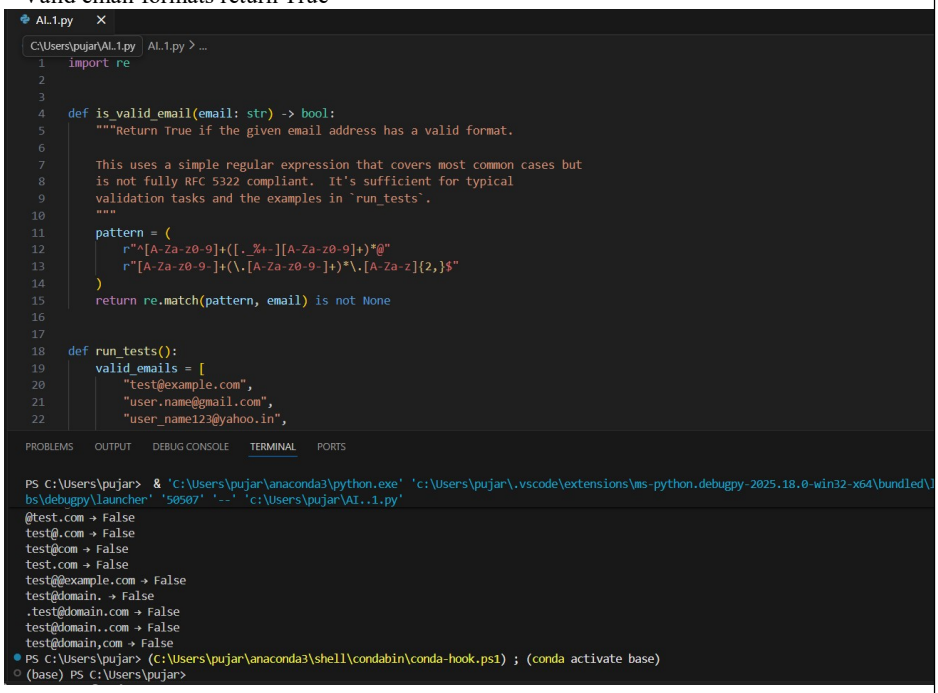
Name : Pujari Pranay Kumar Hall:No : 2303A51829 Batch:26

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

• 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



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

```
C: > Users > pujar > AI...2.py > ...
  1  def assign_grade(score):
  2      """Return a letter grade for a numeric score.
  3
  4      Valid scores are integers or floats between 0 and 100 inclusive.
  5      Non-numeric values or values outside this range produce "Invalid Input".
  6      """
  7      try:
  8          # Ensure we handle things like "80" gracefully as invalid
  9          if not isinstance(score, (int, float)):
 10              raise ValueError
 11          if score < 0 or score > 100:
 12              raise ValueError
 13      except Exception:
 14          return "Invalid Input"
 15
 16      if score >= 90:
 17          return "A"
 18      if score >= 80:
 19          return "B"
 20      if score >= 70:
 21          return "C"
 22      if score >= 60:
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

(base) PS C:\Users\pujar>  c:; cd 'c:\Users\pujar'; & 'C:\Users\pujar\anaconda3\python.exe' 'c:\Users\puja
y-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '54630' '--' 'c:\Users\pujar\AI...2.py'
Score: 69 | Expected: D | Got: D | PASS
Score: 65 | Expected: D | Got: D | PASS
Score: 60 | Expected: D | Got: D | PASS
Score: 59 | Expected: F | Got: F | PASS
Score: 40 | Expected: F | Got: F | PASS
Score: 0 | Expected: F | Got: F | PASS
Score: -5 | Expected: Invalid Input | Got: Invalid Input | PASS
Score: 105 | Expected: Invalid Input | Got: Invalid Input | PASS
Score: eighty | Expected: Invalid Input | Got: Invalid Input | PASS
Score: None | Expected: Invalid Input | Got: Invalid Input | PASS
(base) PS C:\Users\pujar>
```

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

```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})")
```

```
(base) PS C:\Users\pujar>  c:; cd 'c:\Users\pujar'; & 'C:\Users\pujar\anaconda3\python.exe' 'c:\
y-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '64821' '--' 'c:\Users\pujar\ai..4.py'
'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
(base) PS C:\Users\pujar>
```

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

```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
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

(base) PS C:\Users\pujar>  c:; cd 'c:\Users\pujar'; & 'C:\Users\pujar\anaconda3\python.exe' 'c:\Users\pujar\.vscode\extensions\ms-py
y-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '64821' '--' 'c:\Users\pujar\ai..4.py'
'' -> True (expected True)
'a' -> True (expected True)
'A' -> True (expected True)
'ab' -> False (expected False)
'aba' -> True (expected True)
All tests passed
(base) PS C:\Users\pujar>  c:; cd 'c:\Users\pujar'; & 'C:\Users\pujar\anaconda3\python.exe' 'c:\Users\pujar\.vscode\extensions\ms-py
ar\ai...5.py'
All tests passed
(base) PS C:\Users\pujar>
```

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

Name : Pujari Pranay Kumar Hall:No : 2303A51829 Batch:26

C: > Users > pujar > ai...6.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

```
(base) PS C:\Users\pujar>  c:; cd 'c:\Users\pujar'; & 'C:\Users\pujar\anacon
' '65369' '--' 'c:\Users\pujar\ai...5.py'
All tests passed
(base) PS C:\Users\pujar>  c:; cd 'c:\Users\pujar'; & 'C:\Users\pujar\anacon
' '57123' '--' 'c:\Users\pujar\ai...6.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
(base) PS C:\Users\pujar>
```

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.