Unit 05 Functions

Defining Your Own Functions:

1. **Basic Function:**
    - Write a Python function named **add_numbers** that takes two parameters (a and b) and returns their sum.
2. **String Manipulation Function:**
    - Create a function called **reverse_string** that takes a string as input and returns the reversed version of that string.
3. **List Processing Function:**
    - Define a function called **find_max** that takes a list of numbers as input and returns the maximum value in the list.
4. **Default Parameter:**
    - Write a function named **greet_user** that takes a user's name as a parameter and prints a personalized greeting. If the name is not provided, the function should default to greeting "Guest."
5. **Variable Number of Parameters:**
    - Create a function called **calculate_average** that can take a variable number of parameters and returns the average of all the numbers passed.

Function Documentation:

6. **Documenting a Simple Function:**
    - Write a function called **multiply** that multiplies two numbers. Include clear and concise documentation for the function.
7. **Documenting a Complex Function:**
    - Create a function named **calculate_discount** that calculates the discounted price of an item based on the original price and a discount percentage. Provide detailed documentation explaining the input parameters and the calculation process.

Keyword and Optional Parameters:

8. **Greeting Function with Defaults:**
    - Write a Python function called **greet** that takes a person's name and age as parameters, with a default age of 25. The function should print a greeting message including the person's name and age.
9. **Rectangle Area with Keyword Parameters:**
    - Define a function **calculate_area** that calculates the area of a rectangle. The function should take length and width as parameters, but allow them to be provided as keyword arguments. If not provided, assume length is 5 and width is 3.

Passing Collections to a Function:

10. **Sum of List Elements:**

- Write a function **sum_list** that takes a list of numbers as a parameter and returns the sum of all elements in the list.

11. **Concatenate Strings in a Tuple:**

- Create a function called **concatenate_strings** that takes a tuple of strings as a parameter and returns a single string by concatenating all the strings.

Variable Number of Arguments:

12. **Calculate Average with Variable Arguments:**
    • Define a function named **average** that can take a variable number of arguments and returns the average of those numbers.

13. **Join Strings with Separator:**
    • Write a function called **join_with_separator** that takes a separator and any number of strings as parameters. It should return a single string formed by joining the input strings with the specified separator.

Scope:

14. **Global and Local Variables:**

- Create a function **scope_example** that uses both a global variable and a local variable. Print the values of both inside and outside the function to demonstrate scope.

15. **Modify List Inside Function:**

- Write a function called **modify_list** that takes a list as a parameter and modifies it by appending the string "modified" to the end. Print the modified list inside and outside the function to illustrate scope.

Functions as "First Class Citizens":

16. **Higher-Order Function with Function Argument:**

- Write a function named **apply_operation** that takes two numbers and a function as parameters. Apply the given function to the numbers and return the result.

17. **Function in a List:**

- Create a list of functions, each performing a different mathematical operation (addition, subtraction, multiplication, etc.). Write a function that takes two numbers and a function index, applies the selected function, and returns the result.

Passing Functions to a Function:

18. **Function Composition:**

- Define a function **compose** that takes two functions as parameters and returns a new function that is their composition. For example, if f(x) returns x + 1 and g(x) returns 2 * x, then **compose(f, g)(3)** should return 7.

19. **Filter Using a Predicate Function:**

- Write a function named **filter_numbers** that takes a list of numbers and a predicate function as parameters. Return a new list containing only the numbers that satisfy the predicate.

map and filter:

20. **Square Each Element Using map:**

- Use the map function to square each element in a given list and return the resulting list.

21. **Filter Odd Numbers Using filter:**

- Use the filter function to filter out the odd numbers from a given list and return the resulting list.

Mapping Functions in a Dictionary:

22. **Dictionary of Operations:**

- Create a dictionary where keys are operation names ("add", "subtract", etc.) and values are the corresponding functions. Write a function that takes two numbers and an operation name, applies the selected operation, and returns the result.

Lambda Expressions:

23. **Use Lambda for List Transformation:**

- Use a lambda expression with the map function to transform a list of strings into a list of their lengths.

Inner Functions and Closures:

24. **Outer and Inner Functions:**

- Define an outer function that takes a parameter and an inner function. The inner function should use the parameter of the outer function. Demonstrate that the inner function has access to the parameter even after the outer function has finished execution.

25. **Closure for Counter:**

- Write a closure function that acts as a counter. The function should return a new count each time it is called.