Training

Python Language Foundations

Syntax and Basics

- **Understand Python syntax, variables, data types, and basic operators.**
  - **Concepts:**
    - Python indentation rules and syntax basics
    - Declaring and assigning variables
    - Built-in data types: int, float, str, bool, NoneType
    - Operators: Arithmetic (+, -, *, /, //, %), comparison (==, !=, >, <), logical (and, or, not)
  - **Assignments:**
    - Write a program to calculate the area and perimeter of different geometric shapes(square, rectangle, parallelogram, triangle, circle) based on user input. The program should include error handling for invalid inputs.
    - Develop a program to take user input for multiple numbers, store them in a list, and compute basic statistical metrics like mean, median, mode, and standard deviation.
    - Create a real-world problem-solving program, such as calculating monthly loan payments based on principal, rate of interest, and tenure.

Control Flow

- **Master if-else statements, loops (for, while), and exception handling.**
    - **Concepts:**
        - if-elif-else statements
        - Loops: for loops, while loops
        - break, continue, and pass
        - Exception handling: try, except, else, finally
    - **Assignments:**
        - Write a program to classify numbers as prime, composite, or neither (for negative or zero values). Ensure it handles invalid inputs gracefully.
        - Create a program to implement a simple text-based menu system for a library that allows users to add, remove, search, and list books. Use loops and conditional statements effectively.
        - Write a robust ATM transaction simulator that includes options for checking balances, depositing, withdrawing money, and exiting. Handle invalid inputs and edge cases.

Functions and Modules

- **Learn how to define functions, use *args and** kwargs, and organize code into modules and packages.**
  - **Concepts:**
    - Defining and calling functions
    - Default arguments and keyword arguments
    - Variable-length arguments (*args and **kwargs)
    - Importing and using modules
    - Organizing code into reusable packages
    - **The `__init__.py` file**: This file is used to mark a directory as a Python package and can contain initialization code for the package. The `__init__.py` file allows you to define what is imported when the package is imported, making it essential for package-level setup and organization.
  - **Assignments:**
    - Write a function that accepts *args to calculate the product of all numbers provided. Add error handling to manage non-numeric inputs.
    - Develop a program that splits functionality into three separate modules: one for fetching data (e.g., reading a file), one for processing (e.g., filtering or transforming the data), and one for reporting results (e.g., saving or displaying the output).
    - Create a class-based module `MathOperations` that encapsulates common mathematical functions (like addition, subtraction, matrix multiplication). Use this module in a program to perform operations based on user input.
    - Build a package `UserAuth` containing modules for password hashing, login verification, and user activity logging. Create a script that uses this package to simulate user authentication.

Data Structures

- **Get comfortable with lists, tuples, dictionaries, sets, and their methods.**
  - **Concepts:**
    - Lists: append, pop, slice, sort
    - Tuples: immutability
    - Dictionaries: key-value pairs, accessing values, iterating
    - Sets: unique elements, union, intersection
  - **Assignments:**
    - Create a program to manage a to-do list with options for adding, updating, removing, and viewing tasks. Use lists and dictionaries to store and manipulate data.
    - Write a program to manage a voting system where each voter can vote only once. Use sets to track voters and dictionaries to count votes for candidates.
    - Build a library catalog that supports searching by title, author, or category. Use dictionaries and lists for efficient data organization.

File Handling

- **Learn how to read, write, and manipulate files (text, CSV, JSON).**
    - **Concepts:**
        - File operations: open, read, write, close
        - Working with CSV files using the csv module
        - Handling JSON data with the json module
    - **Assignments:**
        - Write a program to analyze a CSV file containing sales data. Generate a report with total sales, average sales, and the top-selling product.
        - Develop a program to merge multiple text files into one while handling exceptions for missing or unreadable files. Include options to remove duplicate lines.
        - Create a program to process a JSON file, validate its schema, and generate a summary report with key statistics (e.g., number of entries, missing fields).

Object-Oriented Programming (OOP) in Python

Classes and Objects

- **Understand class creation, attributes, and methods.**
    - **Concepts:**
        - Creating classes and objects
        - Instance variables and methods
        - Class variables and methods
    - **Assignments:**
        - Design a class `BankAccount` with methods for deposit, withdrawal, and balance check. Include error handling for invalid transactions.
        - Create a class `Inventory` to manage stock levels. Add methods to add, remove, and query items, including checks for insufficient stock.

Inheritance

- **Learn about single and multiple inheritance.**
    - **Concepts:**
        - Single inheritance
        - Multiple inheritance
        - Overriding methods
    - **Assignments:**
        - Extend the `BankAccount` class to create `SavingsAccount` and `CheckingAccount` subclasses. Add unique features such as interest calculation, transaction limits, loan eligibility (based on transaction amounts), reward programs.
        - Create a class hierarchy for employees, with a base class and subclasses for full-time, part-time, and contractor employees. Include shared attributes like name, ID, and salary calculation in the base class. Each subclass should calculate the salary based on its type (full-time, part-time, contractor). Apply tax deductions (e.g., 10%) and Provident Fund (PF) deductions (e.g., 12%) for full-time and part-time employees. For contractors, apply only the

tax deduction and no PF. The final salary after deductions should be returned for each employee type.

Encapsulation and Polymorphism

- **Practice encapsulation and using polymorphic methods.**
  - **Concepts:**
    - Private and protected attributes
    - Getter and setter methods
    - Method overloading and overriding
  - **Assignments:**
    - Create a class with private attributes for sensitive data (e.g., user passwords) and methods for secure access and modification.
    - Implement a polymorphic function to calculate areas of different shapes (e.g., circle, rectangle, triangle) using method overriding.

Magic Methods

- **Explore special methods like `__init__`, `__str__`, `__repr__`, `__eq__`, etc.**
  - **Concepts:**
    - Common dunder methods
    - Customizing object representation
  - **Assignments:**
    - Build a class `Transaction` with dunder methods for comparing, adding, and representing transactions.
    - Implement a class `Book` with methods to sort books by title, author, or year using custom dunder methods.

Python Development Tools and Advanced Topics

Integrated Development Environment (IDE)

- **Learn to use PyCharm and VSCode IDEs.**
    - **Assignment:**
        - Set up a Python project in PyCharm with a virtual environment. Use breakpoints to debug a complex function.
        - Debug a Python script in VSCode that uses multiple modules.

Debugging

- **Learn to use debugger in your IDE.**
    - **Assignment:**
        - Debug a program that calculates the Fibonacci sequence and identify logic errors in specific scenarios.

Testing

- **Write unit tests using unittest and pytest.**
    - **Assignment:**
        - Write unit tests for a function that determines if a number is prime. Add edge case tests for negative numbers and zero.
        - Use pytest to test a module that implements a basic calculator with add, subtract, multiply, and divide functions.

Package Management

- **Get familiar with pip and poetry for installing and managing packages.**
    - **Assignment:**
        - Use pip to install a library and list all installed packages. Create a requirements.txt file for your project.
        - Initialize a new poetry project, add dependencies, and configure scripts for running your program.

Virtual Environments

- **Learn to create and manage virtual environments using venv or virtualenv.**
  - **Assignment:**
    - Create a virtual environment for a project and install the requests package. Write a program that uses this package to fetch and display data from a public API.

Advanced Topics

- **Generators and Iterators, List Comprehensions, Lambda Functions, Decorators**
  - **Assignment:**
    - Write a generator to produce the first 10 Fibonacci numbers. Extend it to handle user-specified ranges.
    - Create a list comprehension to generate a list of squares for even numbers from 1 to 50.
    - Write a decorator to log the execution time of a function that processes large datasets.