements greater than 50.
44. Use filter() to extract palindromes from a list.
45. Use filter() to keep names starting with "A".

🔹 reduce()

46. Use reduce() to find the product of a list.
47. Use reduce() to find the maximum element.
48. Use reduce() to concatenate strings in a list.
49. Use reduce() to compute GCD of a list of numbers.

50. Use reduce() to compute factorial.

🔸 Don’t forget to from functools import reduce

🧠 Section 5: Recursion

51. Recursive function to find factorial of a number.
52. Recursive function to print numbers from n to 1.
53. Recursive function to compute nth Fibonacci number.
54. Recursive function to sum all elements in a list.
55. Recursive function to reverse a string.
56. Recursive function to find GCD of two numbers.
57. Recursive function to compute power x^n.
58. Recursive function to count digits in a number.
59. Recursive function to compute binary representation of a number.
60. Recursive function to flatten a nested list.

🔄 Section 6: Higher-Order Functions (Functions as Arguments / Return Values)

61. Write a function that accepts another function and applies it twice.
62. Write a decorator-like function that logs the result of another function.
63. Function that returns another function which multiplies input by n.
64. Write a function that applies a list of functions to a single input.
65. Use map() with a list of functions on a single value.

🧪 Section 7: Real-World Practice Tasks

66. Function to validate an email address (basic check).
67. Function to simulate a login system (username/password check).
68. Function to generate random OTP.
69. Function to count frequency of words in a string.
70. Function to generate a password from name and DOB.

🔐 Section 8: Function Scope and Closures

71. Demonstrate local and global variables.

72. Function that modifies a global variable.
73. Write a closure to remember last n inputs.
74. Function factory: make_multiplier(n) that returns a multiplier function.
75. Use nonlocal to track state in nested functions.

🧹 Section 9: String Processing Functions

76. Function that removes all vowels from a string.
77. Function that capitalizes each word in a sentence.
78. Function that finds the first non-repeating character.
79. Function that replaces spaces with hyphens.
80. Function that checks for balanced parentheses.

📊 Section 10: List, Tuple, Dict Processing in Functions

81. Function to merge two dictionaries.
82. Function that returns keys of a dictionary with value > 100.
83. Function to group elements of a list by their length.
84. Function that zips two lists into a dictionary.
85. Function that unpacks a list of tuples and returns two separate lists.

🔄 Section 11: Conversion Utilities

86. Convert list of strings to integers.
87. Convert a dictionary to a list of tuples.
88. Convert seconds to hh:mm:ss format.
89. Function that converts binary string to decimal.
90. Function that converts snake_case to camelCase.

🎮 Section 12: Mini Challenges

91. Build a calculator using functions (add, sub, mul, div).
92. Build a quiz app using functions.
93. Implement Rock-Paper-Scissors using functions.
94. Function to simulate a dice roll n times and return frequency.

95. Create a basic math game (random questions).

🧠 Section 13: Function-Based Interview-Style Problems

96. Implement is_prime(n) using a function.
97. Implement next_prime(n) to find the next prime number.
98. Implement is_armstrong(n) function.
99. Implement is_perfect(n) function.
100. Implement a custom map() function from scratch.
101. Implement a custom filter() function from scratch.
102. Write a function that calculates the Levenshtein distance between two strings.
103. Write a function that counts how many function calls it received (using closure).
104. Write a function that takes a list of numbers and returns mean, median, mode.
105. Create a decorator that measures execution time of a function.