Name:I. Abhinay Powar H.No:2303A51811 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)			

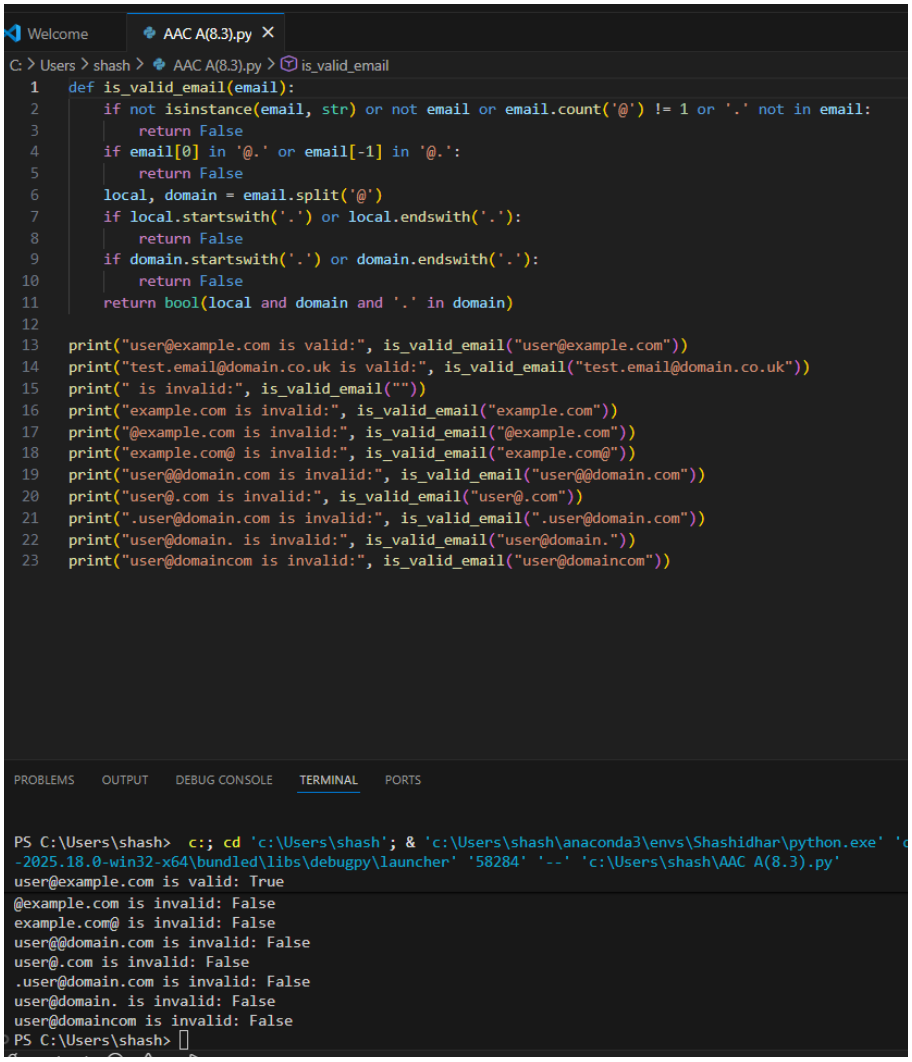
Q.No.	**Question**	***ExpectedTime to complete***
1	**Lab 8: Test-Driven Development with AI – Generating and Working with Test Cases** **Lab Objectives** • Introduce TDD using AI • Generate test cases before implementation • Emphasize testing and validation • Encourage clean, reliable code **Lab Outcomes** Students will be able to: • Write AI-generated test cases	Week4 - Wednesday

Name:I. Abhinay Powar H.No:2303A51811 Batch:26

- 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



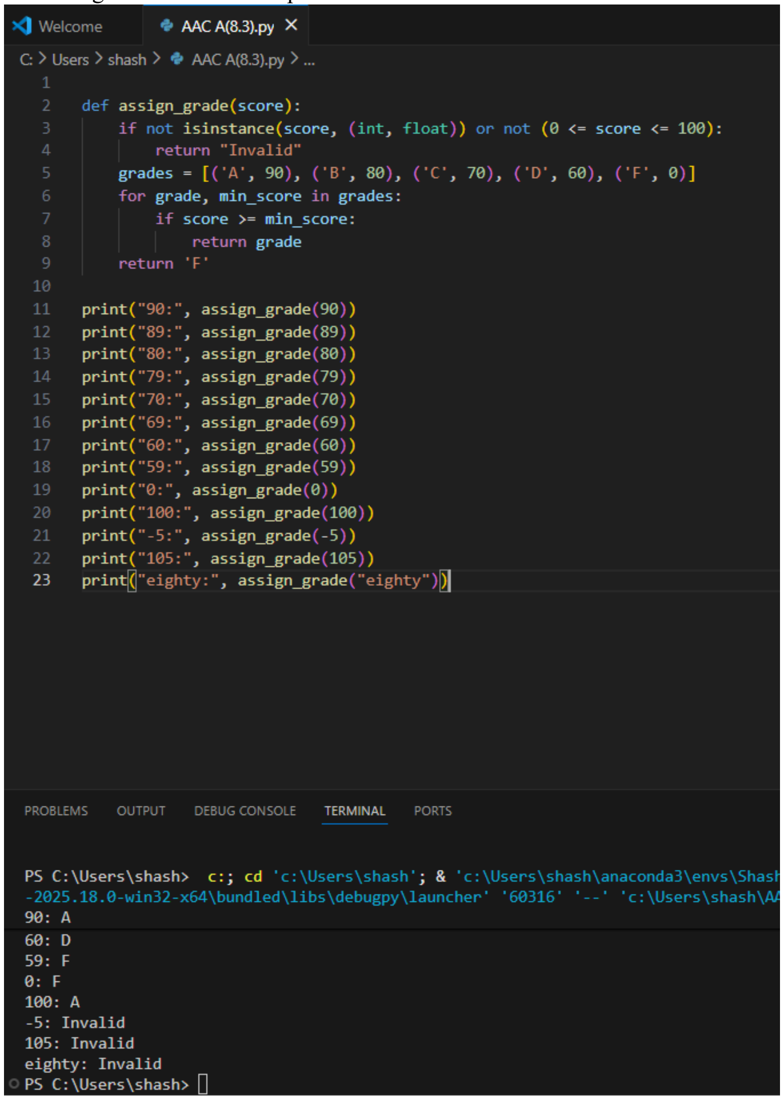
```python
def is_valid_email(email):
    if not isinstance(email, str) or not email or email.count('@') != 1 or '.' not in email:
        return False
    if email[0] in '@.' or email[-1] in '@.':
        return False
    local, domain = email.split('@')
    if local.startswith('.') or local.endswith('.'):
        return False
    if domain.startswith('.') or domain.endswith('.'):
        return False
    return bool(local and domain and '.' in domain)

print("user@example.com is valid:", is_valid_email("user@example.com"))
print("test.email@domain.co.uk is valid:", is_valid_email("test.email@domain.co.uk"))
print(" is invalid:", is_valid_email(""))
print("example.com is invalid:", is_valid_email("example.com"))
print("@example.com is invalid:", is_valid_email("@example.com"))
print("example.com@ is invalid:", is_valid_email("example.com@"))
print("user@@domain.com is invalid:", is_valid_email("user@@domain.com"))
print("user@.com is invalid:", is_valid_email("user@.com"))
print(".user@domain.com is invalid:", is_valid_email(".user@domain.com"))
print("user@domain. is invalid:", is_valid_email("user@domain."))
print("user@domaincom is invalid:", is_valid_email("user@domaincom"))
```

```
PS C:\Users\shash>  c:; cd 'c:\Users\shash'; & 'c:\Users\shash\anaconda3\envs\Shashidhar\python.exe' 'c
-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '58284' '--' 'c:\Users\shash\AAC A(8.3).py'
user@example.com is valid: True
@example.com is invalid: False
example.com@ is invalid: False
user@@domain.com is invalid: False
user@.com is invalid: False
.user@domain.com is invalid: False
user@domain. is invalid: False
user@domaincom is invalid: False
PS C:\Users\shash> 
```

Task 2: Grade Assignment using Loops
Scenario

Name:I. Abhinay Powar   H.No:2303A51811   Batch:26

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



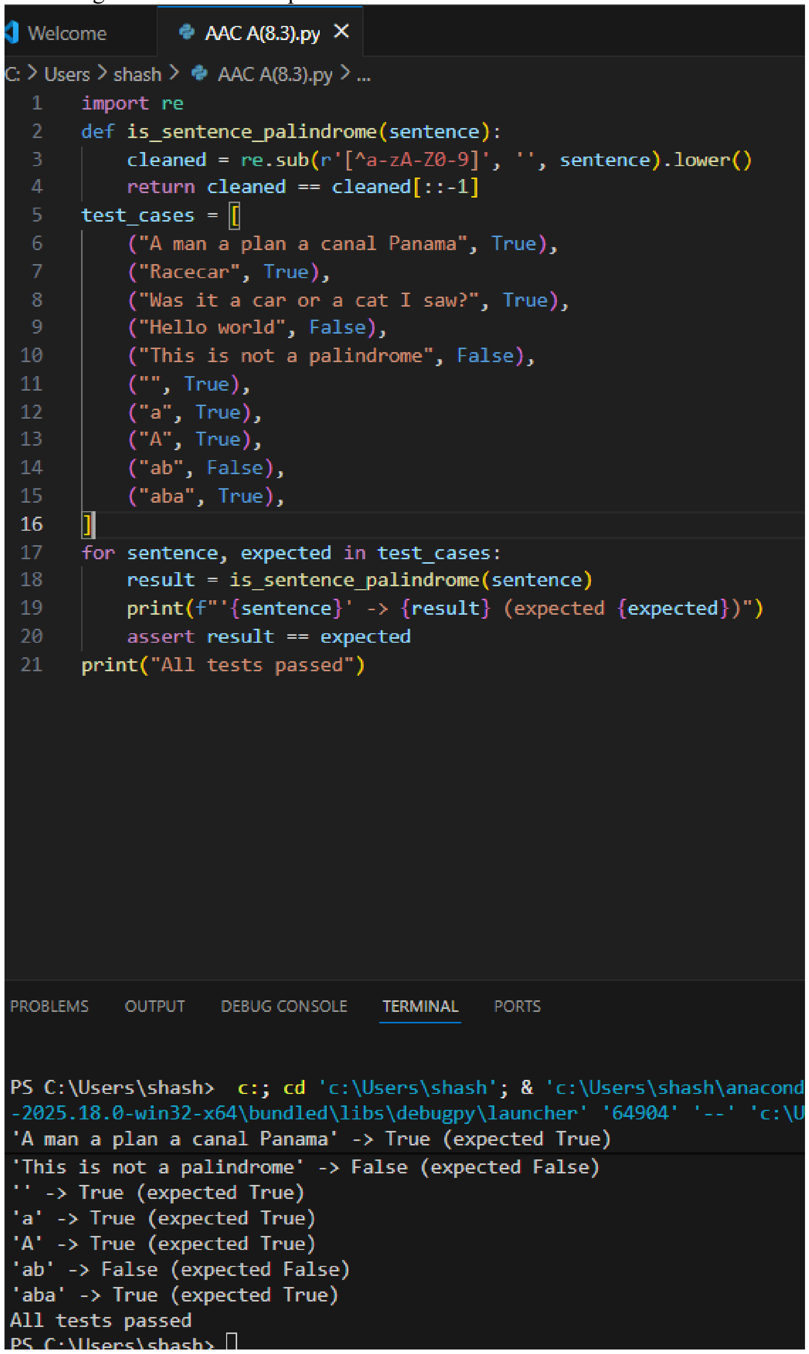
```python

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\shash>  c:; cd 'c:\Users\shash'; & 'c:\Users\shash\anaconda3\envs\Shash
-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '60316' '--' 'c:\Users\shash\AA
90: A
60: D
59: F
0: F
100: A
-5: Invalid
105: Invalid
eighty: Invalid
PS C:\Users\shash>
```

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)

Name:I. Abhinay Powar   H.No:2303A51811   Batch:26

• 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



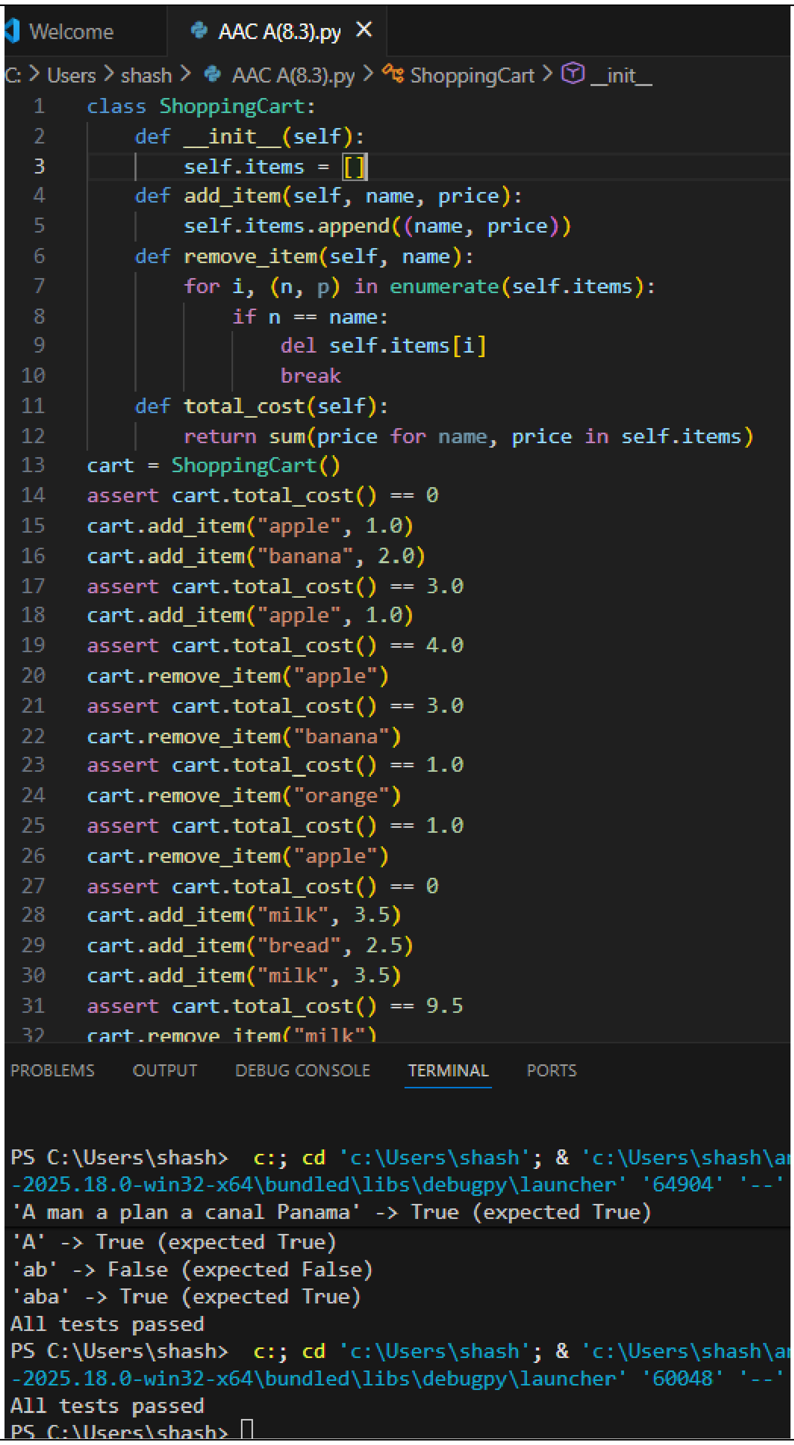
```python
import re
def is_sentence_palindrome(sentence):
    cleaned = re.sub(r'[^a-zA-Z0-9]', '', sentence).lower()
    return cleaned == cleaned[::-1]
test_cases = [
    ("A man a plan a canal Panama", True),
    ("Racecar", True),
    ("Was it a car or a cat I saw?", True),
    ("Hello world", False),
    ("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")
```

```
PS C:\Users\shash>  c:; cd 'c:\Users\shash'; & 'c:\Users\shash\anacond
-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '64904' '--' 'c:\U
'A man a plan a canal Panama' -> True (expected True)
'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
PS C:\Users\shash> 
```

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	

Name:I. Abhinay Powar   H.No:2303A51811   Batch:26



Welcome   AAC A(8.3).py

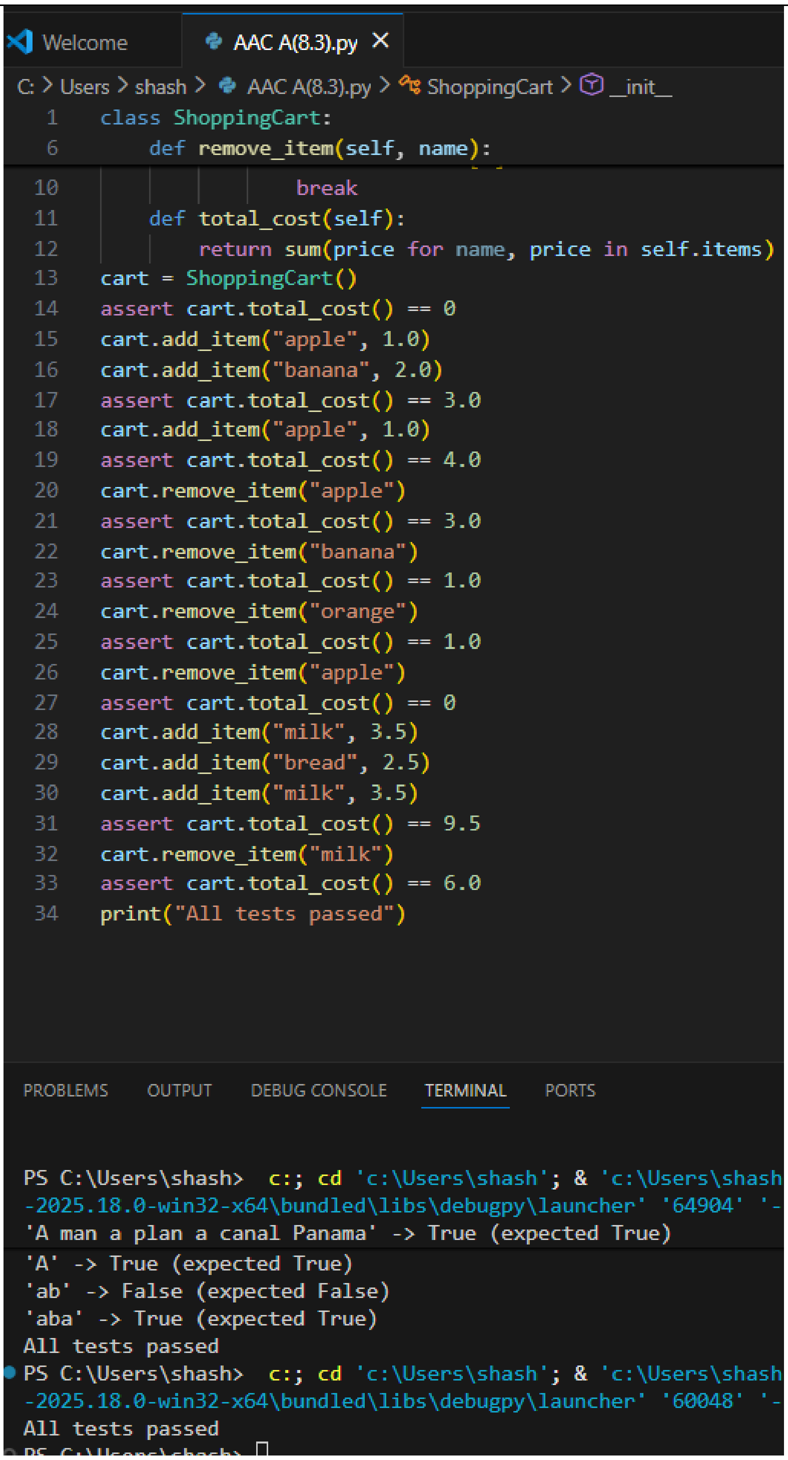
C: > Users > shash > AAC A(8.3).py > ShoppingCart > __init__

```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")
```

PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

```
PS C:\Users\shash>  c:; cd 'c:\Users\shash'; & 'c:\Users\shash\a
-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '64904' '--'
'A man a plan a canal Panama' -> True (expected True)
'A' -> True (expected True)
'ab' -> False (expected False)
'aba' -> True (expected True)
All tests passed
PS C:\Users\shash>  c:; cd 'c:\Users\shash'; & 'c:\Users\shash\a
-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '60048' '--'
All tests passed
PS C:\Users\shash>
```

Name:I. Abhinay Powar H.No:2303A51811 Batch:26


Welcome
AAC A(8.3).py
C: > Users > shash > AAC A(8.3).py > ShoppingCart > __init__
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS

```python
class ShoppingCart:
    def remove_item(self, name):
                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\shash>  c:; cd 'c:\Users\shash'; & 'c:\Users\shash
-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '64904' '-
'A man a plan a canal Panama' -> True (expected True)
'A' -> True (expected True)
'ab' -> False (expected False)
'aba' -> True (expected True)
All tests passed
PS C:\Users\shash>  c:; cd 'c:\Users\shash'; & 'c:\Users\shash
-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '60048' '-
All tests passed
PS C:\Users\shash> 
```

Task 5: Date Format Conversion

Scenario

You are creating a utility function to convert date formats for reports.

Requirements

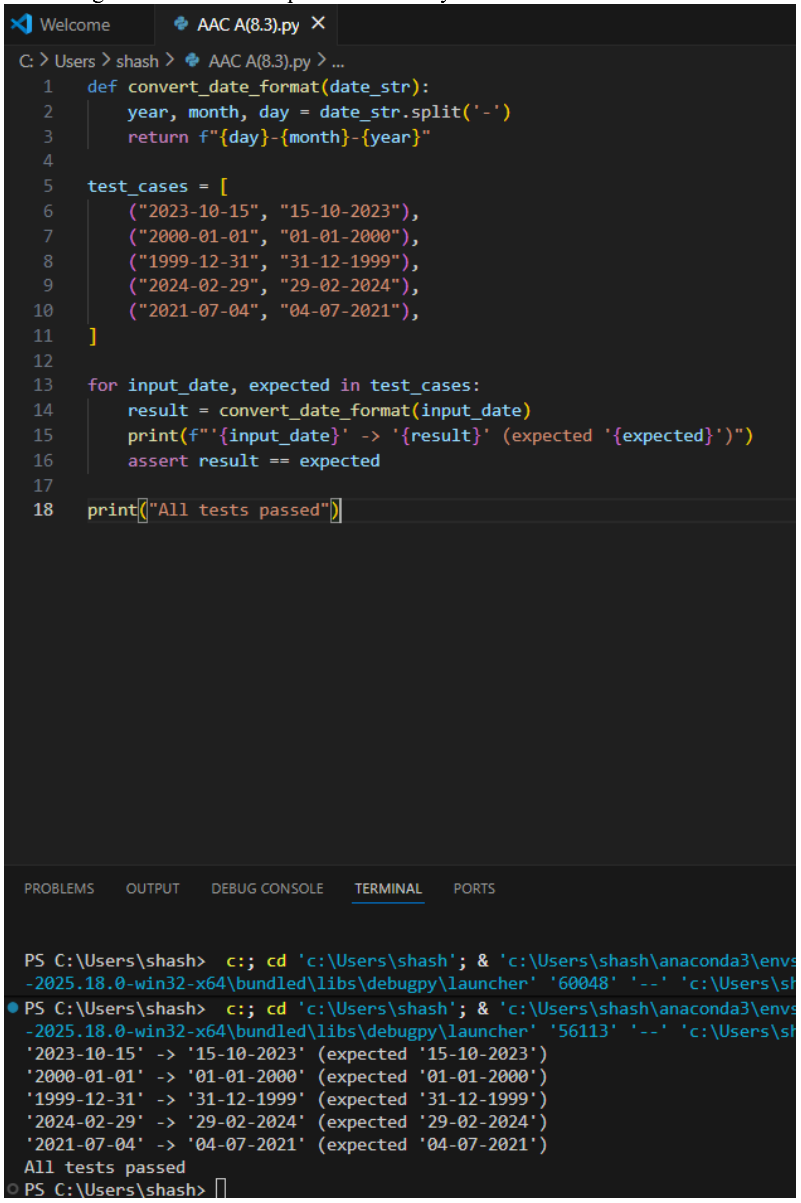
- AI should generate test cases for convert_date_format(date_str)
- Input format must be "YYYY-MM-DD"
- Output format must be "DD-MM-YYYY"
- Example:

Name:I. Abhinay Powar H.No:2303A51811 Batch:26

– "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



```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")
```

```
PS C:\Users\shash>  c:; cd 'c:\Users\shash'; & 'c:\Users\shash\anaconda3\envs
-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '60048' '--' 'c:\Users\sh
PS C:\Users\shash>  c:; cd 'c:\Users\shash'; & 'c:\Users\shash\anaconda3\envs
-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '56113' '--' 'c:\Users\sh
'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\shash>
```

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.