Phase-1 Python Practical Test

Exercise 1 - User Input, Data Types & Control Flow

Problem

Write a program that:

1. Takes a number n from the user
2. If n is:
    - divisible by **3 and 5** → print "FizzBuzz"
    - divisible by **3 only** → print "Fizz"
    - divisible by **5 only** → print "Buzz"
    - else → print the number itself
3. Repeat this for all numbers from 1 to n

Must Use

- input()
- int
- if / elif / else
- for loop
- % operator

Exercise 2 - Data Structures & Iterations

Problem

Write a program that:

1. Takes **comma-separated numbers** from user
   Example: 10,20,30,40
2. Stores them in a **list**
3. Create:
    - a **tuple** of the same numbers
    - a **set** from the list
4. Print:

    - Sum of numbers
    - Unique numbers
    - Sorted list (ascending)

Must Use

- String operations
- list, tuple, set
- Looping
- Type casting
- Basic operators

Exercise 3 – Functions, *args, kwargs

Problem

Create a function calculate_bill(*prices, **tax) that:

1. Accepts any number of item prices
2. Accepts tax percentage as keyword argument
   Example: tax=18
3. Returns:
    - Total amount
    - Tax amount
    - Final payable amount

Example Call

calculate_bill(100, 250, 300, tax=18)

Must Use

- Function creation
- *args
- **kwargs
- Return multiple values

Exercise 4 - Generators, Lambda & Loops

Problem

1. Write a **generator** even_numbers(n) that yields even numbers up to n
2. Use a **lambda function** to:
    - filter numbers greater than 10
3. Print the final list

Example

Input: 30
Output:

[12, 14, 16, 18, 20, 22, 24, 26, 28, 30]

Must Use

- yield
- Generator function
- lambda
- filter or loop

Exercise 5 - Decorators, Scope & First-Class Functions

Problem

1. Create a decorator execution_logger that:
    - Prints "Starting function"
    - Prints "Ending function"
2. Apply it to a function factorial(n)
3. factorial should:
    - Use recursion
    - Return factorial value

Must Use

- Decorators
- Nested functions
- Scope
- Function as an argument
- Recursion