Question Paper Set A1

Subgroup J

J.1 — [S01J1] Document & Improve Order Discount Function

Scenario (e-commerce):
A developer implemented a discount function for orders. The code works but is undocumented and not easily understandable by other team members.

```
def apply_discount(price, discount):
    final = price - (price * discount/100)
    if final < 0:
        final = 0
    return final
```

Your Task:

- Use AI to generate a **docstring (PEP 257 style)**.
- Add inline comments for clarity.
- Use AI to suggest improvements in **type safety** and **input validation** (e.g., handling negative discount or non-numeric inputs).

Data & Edge Cases:

- Discount should be between 0 and 100.
- Price should never be negative.

AI Assistance Expectation:

- Ask AI to propose improved type annotations.
- AI should generate professional, human-readable documentation.

Sample Input

```
apply_discount(200, 10)
```

Sample Output

```
180.0
```

Acceptance Criteria:

- Code logic unchanged (except validation improvements).
- AI-generated documentation is clear and maintainable.
- Inline comments explain "why" not just "what".

Deliverables:

- Documented & refactored function in `.py` file.
- AI prompt + generated documentation snippet.

J.2 — [S01J2] Code Review for User Login Check

Scenario (web platform):
The following function checks whether a user is allowed to log in. It was written quickly and lacks quality and security considerations.

```
def can_login(username, password, db):
    if username in db and db[username] == password:
        return True
    else:
        return False
```

Your Task:

- Perform an **AI-assisted code review**.
- Identify risks:
  - Password stored in plain text.
  - No input validation.
  - Logic readability issues.
- Propose improvements with AI:
  - Use `hashlib` (or other library) for password checks.
  - Improve code readability with early return.
  - Document function clearly.

Data & Edge Cases:

- `username` missing from db.
- Empty strings for credentials.
- Invalid types (`None`, numbers).

AI Assistance Expectation:

- Ask AI to propose review comments.
- Use AI to generate a refactored, documented version.

Sample Input

```
db = {"alice": "secret123"}
can_login("alice", "secret123", db)
```

Sample Output

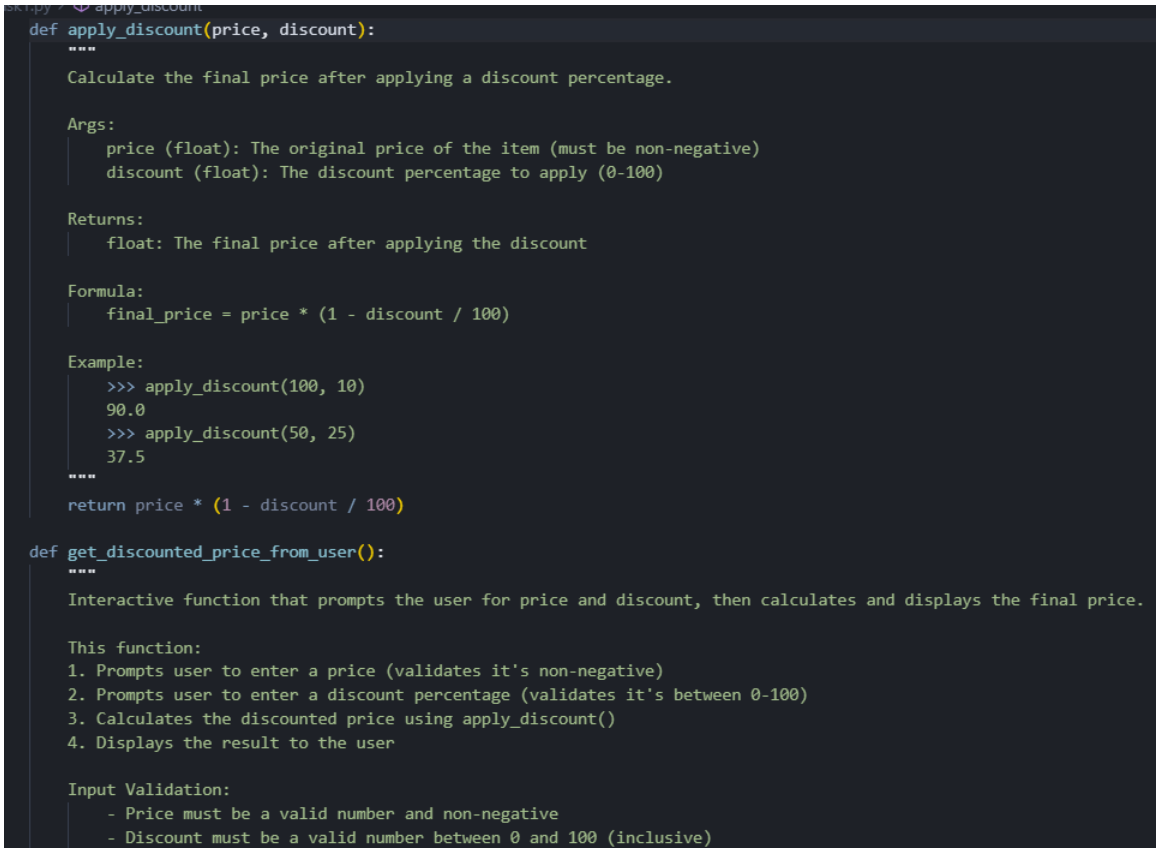
```
True
```

Acceptance Criteria:

- Reviewed code improves readability & security.
- AI-generated documentation added.
- Clear before → after record of review suggestions.

Deliverables:

- Original + reviewed function.
- AI review comments captured (as report).
- Final refactored function with docstring



```python
def apply_discount(price, discount):
    """
    Calculate the final price after applying a discount percentage.

    Args:
        price (float): The original price of the item (must be non-negative)
        discount (float): The discount percentage to apply (0-100)

    Returns:
        float: The final price after applying the discount

    Formula:
        final_price = price * (1 - discount / 100)

    Example:
        >>> apply_discount(100, 10)
        90.0
        >>> apply_discount(50, 25)
        37.5
    """
    return price * (1 - discount / 100)

def get_discounted_price_from_user():
    """
    Interactive function that prompts the user for price and discount, then calculates and displays the final price.

    This function:
    1. Prompts user to enter a price (validates it's non-negative)
    2. Prompts user to enter a discount percentage (validates it's between 0-100)
    3. Calculates the discounted price using apply_discount()
    4. Displays the result to the user

    Input Validation:
        - Price must be a valid number and non-negative
        - Discount must be a valid number between 0 and 100 (inclusive)
```

```
    User Experience:
        - Provides clear error messages for invalid input
        - Continues prompting until valid input is received
        - Shows the final calculated price with discount percentage

    Example Usage:
        When run, the function will prompt:
        Enter the price (must be non-negative): 100
        Enter the discount percentage (0-100): 10
        Final price after 10.0% discount: 90.0
    """
    while True:
        try:
            # Get price input from user with validation
            price = float(input("Enter the price (must be non-negative): "))
            if price < 0:
                print("Price cannot be negative. Please try again.")
                continue

            # Get discount input from user with validation
            discount = float(input("Enter the discount percentage (0-100): "))
            if discount < 0 or discount > 100:
                print("Discount must be between 0 and 100. Please try again.")
                continue

            # If both inputs are valid, break out of the loop
            break

        except ValueError:
            # Handle non-numeric input
            print("Invalid input. Please enter numeric values.")

    # Calculate the discounted price using the helper function
```

```
    discounted_price = apply_discount(price, discount)

    # Display the result to the user
    print(f"Final price after {discount}% discount: {discounted_price}")

# Example usage and main execution
if __name__ == "__main__":
    """
    Main execution block that runs when the script is executed directly.

    This ensures the interactive discount calculator only runs when the script
    is run directly (not when imported as a module).
    """
    print("Starting the discount calculator...")
    get_discounted_price_from_user()
```

OUTPUT:

```
EST-2/Task1.py"
Starting the discount calculator...
Enter the price (must be non-negative): 200
Enter the discount percentage (0-100): 10
```

TASK: 2

```
"""
Simple Login System
"""

def login():
    """
    Interactive login function that prompts user for credentials and validates them.

    This function provides a simple login system that:
    1. Displays a predefined list of allowed users and their passwords
    2. Prompts the user to enter their username and password
    3. Validates the credentials against the allowed users dictionary
    4. Provides feedback on login success or failure

    Features:
        - Simple username/password authentication
        - Interactive user input prompts
        - Clear success/failure messages
        - Predefined user database for testing

    User Database:
        - alice: password123
        - bob: qwerty
        - charlie: letmein

    Returns:
        None: This function only prints messages and doesn't return values

    Example Usage:
        When called, the function will prompt:
        Enter your username: alice
        Enter your password: password123
        Login successful! Welcome, alice
```

```
    Security Note:
        This is a basic implementation for demonstration purposes.
        In production, passwords should be hashed and stored securely.
    """
    # Define allowed users and their passwords
    allowed_users = {
            "alice": "password123",
            "bob": "qwerty",
            "charlie": "letmein"
        }

    # Get username input from user
    username = input("Enter your username: ")

    # Get password input from user
    password = input("Enter your password: ")

    # Validate credentials and provide feedback
    if username in allowed_users and allowed_users[username] == password:
        print("Login successful! Welcome,", username)
    else:
        print("Login failed! Invalid username or password.")

# Call the login function to run the program
if __name__ == "__main__":
    login()
```

OUTPUT:

```
EST-2/Task2.py"
Enter your username: alice
Enter your password: password123
Login successful! Welcome, alice
```