Practice 1: Define a Class

Objective: Create a simple class to understand how classes work in Python.

Task:

1. Define a `Book` class with the following attributes:
   - Title
   - Author
   - Published Year
2. Create an instance of the `Book` class and print its details.

Practice 2: Add Methods to a Class

Objective: Learn how to add methods to a class.

Task:

1. Add a method to the `Book` class that returns a summary of the book's details.
2. Call this method from an instance of the `Book` class and print the result.

Practice 3: Encapsulation

Objective: Understand encapsulation by using private attributes and getter/setter methods.

Task:

1. Define a `BankAccount` class with the following private attributes:
   - Account Number
   - Balance
2. Add getter and setter methods to access and modify the balance safely.

Practice 4: Class Method and Static Method

Objective: Learn the difference between class methods and static methods.

Task:

1. Define a `Temperature` class with a method to convert Celsius to Fahrenheit and another method to convert Fahrenheit to Celsius.
2. Use a class method to create an instance from a Fahrenheit temperature.

Practice 5: Inheritance

Objective: Understand inheritance by creating a base class and a derived class.

Task:

1. Define a `Person` class with attributes:
   - Name
   - Age
2. Create a `Student` class that inherits from `Person` and adds a new attribute `Student ID`.

Practice 6: Class Method for Utility Calculation

Objective: Create a class method to perform a utility calculation.

Task:

1. Define a `Calculator` class.
2. Implement a class method `add()` that takes two numbers as arguments and returns their sum.

Practice 7: Class Method for Date Manipulation

Objective: Use a class method to manipulate date objects.

Task:

1. Define a `DateHelper` class.
2. Implement a class method `get_next_day()` that takes a date object and returns the next day's date.

Practice 8: Class Method for Validation

Objective: Implement a class method for data validation.

Task:

1. Define a `Validator` class.
2. Implement a class method `is_valid_email()` that checks if a given string is a valid email address.

Practice 9: Class Method for Configuration

Objective: Use a class method to set configuration parameters.

Task:

1. Define a `Config` class.
2. Implement a class method `set_config()` that sets a class-level configuration parameter.

Practice 10: Class Method for File Handling

Objective: Create a class method to read data from a file.

Task:

1. Define a `FileHandler` class.
2. Implement a class method `read_file()` that reads and returns the contents of a specified file.

Practice 11: Class Method for Database Operations

Objective: Implement a class method to perform database operations.

Task:

1. Define a `DatabaseManager` class.
2. Implement a class method `fetch_data()` that queries a database and returns the result.

Practice 12: Class Method for String Manipulation

Objective: Create a class method to manipulate strings.

Task:

1. Define a `StringUtil` class.
2. Implement a class method `reverse_string()` that takes a string as input and returns its reversed version.

Practice 13: Class Method for Number Processing

Objective: Use a class method to process numerical data.

Task:

1. Define a `MathUtil` class.
2. Implement a class method `is_prime()` that checks if a given number is prime.

Practice 14: Class Method for URL Handling

Objective: Implement a class method to handle URL operations.

Task:

1. Define a `URLHandler` class.
2. Implement a class method `parse_url()` that extracts components (like protocol, domain, and path) from a given URL string.

Practice 15: Class Method for Data Conversion

Objective: Create a class method to convert data between formats.

Task:

1. Define a `Converter` class.
2. Implement a class method `json_to_dict()` that converts a JSON string to a Python dictionary.

Practice 16: Class Method for File Writing

Objective: Use a class method to write data to a file.

Task:

1. Define a `FileWriter` class.
2. Implement a class method `write_to_file()` that writes a given string to a specified file.

Practice 17: Class Method for Environment Information

Objective: Implement a class method to fetch environment information.

Task:

1. Define an `EnvironmentInfo` class.
2. Implement a class method `get_os_info()` that retrieves and returns information about the operating system.