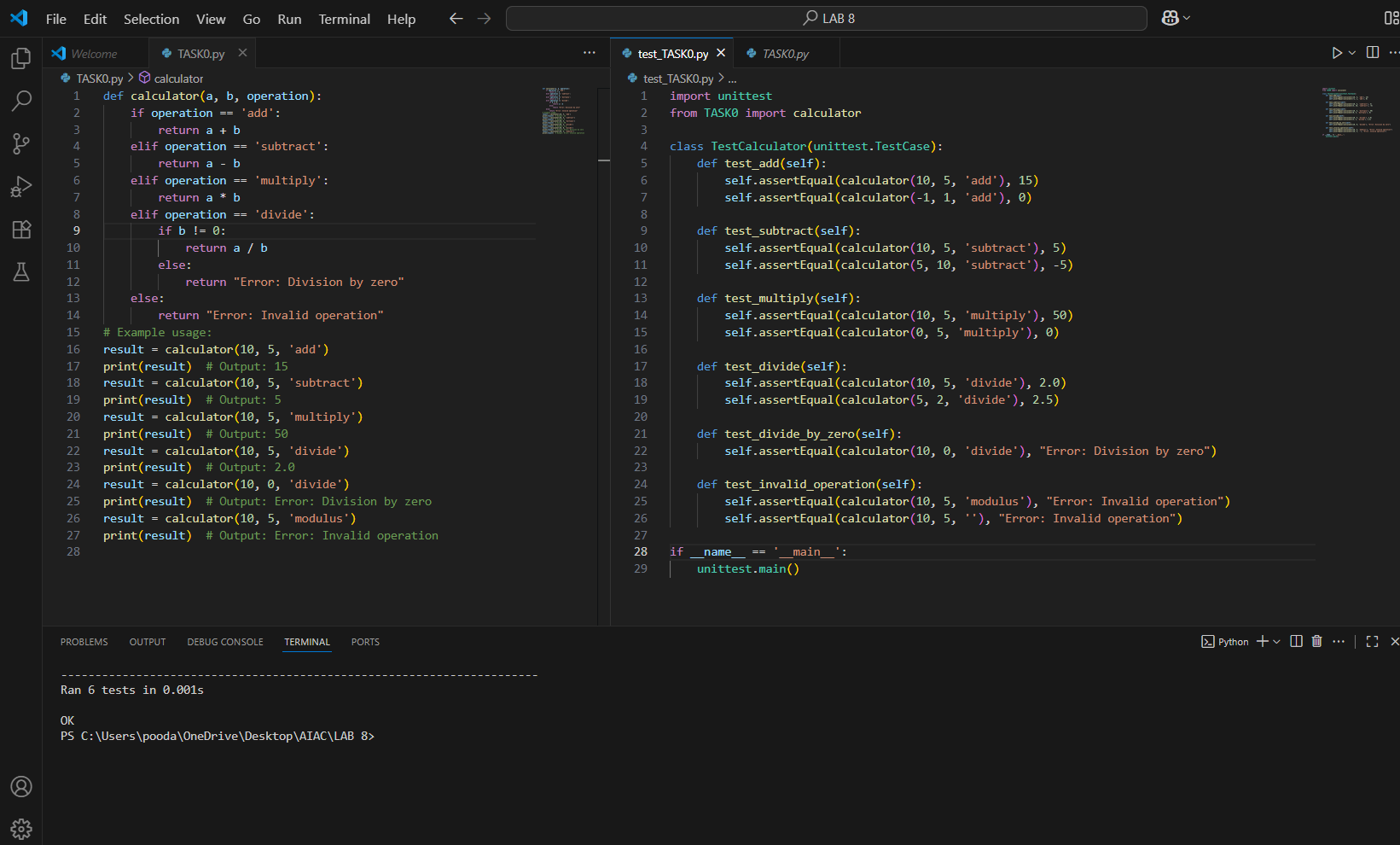
|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **SCHOOL OF COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE** | | | | | **DEPARTMENT OF COMPUTER SCIENCE ENGINEERING** | | | | |
| **ProgramName:**B. Tech | | | | **Assignment Type: Lab** | | | **AcademicYear:**2025-2026 | | |
| **CourseCoordinatorName** | | | | Venkataramana Veeramsetty | | | | | |
| **Instructor(s)Name** | | | | |  | | --- | | Dr. V. Venkataramana (Co-ordinator) | | Dr. T. Sampath Kumar | | Dr. Pramoda Patro | | Dr. Brij Kishor Tiwari | | Dr.J.Ravichander | | Dr. Mohammand Ali Shaik | | Dr. Anirodh Kumar | | Mr. S.Naresh Kumar | | Dr. RAJESH VELPULA | | Mr. Kundhan Kumar | | Ms. Ch.Rajitha | | Mr. M Prakash | | Mr. B.Raju | | Intern 1 (Dharma teja) | | Intern 2 (Sai Prasad) | | Intern 3 (Sowmya) | | NS\_2 ( Mounika) | | | | | | |
| **CourseCode** | | | 24CS002PC215 | **CourseTitle** | | AI Assisted Coding | | | |
| **Year/Sem** | | | II/I | **Regulation** | | R24 | | | |
| **Date and Day**  **of Assignment** | | | Week4 - Wednesday | **Time(s)** | |  | | | |
| **Duration** | | | 2 Hours | **Applicableto**  **Batches** | |  | | | |
| **AssignmentNumber:8.3**(Present assignment number)/**24**(Total number of assignments) | | | | | | | | | |
|  | | | | | | | | | |
|  | | | | | | | | | |
|  | **Q.No.** | **Question** | | | | | | ***ExpectedTime***  ***to complete*** |  |
|  | 1 | Lab 8: Test-Driven Development with AI – Generating and Working with Test Cases  **Lab Objectives:**   * To introduce students to test-driven development (TDD) using AI code generation tools. * To enable the generation of test cases before writing code implementations. * To reinforce the importance of testing, validation, and error handling. * To encourage writing clean and reliable code based on AI-generated test expectations.     **Lab Outcomes (LOs):**  After completing this lab, students will be able to:   * Use AI tools to write test cases for Python functions and classes. * Implement functions based on test cases in a test-first development style. * Use unittest or pytest to validate code correctness. * Analyze the completeness and coverage of AI-generated tests. * Compare AI-generated and manually written test cases for quality and logic   **Task Description#1**  Use AI to generate test cases for is\_valid\_email(email) and then implement the validator function.  **Requirements:**   * Must contain @ and . characters. * Must not start or end with special characters. * Should not allow multiple @.   **Expected Output#1**   * Email validation logic passing all test cases   **Task Description#2 (Loops)**   * Ask AI to generate test cases for assign\_grade(score) function. Handle boundary and invalid inputs.   **Requirements**   * AI should generate test cases for assign\_grade(score) where: 90-100: A, 80-89: B, 70-79: C, 60-69: D, <60: F * Include boundary values and invalid inputs (e.g., -5, 105, "eighty").   **Expected Output#2**  Grade assignment function passing test suite  **Task Description#3**   * Generate test cases using AI for is\_sentence\_palindrome(sentence). Ignore case, punctuation, and spaces   **Requirement**   * Ask AI to create test cases for is\_sentence\_palindrome(sentence)   (ignores case, spaces, and punctuation).   * Example:   "A man a plan a canal Panama" → True  **Expected Output#3**   * Function returns True/False for cleaned sentences * Implement the function to pass AI-generated tests.   **Task Description#4**   * Let AI fix it Prompt AI to generate test cases for a ShoppingCart class (add\_item, remove\_item, total\_cost).   **Methods:**  Add\_item(name,orice)  Remove\_item(name)  Total\_cost()  **Expected Output#4**   * Full class with tested functionalities   **Task Description#5**   * Use AI to write test cases for convert\_date\_format(date\_str) to switch from "YYYY-MM-DD" to "DD-MM-YYYY".   **Example: "2023-10-15" → "15-10-2023"**  **Expected Output#5**   * Function converts input format correctly for all test cases   **Note: Report should be submitted a word document for all tasks in a single document with prompts, comments & code explanation, and output and if required, screenshots**  **Evaluation Criteria:**   | **Criteria** | **Max Marks** | | --- | --- | | Task #1 | 0.5 | | Task #2 | 0.5 | | Task #3 | 0.5 | | Task #4 | 0.5 | | Task #5 | 0.5 | | **Total** | **2.5 Marks** | | | | | | | Week4 - Wednesday |  |

**TASK 0 :**



**Task Description#1**

Use AI to generate test cases for is\_valid\_email(email) and then implement the validator function.

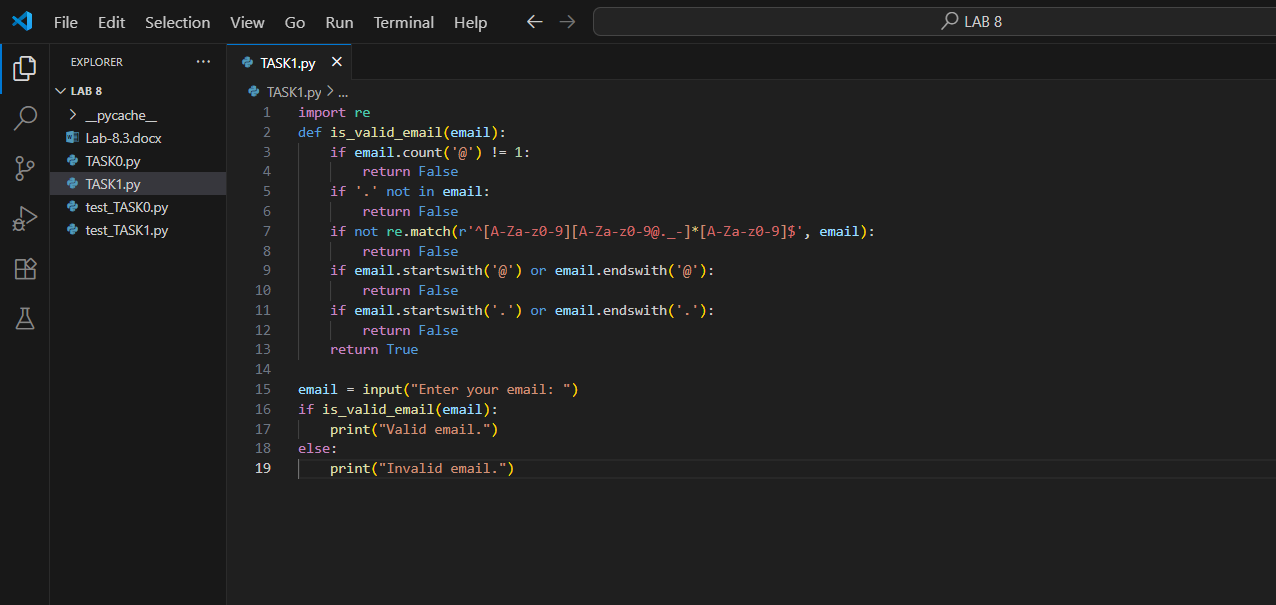
**Requirements:**

* Must contain @ and . characters.
* Must not start or end with special characters.
* Should not allow multiple @.

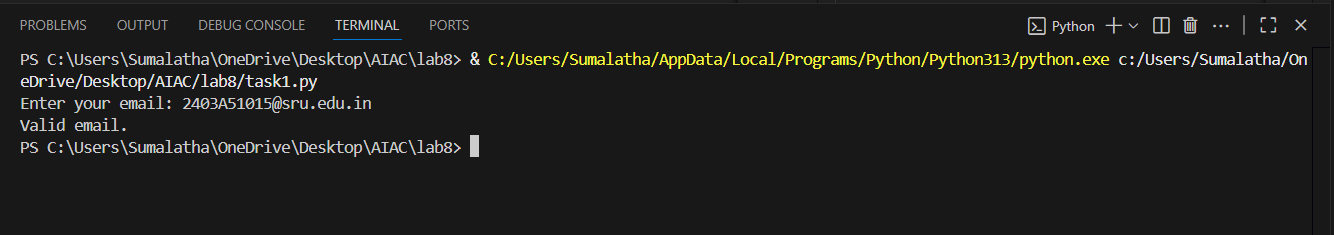
**Expected Output#1**

* Email validation logic passing all test cases

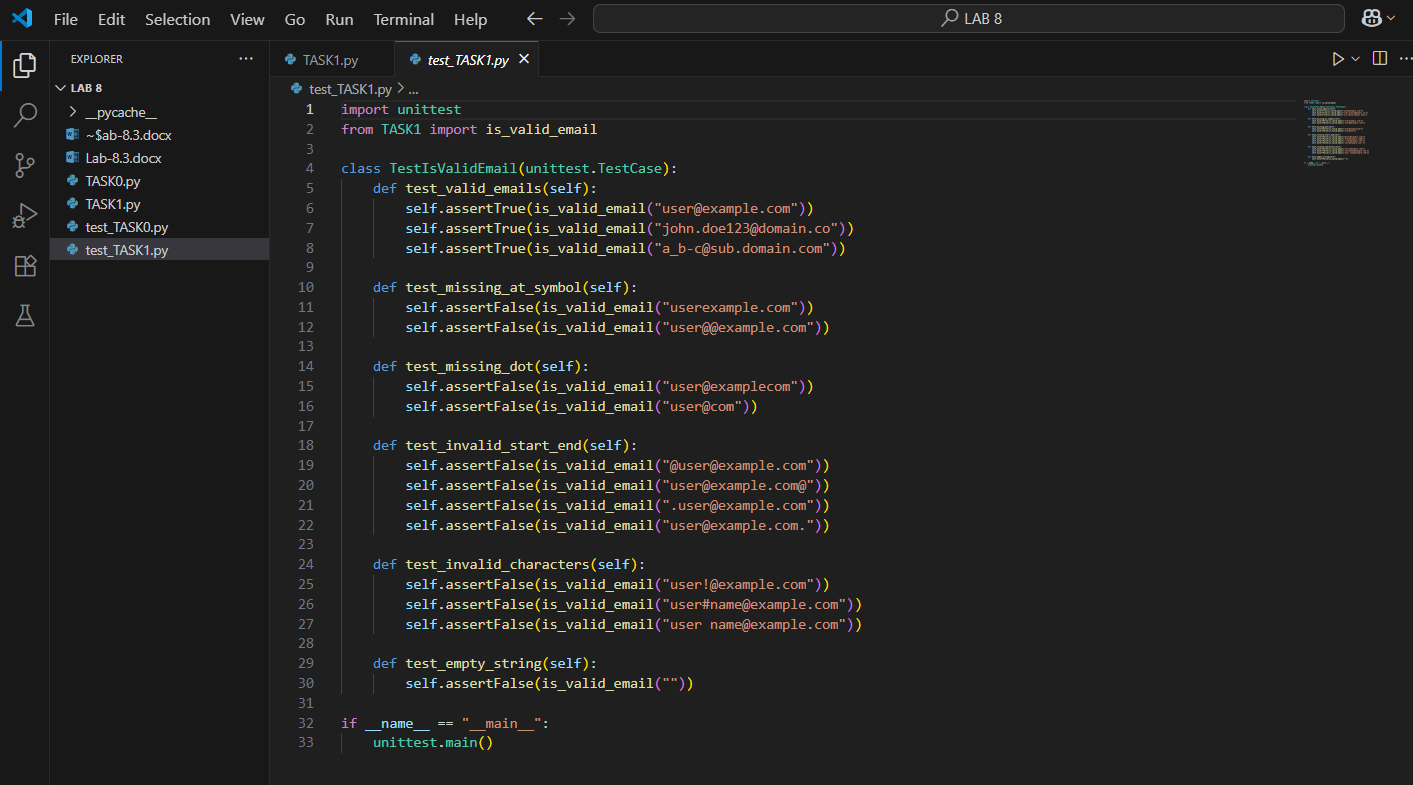
TASK 1:



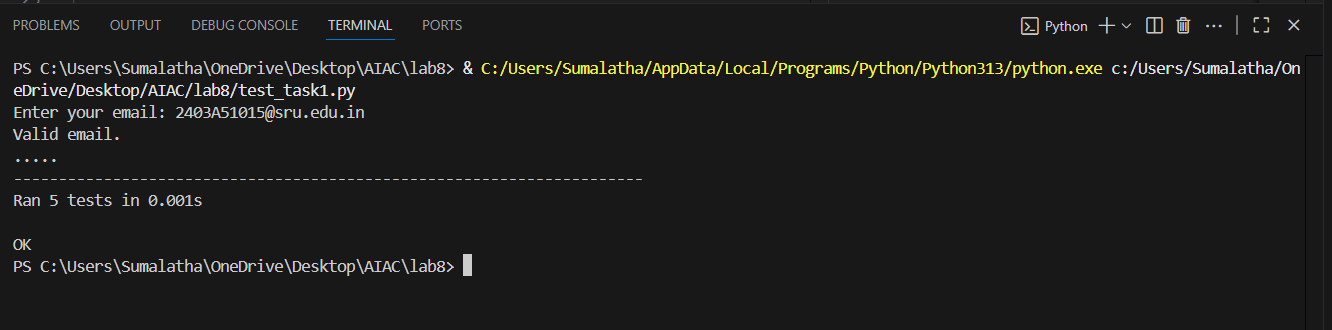
OUTPUT:



TEST CASE:



OUTPUT:



**Task Description#2 (Loops)**

* Ask AI to generate test cases for assign\_grade(score) function. Handle boundary and invalid inputs.

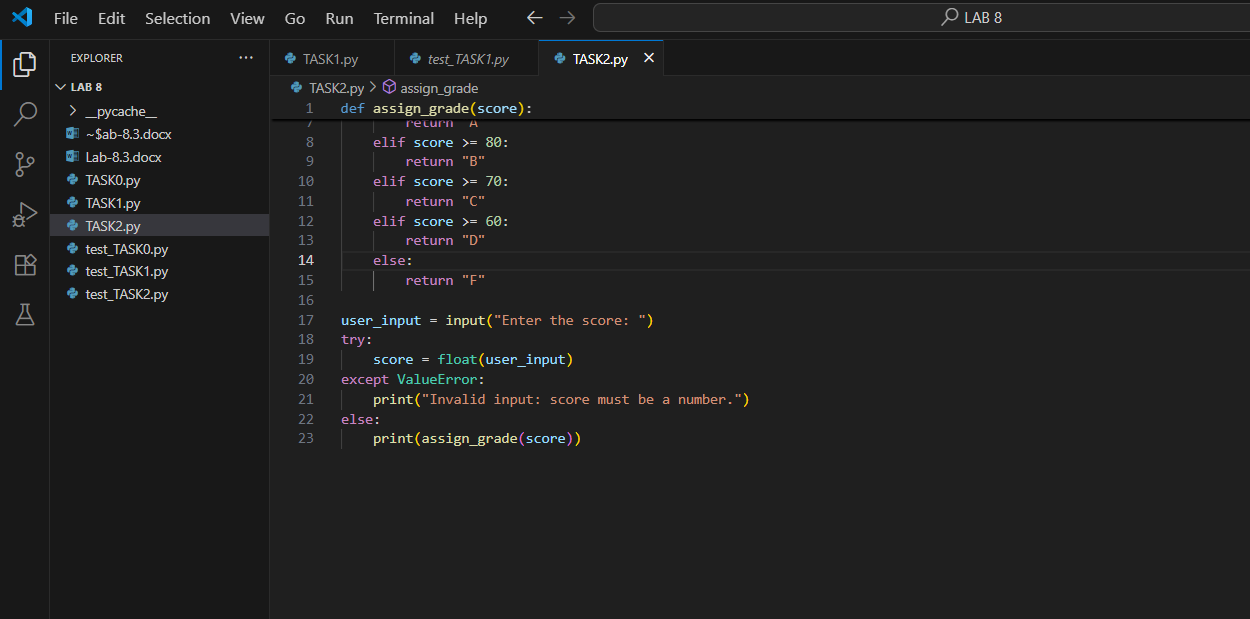
**Requirements**

* AI should generate test cases for assign\_grade(score) where: 90-100: A, 80-89: B, 70-79: C, 60-69: D, <60: F
* Include boundary values and invalid inputs (e.g., -5, 105, "eighty").

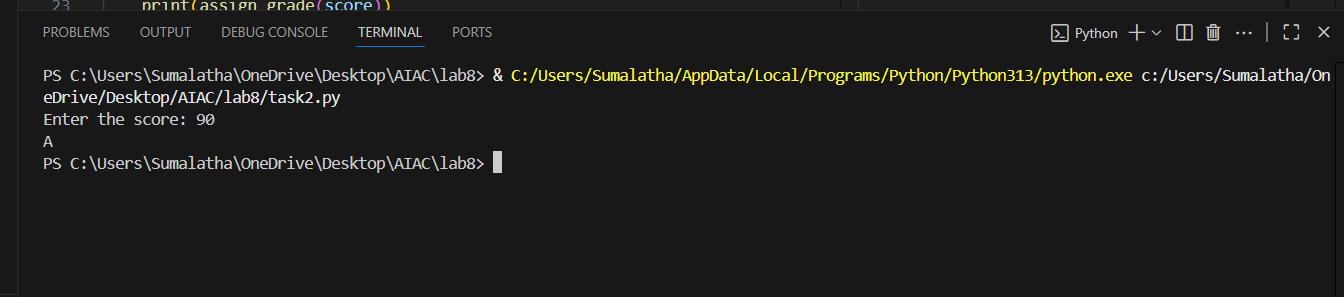
**Expected Output#2**

Grade assignment function passing test suite

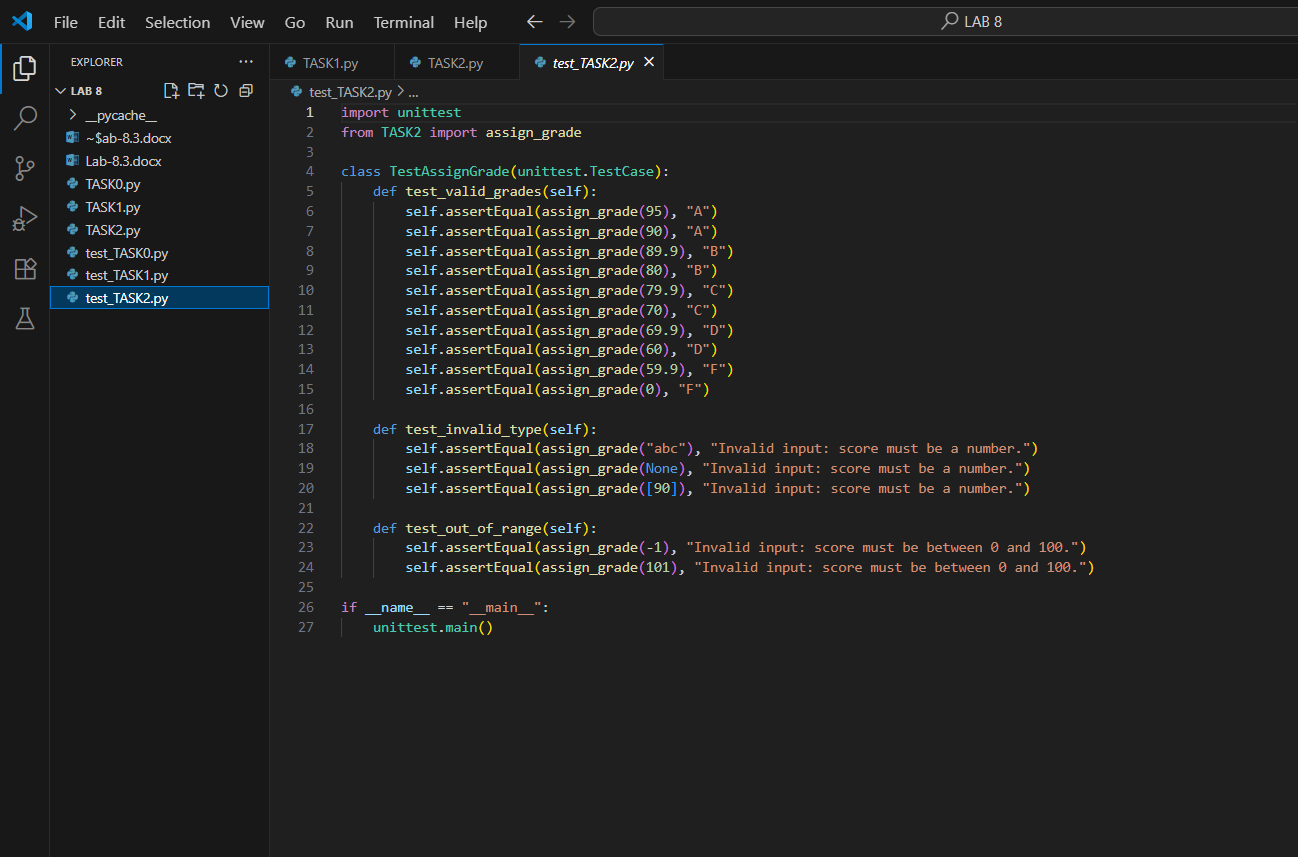
TASK 2:



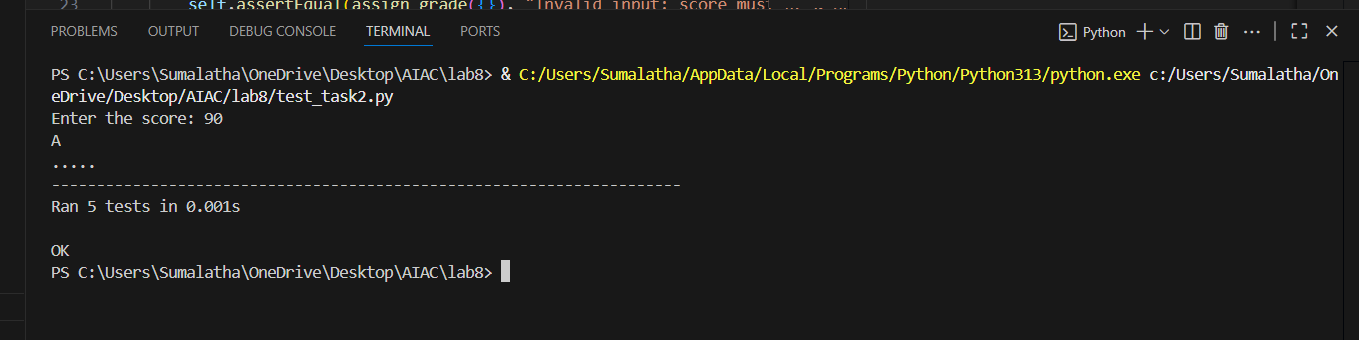
**OUTPUT:**



**TEST CASE ;**



**OUTPUT:**



**Task Description#3**

* Generate test cases using AI for is\_sentence\_palindrome(sentence). Ignore case, punctuation, and spaces

**Requirement**

* Ask AI to create test cases for is\_sentence\_palindrome(sentence)

(ignores case, spaces, and punctuation).

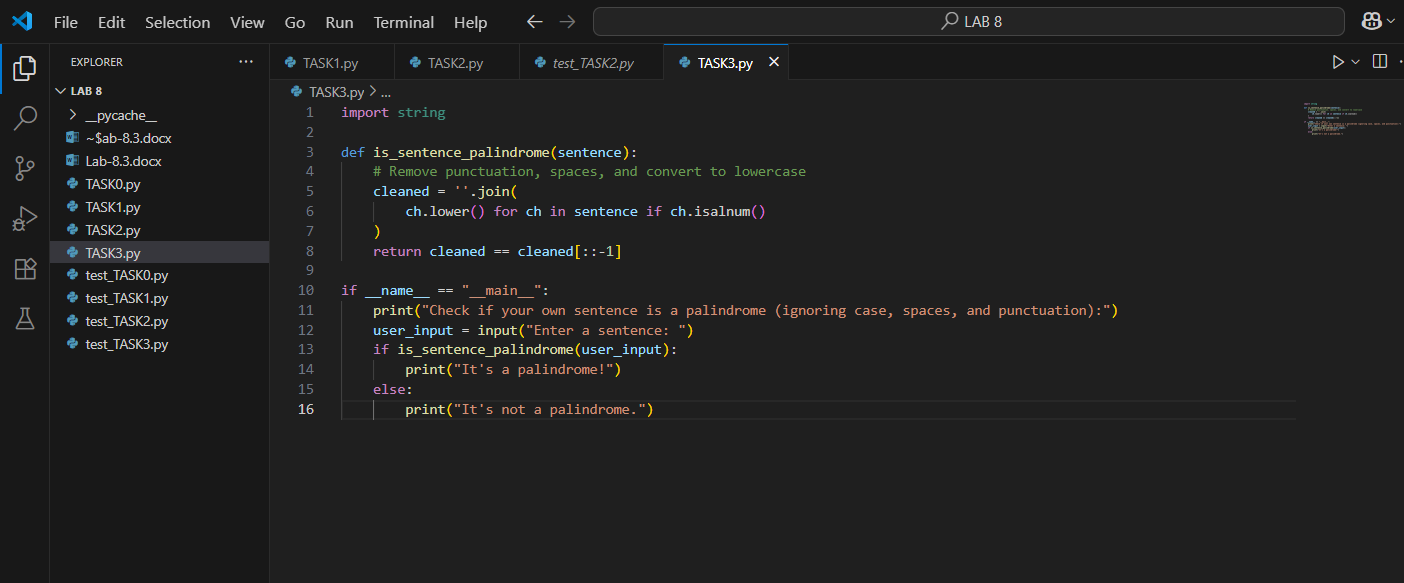
* Example:

"A man a plan a canal Panama" → True

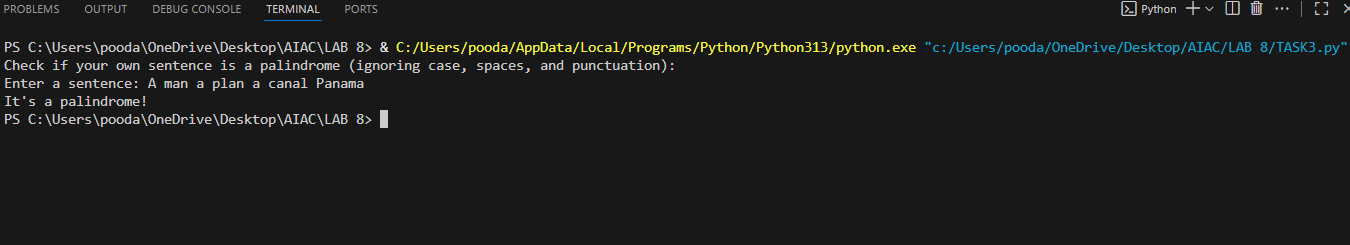
**Expected Output#3**

* Function returns True/False for cleaned sentences
* Implement the function to pass AI-generated tests.

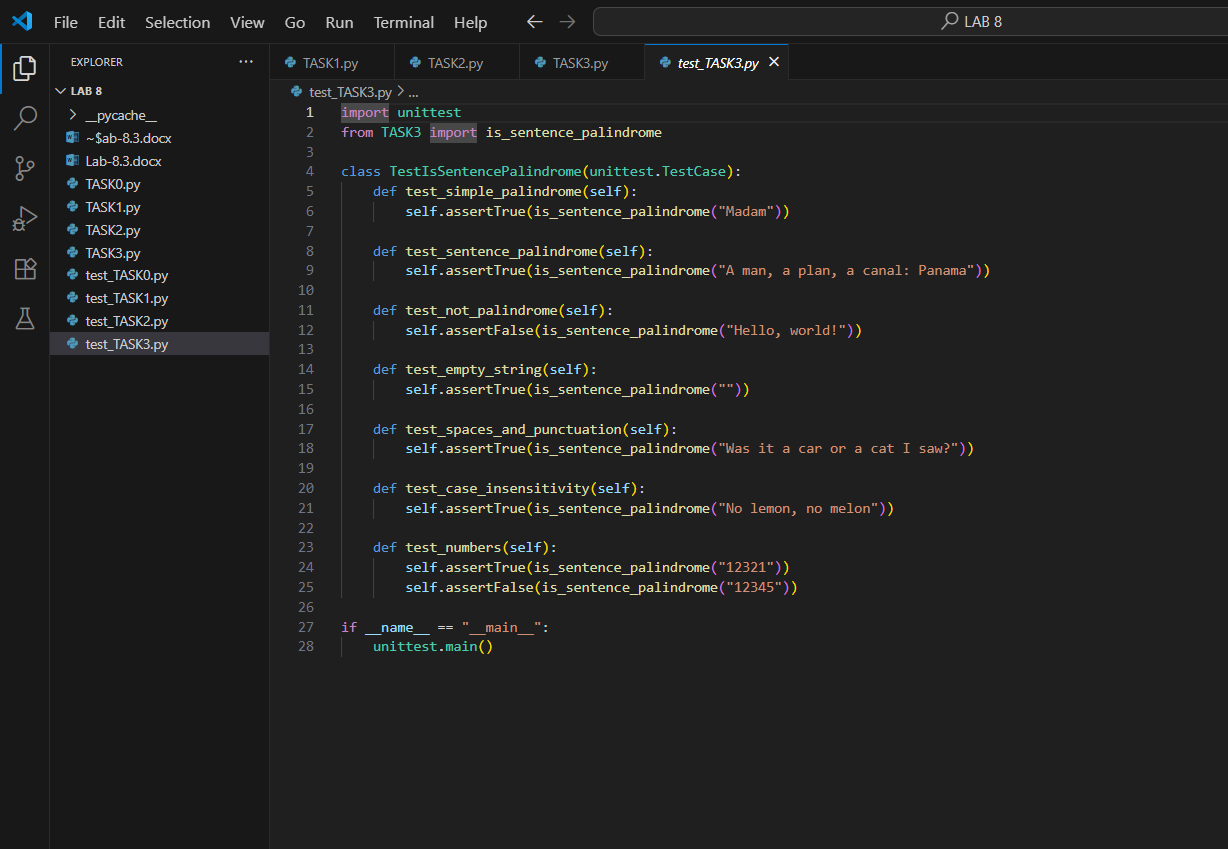
TASK 3:



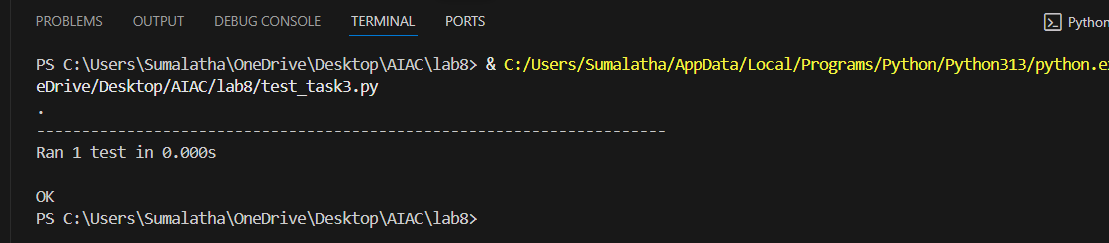
OUTPUT:



TEST CASE :



OUTPUT:



**Task Description#4**

* Let AI fix it Prompt AI to generate test cases for a ShoppingCart class (add\_item, remove\_item, total\_cost).

**Methods:**

Add\_item(name,orice)

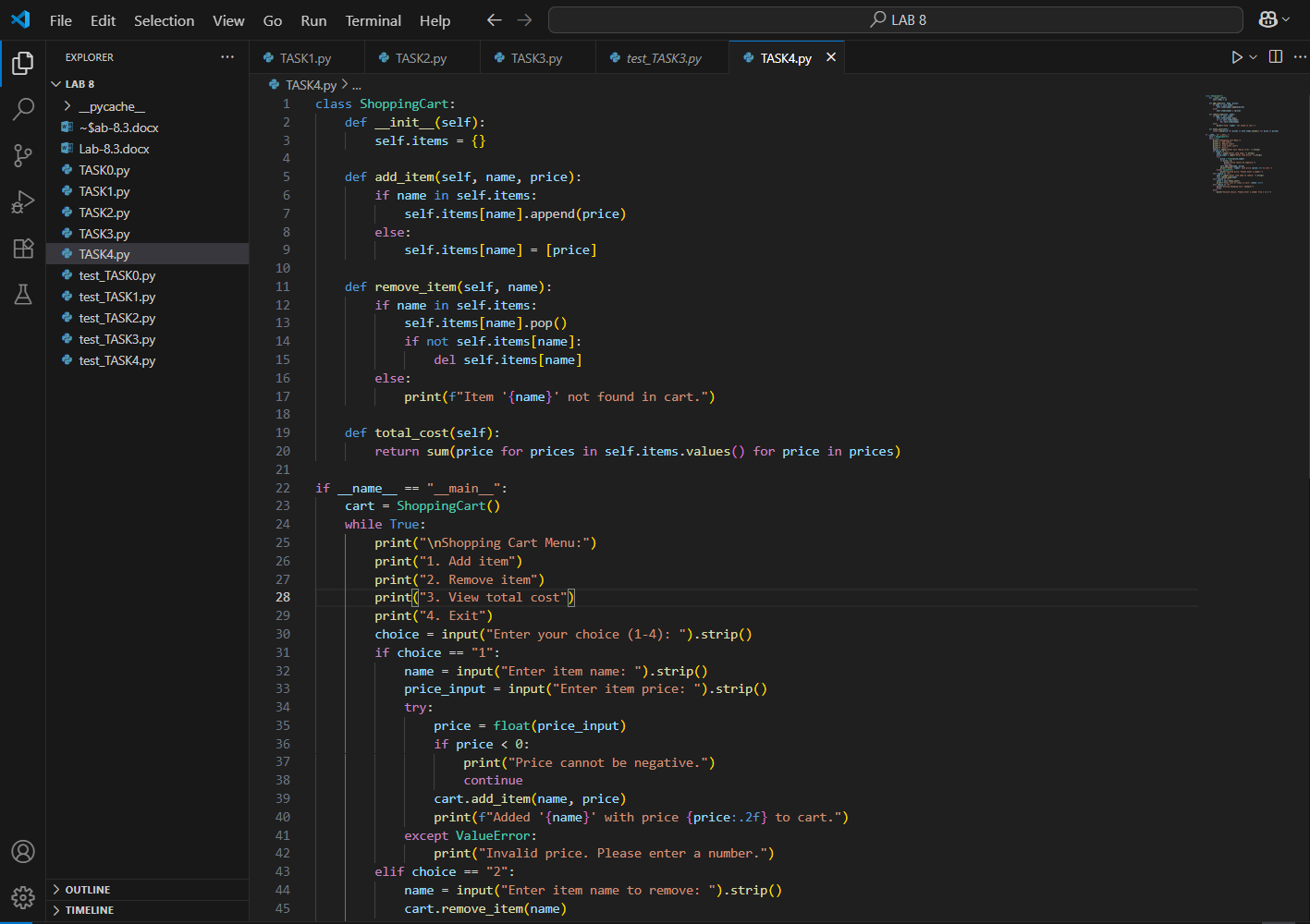
Remove\_item(name)

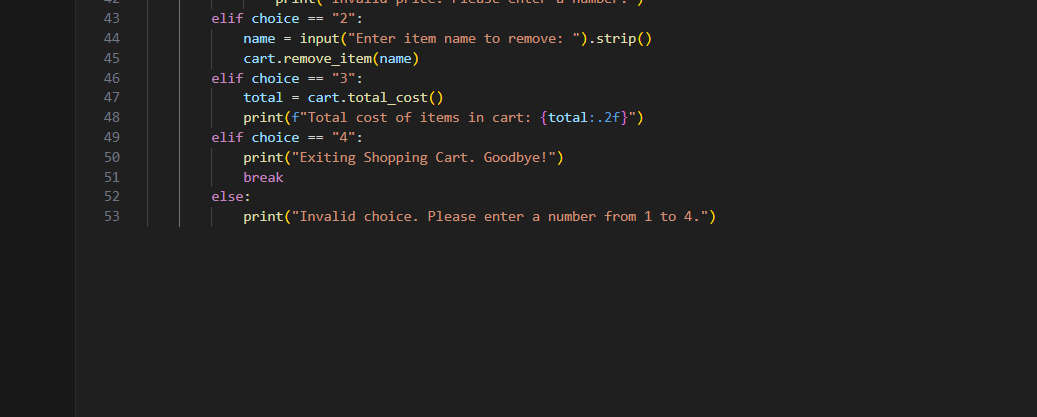
Total\_cost()

**Expected Output#4**

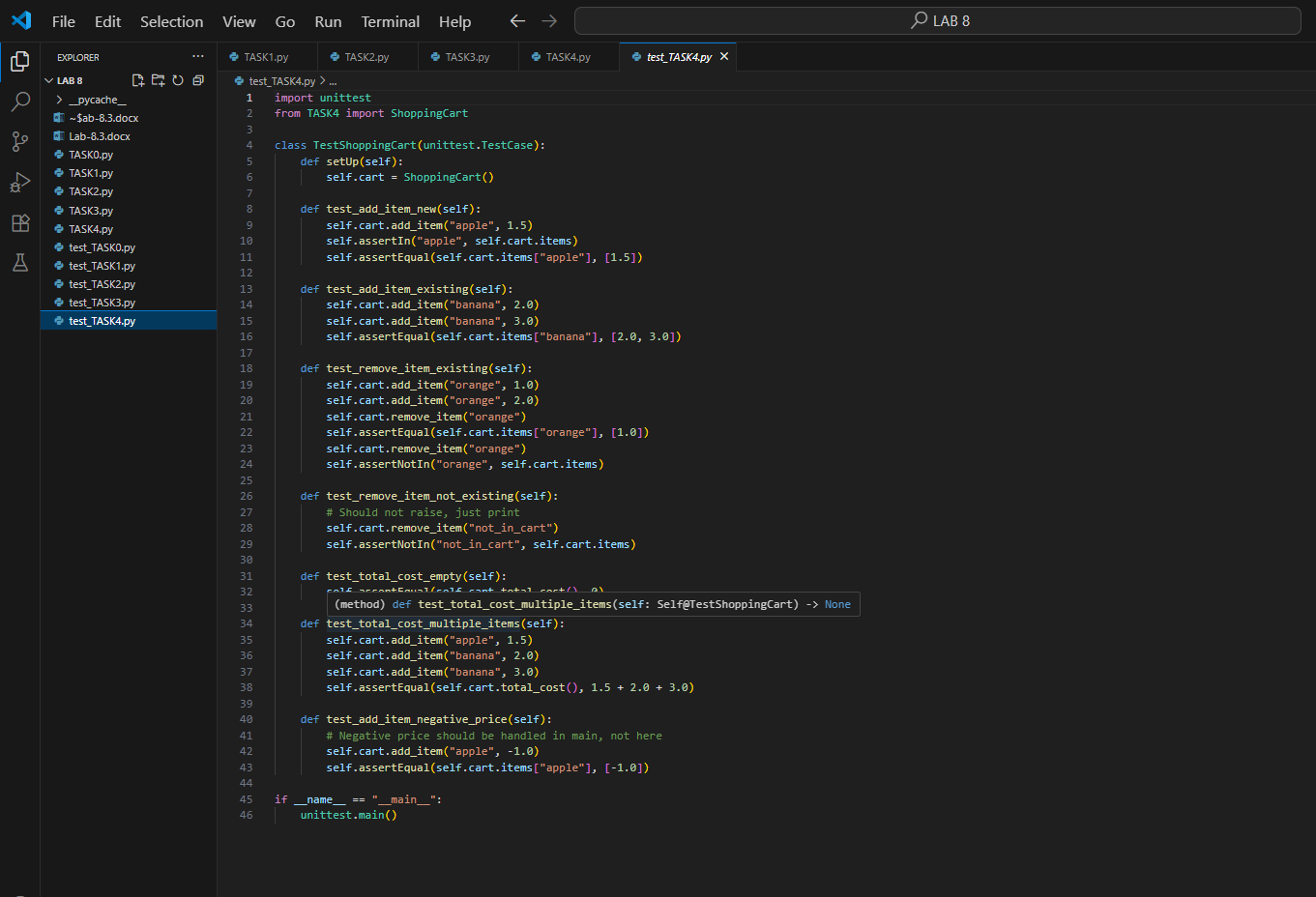
* Full class with tested functionalities

TASK 4 :

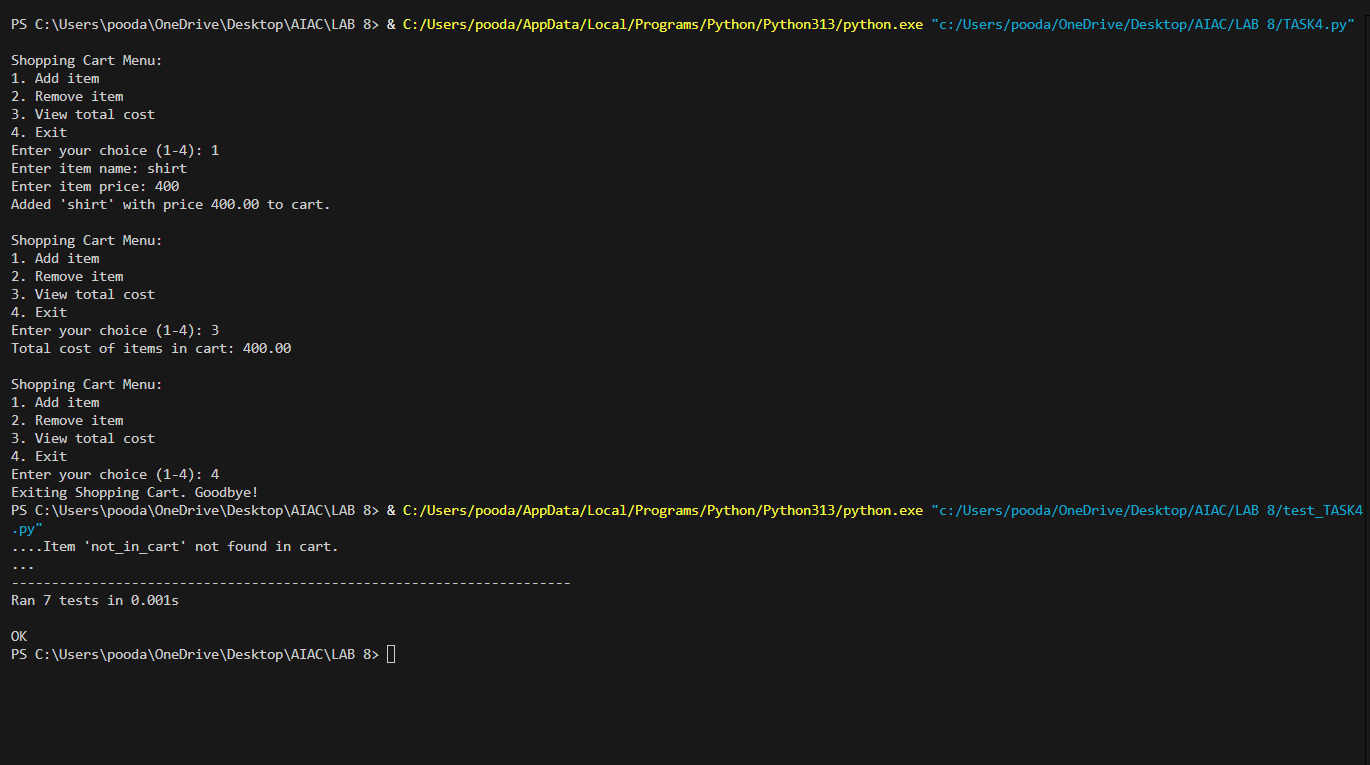




TEST CASE:



OUTPUT:



**Task Description#5**

* Use AI to write test cases for convert\_date\_format(date\_str) to switch from "YYYY-MM-DD" to "DD-MM-YYYY".

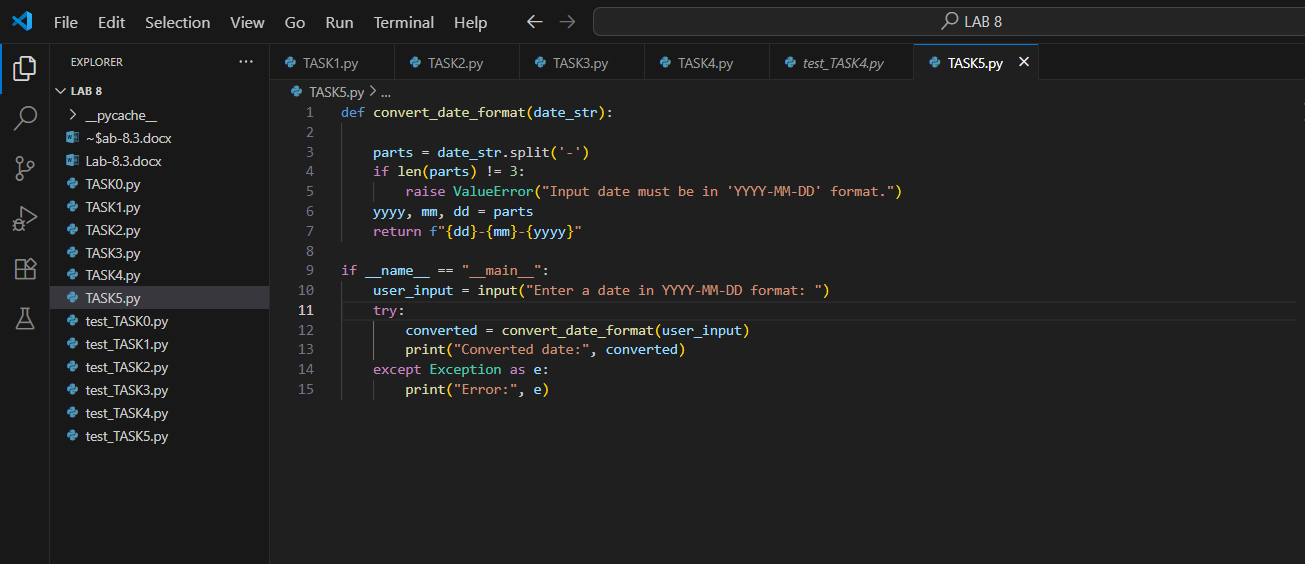
**Example: "2023-10-15" → "15-10-2023"**

**Expected Output#5**

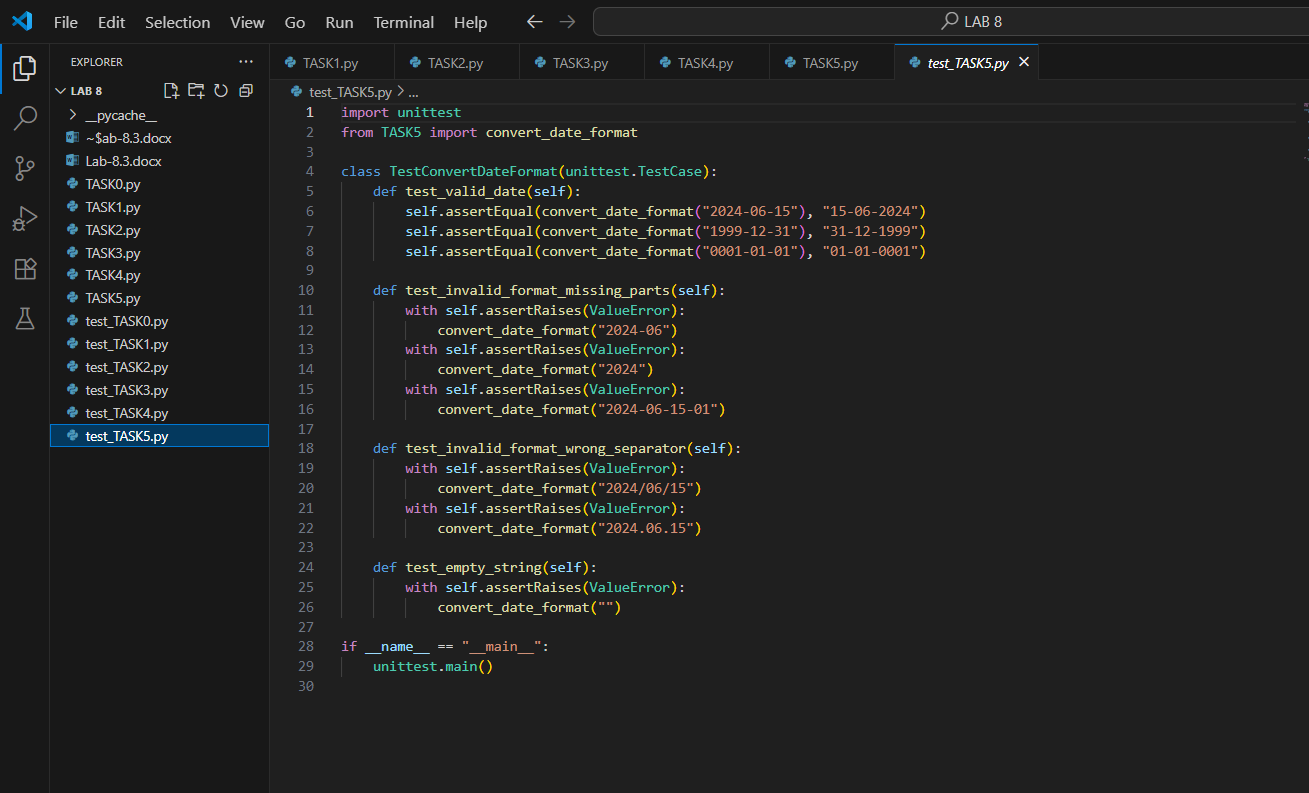
* Function converts input format correctly for all test cases

**Note: Report should be submitted a word document for all tasks in a single document with prompts, comments & code explanation, and output and if required, screenshots**

TASK 5 :



TEST CASE :



OUTPUT :

