NAME:CH.Kruthankiran H.NO:2303A51404 BATCH:26

SCHOOL OF COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE		**DEPARTMENT OF COMPUTER SCIENCE ENGINEERING**	
Program Name: B. Tech	**Assignment Type: Lab**		**Academic Year:**2025-2026
Course Coordinator Name	Dr. Rishabh Mittal		
Instructor(s) Name	Mr. S Naresh Kumar Ms. B. Swathi Dr. Sasanko Shekhar Gantayat Mr. Md Sallauddin Dr. Mathivanan Mr. Y Srikanth Ms. N Shilpa Dr. Rishabh Mittal (Coordinator) Dr. R. Prashant Kumar Mr. Ankushavali MD Mr. B Viswanath Ms. Sujitha Reddy Ms. A. Anitha Ms. M.Madhuri Ms. Katherashala Swetha Ms. Velpula sumalatha Mr. Bingi Raju		
CourseCode	23CS002PC304	**Course Title**	AI Assisted Coding
Year/Sem	III/II	**Regulation**	R23
Date and Day of Assignment	**Week3 – Wednusday**	**Time(s)**	23CSBTB01 To 23CSBTB52
Duration	2 Hours	**Applicable to Batches**	All batches
Assignment Number: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** • Introduce TDD using AI • Generate test cases before implementation • Emphasize testing and validation	Week4 - Wednesday

• Encourage clean, reliable code

Lab Outcomes

Students will be able to:

• Write AI-generated test cases
• Implement code using test-first approach
• Validate using unittest
• Analyze test coverage
• Compare AI vs manual tests

Task 1: Email Validation using TDD

Scenario

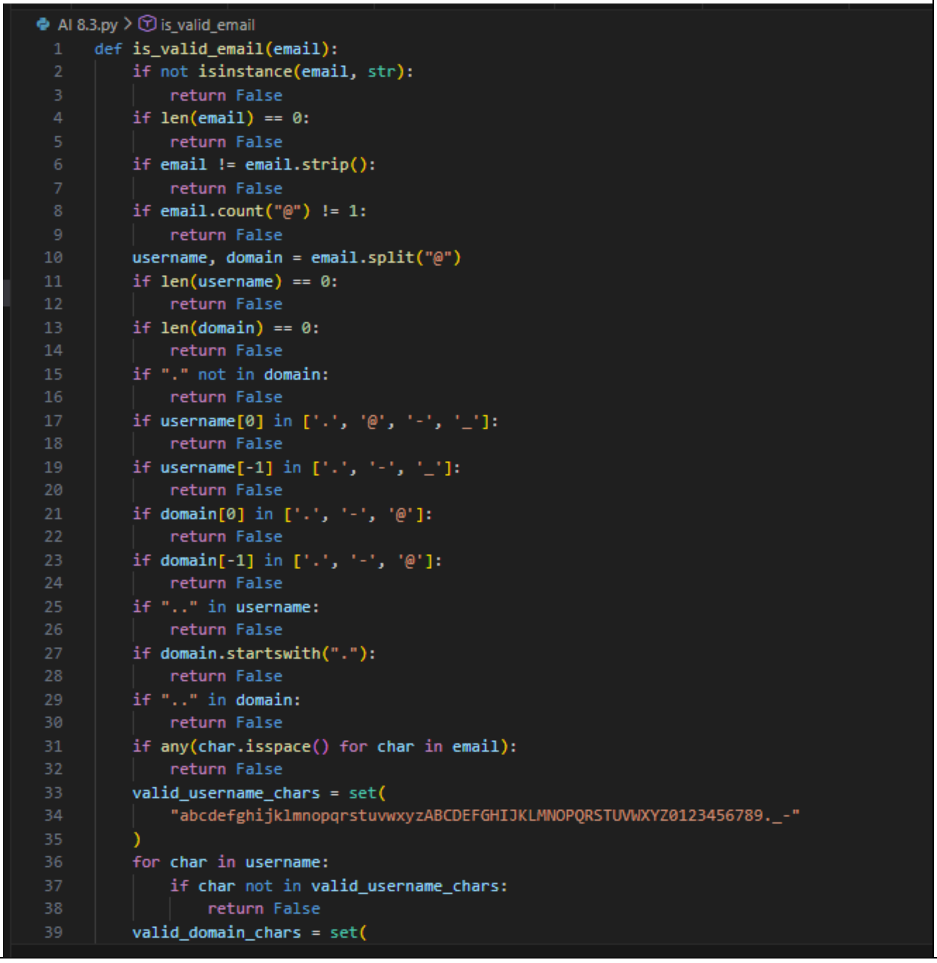
You are developing a user registration system that requires reliable email input validation**.**

Requirements

• Must contain @ and . characters
• Must not start or end with special characters
• Should not allow multiple @ symbols
• AI should generate test cases covering valid and invalid email formats
• Implement is_valid_email(email) to pass all AI-generated test cases

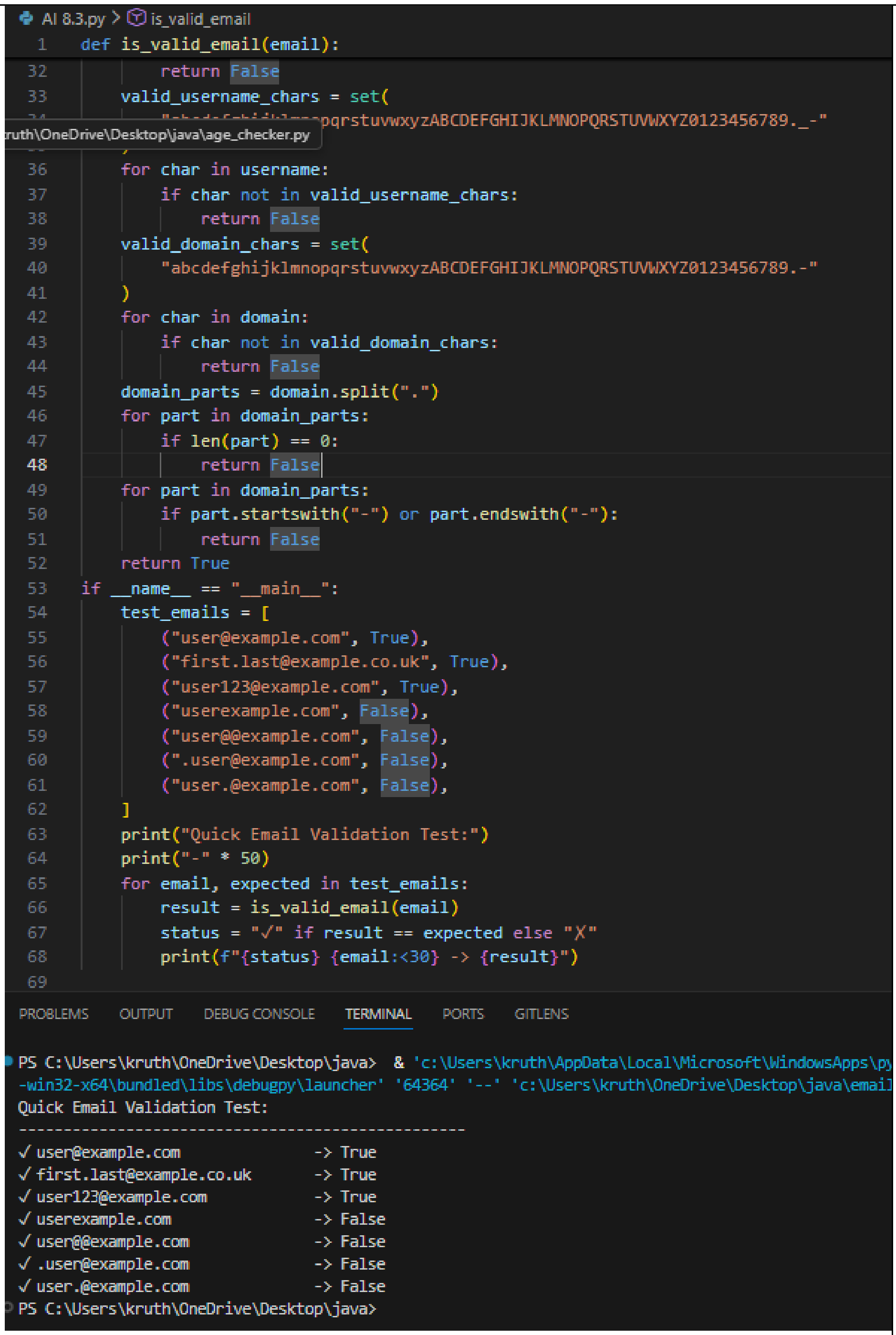
Expected Output

• Python function for email validation
• All AI-generated test cases pass successfully
• Invalid email formats are correctly rejected
• Valid email formats return True



AI 8.3.py > is_valid_email

```python
def is_valid_email(email):
    if not isinstance(email, str):
        return False
    if len(email) == 0:
        return False
    if email != email.strip():
        return False
    if email.count("@") != 1:
        return False
    username, domain = email.split("@")
    if len(username) == 0:
        return False
    if len(domain) == 0:
        return False
    if "." not in domain:
        return False
    if username[0] in ['.', '@', '-', '_']:
        return False
    if username[-1] in ['.', '-', '_']:
        return False
    if domain[0] in ['.', '-', '@']:
        return False
    if domain[-1] in ['.', '-', '@']:
        return False
    if ".." in username:
        return False
    if domain.startswith("."):
        return False
    if ".." in domain:
        return False
    if any(char.isspace() for char in email):
        return False
    valid_username_chars = set(
        "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789._-"
    )
    for char in username:
        if char not in valid_username_chars:
            return False
    valid_domain_chars = set(
```



```python
def is_valid_email(email):
        return False
    valid_username_chars = set(
        "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789._-"
ruth\OneDrive\Desktop\java\age_checker.py
    for char in username:
        if char not in valid_username_chars:
            return False
    valid_domain_chars = set(
        "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789.-"
    )
    for char in domain:
        if char not in valid_domain_chars:
            return False
    domain_parts = domain.split(".")
    for part in domain_parts:
        if len(part) == 0:
            return False
    for part in domain_parts:
        if part.startswith("-") or part.endswith("-"):
            return False
    return True
if __name__ == "__main__":
    test_emails = [
        ("user@example.com", True),
        ("first.last@example.co.uk", True),
        ("user123@example.com", True),
        ("userexample.com", False),
        ("user@@example.com", False),
        (".user@example.com", False),
        ("user.@example.com", False),
    ]
    print("Quick Email Validation Test:")
    print("-" * 50)
    for email, expected in test_emails:
        result = is_valid_email(email)
        status = "✓" if result == expected else "✗"
        print(f"{status} {email:<30} -> {result}")
```

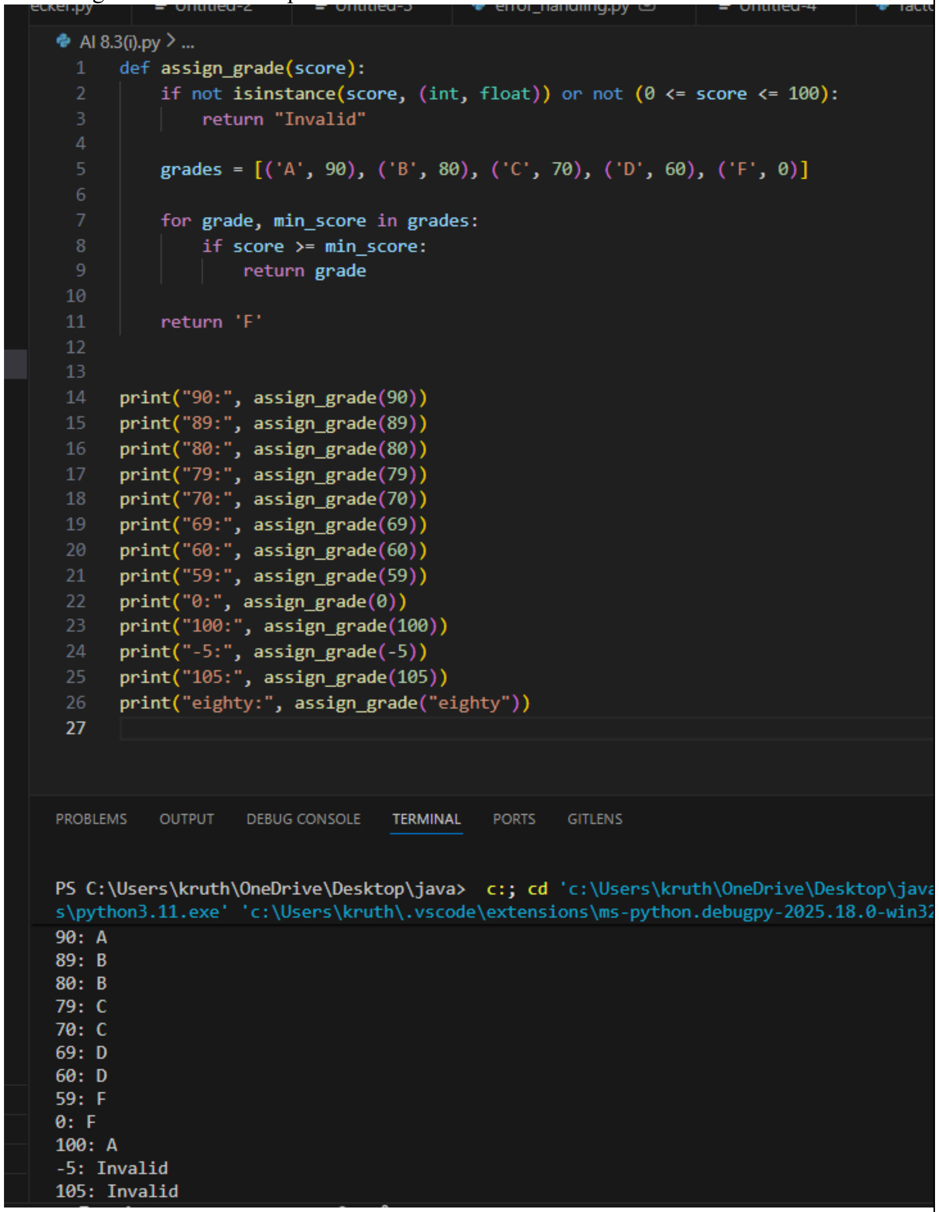
```
PS C:\Users\kruth\OneDrive\Desktop\java>  & 'c:\Users\kruth\AppData\Local\Microsoft\WindowsApps\py
-win32-x64\bundled\libs\debugpy\launcher' '64364' '--' 'c:\Users\kruth\OneDrive\Desktop\java\emai
Quick Email Validation Test:
--------------------------------------------------
✓ user@example.com               -> True
✓ first.last@example.co.uk       -> True
✓ user123@example.com            -> True
✓ userexample.com                -> False
✓ user@@example.com              -> False
✓ .user@example.com              -> False
✓ user.@example.com              -> False
PS C:\Users\kruth\OneDrive\Desktop\java>
```

Task 2: Grade Assignment using Loops
Scenario
You are building an automated grading system for an online examination platform.
Requirements
• AI should generate test cases for assign_grade(score) where:
– 90–100 → A
– 80–89 → B
– 70–79 → C
– 60–69 → D
– Below 60 → F

• Include boundary values (60, 70, 80, 90)
• Include invalid inputs such as -5, 105, "eighty"
• Implement the function using a test-driven approach

Expected Output

• Grade assignment function implemented in Python
• Boundary values handled correctly
• Invalid inputs handled gracefully
• All AI-generated test cases pass



AI 8.3(i).py > ...

```python
def assign_grade(score):
    if not isinstance(score, (int, float)) or not (0 <= score <= 100):
        return "Invalid"

    grades = [('A', 90), ('B', 80), ('C', 70), ('D', 60), ('F', 0)]

    for grade, min_score in grades:
        if score >= min_score:
            return grade

    return 'F'


print("90:", assign_grade(90))
print("89:", assign_grade(89))
print("80:", assign_grade(80))
print("79:", assign_grade(79))
print("70:", assign_grade(70))
print("69:", assign_grade(69))
print("60:", assign_grade(60))
print("59:", assign_grade(59))
print("0:", assign_grade(0))
print("100:", assign_grade(100))
print("-5:", assign_grade(-5))
print("105:", assign_grade(105))
print("eighty:", assign_grade("eighty"))
```

```
PS C:\Users\kruth\OneDrive\Desktop\java>  c:; cd 'c:\Users\kruth\OneDrive\Desktop\java
s\python3.11.exe' 'c:\Users\kruth\.vscode\extensions\ms-python.debugpy-2025.18.0-win32
90: A
89: B
80: B
79: C
70: C
69: D
60: D
59: F
0: F
100: A
-5: Invalid
105: Invalid
```

Task 3: Sentence Palindrome Checker

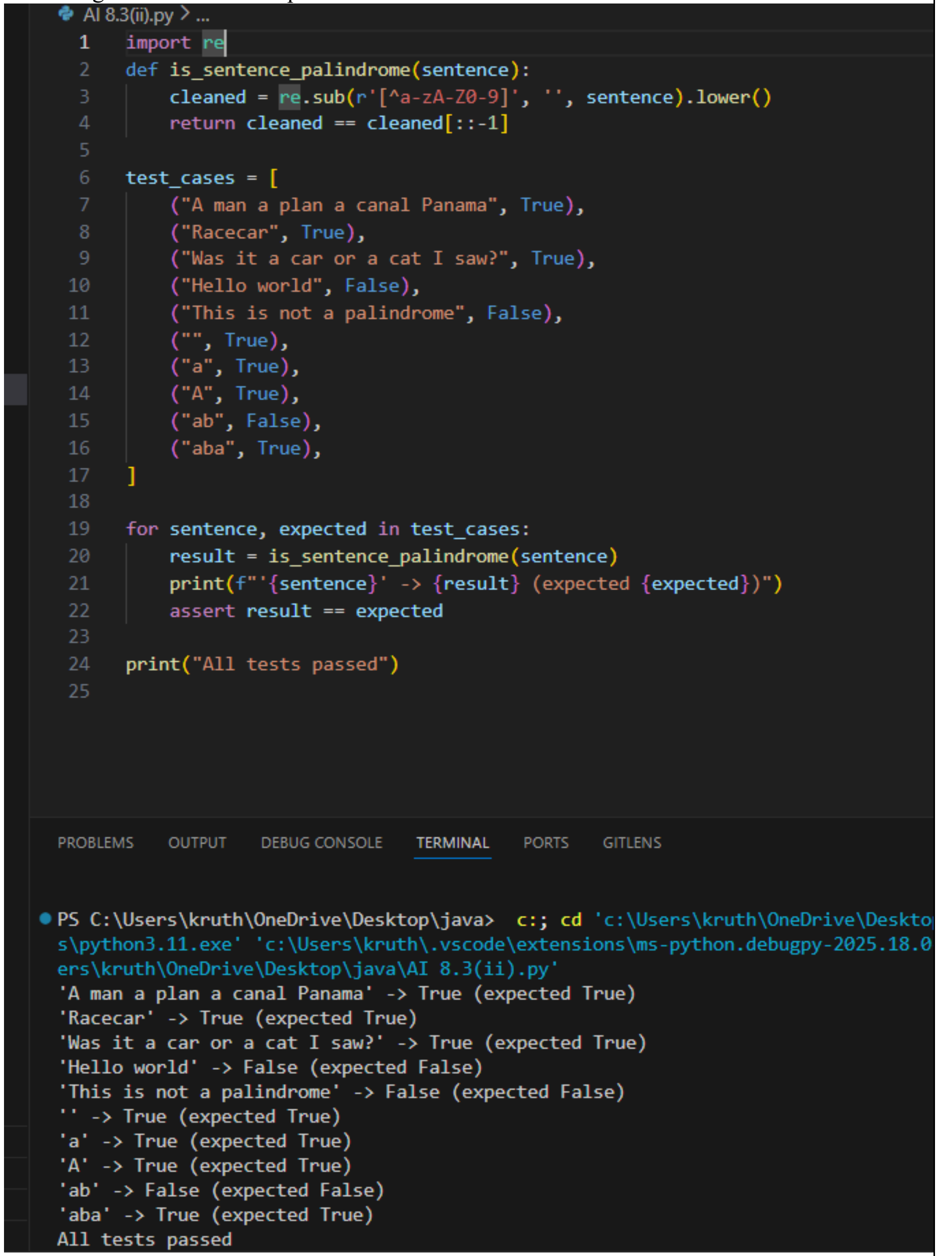
Scenario

You are developing a text-processing utility to analyze sentences.

Requirements

• AI should generate test cases for is_sentence_palindrome(sentence)
• Ignore case, spaces, and punctuation
• Test both palindromic and non-palindromic sentences
• Example:

– "A man a plan a canal Panama" → True
Expected Output
• Function correctly identifies sentence palindromes
• Case and punctuation are ignored
• Returns True or False accurately
• All AI-generated test cases pass



AI 8.3(ii).py > ...

```python
import re
def is_sentence_palindrome(sentence):
    cleaned = re.sub(r'[^a-zA-Z0-9]', '', sentence).lower()
    return cleaned == cleaned[::-1]

test_cases = [
    ("A man a plan a canal Panama", True),
    ("Racecar", True),
    ("Was it a car or a cat I saw?", True),
    ("Hello world", False),
    ("This is not a palindrome", False),
    ("", True),
    ("a", True),
    ("A", True),
    ("ab", False),
    ("aba", True),
]

for sentence, expected in test_cases:
    result = is_sentence_palindrome(sentence)
    print(f"'{sentence}' -> {result} (expected {expected})")
    assert result == expected

print("All tests passed")
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS

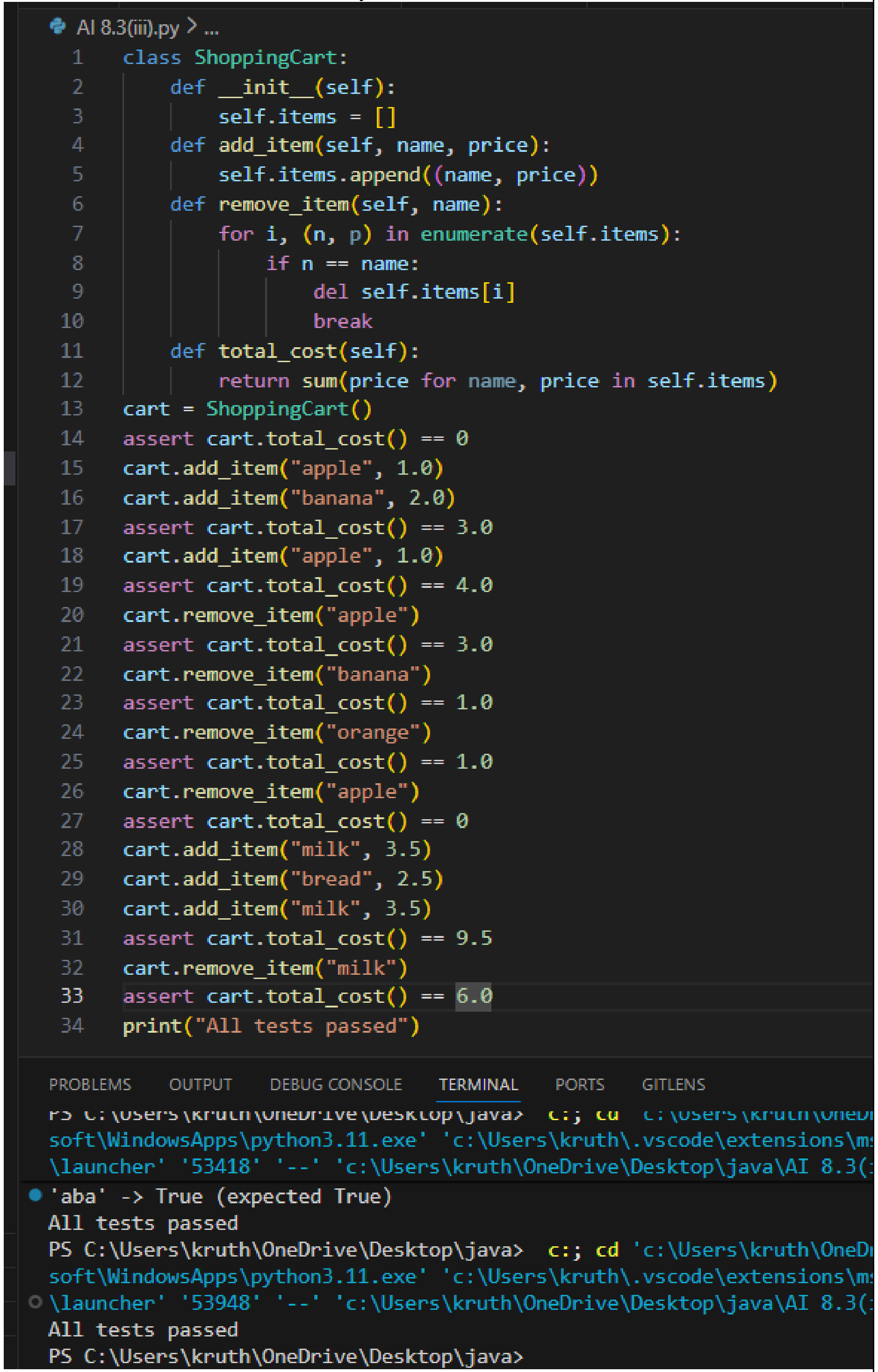
```
PS C:\Users\kruth\OneDrive\Desktop\java>  c:; cd 'c:\Users\kruth\OneDrive\Deskto
s\python3.11.exe' 'c:\Users\kruth\.vscode\extensions\ms-python.debugpy-2025.18.0
ers\kruth\OneDrive\Desktop\java\AI 8.3(ii).py'
'A man a plan a canal Panama' -> True (expected True)
'Racecar' -> True (expected True)
'Was it a car or a cat I saw?' -> True (expected True)
'Hello world' -> False (expected False)
'This is not a palindrome' -> False (expected False)
'' -> True (expected True)
'a' -> True (expected True)
'A' -> True (expected True)
'ab' -> False (expected False)
'aba' -> True (expected True)
All tests passed
```

Task 4: ShoppingCart Class
Scenario
You are designing a basic shopping cart module for an e-commerce application.
Requirements
• AI should generate test cases for the ShoppingCart class
• Class must include the following methods:
– add_item(name, price)
– remove_item(name)
– total_cost()

• Validate correct addition, removal, and cost calculation
• Handle empty cart scenarios

Expected Output

• Fully implemented ShoppingCart class
• All methods pass AI-generated test cases
• Total cost is calculated accurately
• Items are added and removed correctly

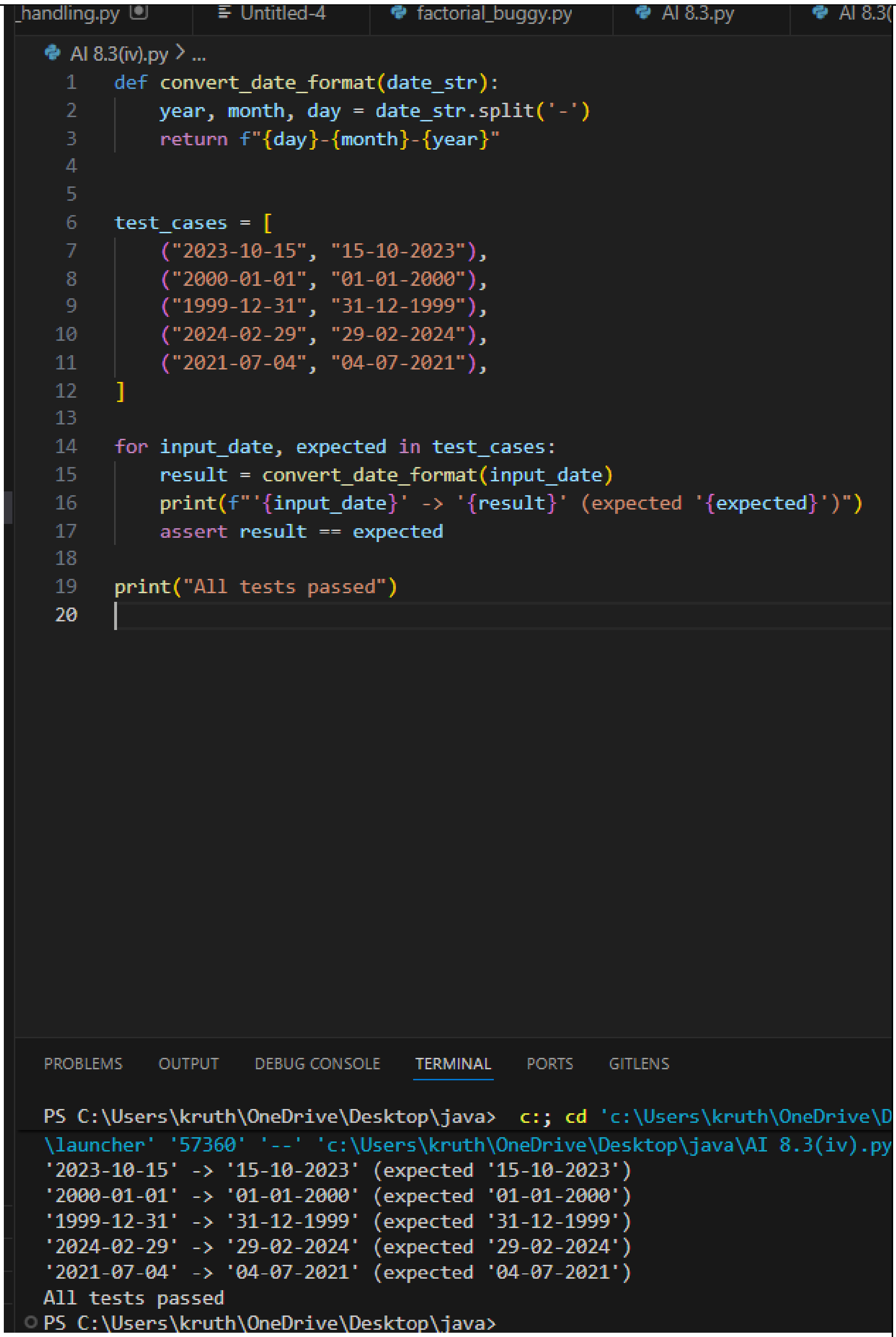


AI 8.3(iii).py > ...

```python
class ShoppingCart:
    def __init__(self):
        self.items = []
    def add_item(self, name, price):
        self.items.append((name, price))
    def remove_item(self, name):
        for i, (n, p) in enumerate(self.items):
            if n == name:
                del self.items[i]
                break
    def total_cost(self):
        return sum(price for name, price in self.items)
cart = ShoppingCart()
assert cart.total_cost() == 0
cart.add_item("apple", 1.0)
cart.add_item("banana", 2.0)
assert cart.total_cost() == 3.0
cart.add_item("apple", 1.0)
assert cart.total_cost() == 4.0
cart.remove_item("apple")
assert cart.total_cost() == 3.0
cart.remove_item("banana")
assert cart.total_cost() == 1.0
cart.remove_item("orange")
assert cart.total_cost() == 1.0
cart.remove_item("apple")
assert cart.total_cost() == 0
cart.add_item("milk", 3.5)
cart.add_item("bread", 2.5)
cart.add_item("milk", 3.5)
assert cart.total_cost() == 9.5
cart.remove_item("milk")
assert cart.total_cost() == 6.0
print("All tests passed")
```

```
PS C:\Users\kruth\OneDrive\Desktop\java>  c:; cd 'c:\Users\kruth\OneDr
soft\WindowsApps\python3.11.exe' 'c:\Users\kruth\.vscode\extensions\ms
\launcher' '53418' '--' 'c:\Users\kruth\OneDrive\Desktop\java\AI 8.3(
'aba' -> True (expected True)
All tests passed
PS C:\Users\kruth\OneDrive\Desktop\java>  c:; cd 'c:\Users\kruth\OneDr
soft\WindowsApps\python3.11.exe' 'c:\Users\kruth\.vscode\extensions\ms
\launcher' '53948' '--' 'c:\Users\kruth\OneDrive\Desktop\java\AI 8.3(
All tests passed
PS C:\Users\kruth\OneDrive\Desktop\java>
```

	Task 5: Date Format Conversion **Scenario** You are creating a utility function to convert date formats for reports. **Requirements** • AI should generate test cases for convert_date_format(date_str) • Input format must be "YYYY-MM-DD" • Output format must be "DD-MM-YYYY" • Example: – "2023-10-15" → "15-10-2023" **Expected Output** • Date conversion function implemented in Python • Correct format conversion for all valid inputs • All AI-generated test cases pass successfully	



_handling.py   Untitled-4   factorial_buggy.py   AI 8.3.py   AI 8.3(

AI 8.3(iv).py > ...

```python
def convert_date_format(date_str):
    year, month, day = date_str.split('-')
    return f"{day}-{month}-{year}"


test_cases = [
    ("2023-10-15", "15-10-2023"),
    ("2000-01-01", "01-01-2000"),
    ("1999-12-31", "31-12-1999"),
    ("2024-02-29", "29-02-2024"),
    ("2021-07-04", "04-07-2021"),
]

for input_date, expected in test_cases:
    result = convert_date_format(input_date)
    print(f"'{input_date}' -> '{result}' (expected '{expected}')")
    assert result == expected

print("All tests passed")
```

PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS   GITLENS

```
PS C:\Users\kruth\OneDrive\Desktop\java>  c:; cd 'c:\Users\kruth\OneDrive\D
\launcher' '57360' '--' 'c:\Users\kruth\OneDrive\Desktop\java\AI 8.3(iv).py
'2023-10-15' -> '15-10-2023' (expected '15-10-2023')
'2000-01-01' -> '01-01-2000' (expected '01-01-2000')
'1999-12-31' -> '31-12-1999' (expected '31-12-1999')
'2024-02-29' -> '29-02-2024' (expected '29-02-2024')
'2021-07-04' -> '04-07-2021' (expected '04-07-2021')
All tests passed
PS C:\Users\kruth\OneDrive\Desktop\java>
```

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