### Problem Statement-1: Using Multiple Functions from a Module

#### Problem Description:

You need to write a Python program that calculates the total payment for employees, including their base salary, bonus, and tax deduction. The solution should use multiple functions and import them from an external module. The program should have the following requirements:

1. **Function 1 (calculate\_bonus)**: This function takes the employee's salary and calculates a 10% bonus.
2. **Function 2 (calculate\_tax)**: This function takes the employee's salary and calculates a 15% tax deduction.
3. **Function 3 (calculate\_total\_payment)**: This function calculates the total payment for an employee. It takes the salary, bonus, and tax as arguments and returns the total payment after adding the bonus and subtracting the tax.

#### Steps:

1. Write a module (payment.py) that contains the three functions:
   * calculate\_bonus(salary)
   * calculate\_tax(salary)
   * calculate\_total\_payment(salary)
2. Write a main program that:
   * Imports the functions from the module payment.py.
   * Accepts an employee's base salary as input.
   * Calls the functions to calculate the bonus, tax, and total payment.
   * Prints the results.

### Problem Statement-2: ****Online Bookstore Management System****

You are tasked with building an **Online Bookstore Management System** that can manage books, customers, and orders. The system will be split into several Python files for modularity. The main goal is to practice working with multiple files and importing functions between them to implement a solution.

### Requirements

The system will include the following components, each in a separate file:

1. **book.py**:
   * **Function 1 (add\_book)**: Add a new book to the bookstore inventory. The function should take parameters like the book's title, author, genre, and price.
   * **Function 2 (list\_books)**: List all the books currently in the inventory. Each book should have a title, author, genre, and price.
2. **customer.py**:
   * **Function 1 (add\_customer)**: Add a new customer to the system. The function should take parameters like the customer's name, email, and phone number.
   * **Function 2 (list\_customers)**: List all customers in the system.
3. **order.py**:
   * **Function 1 (place\_order)**: Create a new order for a customer. This function should take the customer’s ID, the book’s title, and the quantity of the book.
   * **Function 2 (list\_orders)**: List all orders placed by customers.
4. **main.py**:
   * This file will be the user interface where the customer can perform actions like adding books, customers, and placing orders. It will interact with the functions in book.py, customer.py, and order.py.

### Task Breakdown:

1. **book.py**:
   * Implement a function to add a new book to the inventory and store it in a list or dictionary.
   * Implement a function to list all books in the inventory.
2. **customer.py**:
   * Implement a function to add a new customer and store customer data.
   * Implement a function to list all customers.
3. **order.py**:
   * Implement a function to create an order for a customer and store the order details.
   * Implement a function to list all orders.
4. **main.py**:
   * Implement a user interface to interact with the system and allow the user to choose options like:
     + Add a new book
     + List all books
     + Add a new customer
     + List all customers
     + Place a new order
     + List all orders

### Expected Behavior:

1. **Adding a Book**:
   * The user can add a book by entering its title, author, genre, and price.
   * The book should be added to the bookstore inventory.
2. **Listing Books**:
   * The system should display all books in the inventory with their title, author, genre, and price.
3. **Adding a Customer**:
   * The user can add a new customer by entering their name, email, and phone number.
   * The customer should be added to the customer list.
4. **Listing Customers**:
   * The system should display a list of all customers, including their name, email, and phone number.
5. **Placing an Order**:
   * The user can place an order by selecting a customer and a book from the inventory, and specifying the quantity of the book.
   * The order should be stored and linked to the customer.
6. **Listing Orders**:
   * The system should display a list of all orders, showing the customer’s name, book title, and quantity.

### Example

#### Folder Structure:

css

Copy code

online\_bookstore/

├── book.py

├── customer.py

├── order.py

└── main.py