SCHOOL OF COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE		DEPARTMENT OF COMPUTER SCIENCE ENGINEERING	
ProgramName: M. Tech		**Assignment Type: Lab**	**AcademicYear:** 2025-2026
CourseCoordinatorName		Venkataramana Veeramsetty	
CourseCode		**CourseTitle**	AI Assisted Problem Solving Using Python
Year/Sem	II/I	**Regulation**	R24
Date and Day of Assignment	10.11.2025	**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	Week4 - Wednesday

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**	

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 @.

PROMPT: Generate test cases for is_valid_email(email) and implement the function. The email must contain @ and ., must not start or end with special characters, and should not allow multiple @.

TASK 1.py C:\...\LAB 7 | TASK 1.py LAB 8

Mohammed Farnas Ali Mudabbir > LAB 8 > TASK 1.py > is_valid_email

```python
def is_valid_email(email):
    # Must contain @ and .
    if '@' not in email or '.' not in email:
        return False

    # Should not allow multiple @
    if email.count('@') > 1:
        return False

    # Should not start or end with special characters
    if not email[0].isalnum() or not email[-1].isalnum():
        return False

    return True


# --- User Input ---
email = input("Enter your email: ")

# --- Output ---
if is_valid_email(email):
    print("Valid Email ✅")
else:
    print("Invalid Email ❌")

```

Expected Output#1

- Email validation logic passing all test cases

Practical output:

```
PS C:\Users\rimsha\OneDrive\Desktop\Mohammed Farnas Ali Mudabbir> python -u "c:\Users\rimsha\OneDrive\Desktop\Mohammed Farnas Ali Mudabbir\tempCodeRunnerFile.python"
Enter your email: farnas@gmail.com
Valid Email ✅
PS C:\Users\rimsha\OneDrive\Desktop\Mohammed Farnas Ali Mudabbir> python -u "c:\Users\rimsha\OneDrive\Desktop\Mohammed Farnas Ali Mudabbir\tempCodeRunnerFile.python"
Enter your email: @domain.com
Invalid Email ❌
PS C:\Users\rimsha\OneDrive\Desktop\Mohammed Farnas Ali Mudabbir>
```

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").

PROMPT: Write a Python program to take user input for score and assign grade (A–F). Handle invalid inputs and show test cases.

TASK 1.py C:\...\LAB 7 | TASK 1.py LAB 8 | TASK 2.py

Mohammed Farnas Ali Mudabbir > LAB 8 > TASK 2.py > ...

```python
def assign_grade(score):
    # Check if score is a number and within valid range
    if not isinstance(score, (int, float)):
        return "Invalid input"
    if score < 0 or score > 100:
        return "Invalid score"

    # Grade conditions
    if score >= 90:
        return "A"
    elif score >= 80:
        return "B"
    elif score >= 70:
        return "C"
    elif score >= 60:
        return "D"
    else:
        return "F"


# --- User Input ---
try:
    score = float(input("Enter your marks: "))
    print("Your Grade:", assign_grade(score))
except:
    print("Invalid input ❌ (Please enter a number)")

```

Expected Output#2

Grade assignment function passing test suite

Practical output:

```
PS C:\Users\rimsha\OneDrive\Desktop\Mohammed Farnas Ali Mudabbir> python -u "c:\Users\rimsha\OneDrive\Desktop\Mohammed Farnas Ali
 Mudabbir\tempCodeRunnerFile.python"
Enter your marks: 80
Your Grade: B
PS C:\Users\rimsha\OneDrive\Desktop\Mohammed Farnas Ali Mudabbir> python -u "c:\Users\rimsha\OneDrive\Desktop\Mohammed Farnas Ali
 Mudabbir\tempCodeRunnerFile.python"
Enter your marks: 67
Your Grade: D
PS C:\Users\rimsha\OneDrive\Desktop\Mohammed Farnas Ali Mudabbir> python -u "c:\Users\rimsha\OneDrive\Desktop\Mohammed Farnas Ali
 Mudabbir\tempCodeRunnerFile.python"
Enter your marks: 42
Your Grade: F
PS C:\Users\rimsha\OneDrive\Desktop\Mohammed Farnas Ali Mudabbir> python -u "c:\Users\rimsha\OneDrive\Desktop\Mohammed Farnas Ali
 Mudabbir\tempCodeRunnerFile.python"
Enter your marks: 92
Your Grade: A
PS C:\Users\rimsha\OneDrive\Desktop\Mohammed Farnas Ali Mudabbir>
```

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

PROMPT: Write a Python program to check if a sentence is a palindrome, ignoring case, spaces, and punctuation.

TASK 1.py C:\...\LAB 7 | TASK 1.py LAB 8 | TASK 2.py | TASK 3.py

Mohammed Farnas Ali Mudabbir > LAB 8 > TASK 3.py > ...

```python
def is_sentence_palindrome(sentence):
    # Remove spaces, punctuation, and convert to lowercase
    cleaned = ''.join(ch.lower() for ch in sentence if ch.isalnum())

    # Check palindrome condition
    return cleaned == cleaned[::-1]


# --- User Input ---
sentence = input("Enter a sentence: ")

if is_sentence_palindrome(sentence):
    print("✅ It's a palindrome!")
else:
    print("❌ Not a palindrome.")

```

Expected Output#3

- Function returns True/False for cleaned sentences

Implement the function to pass AI-generated tests

Practical output:

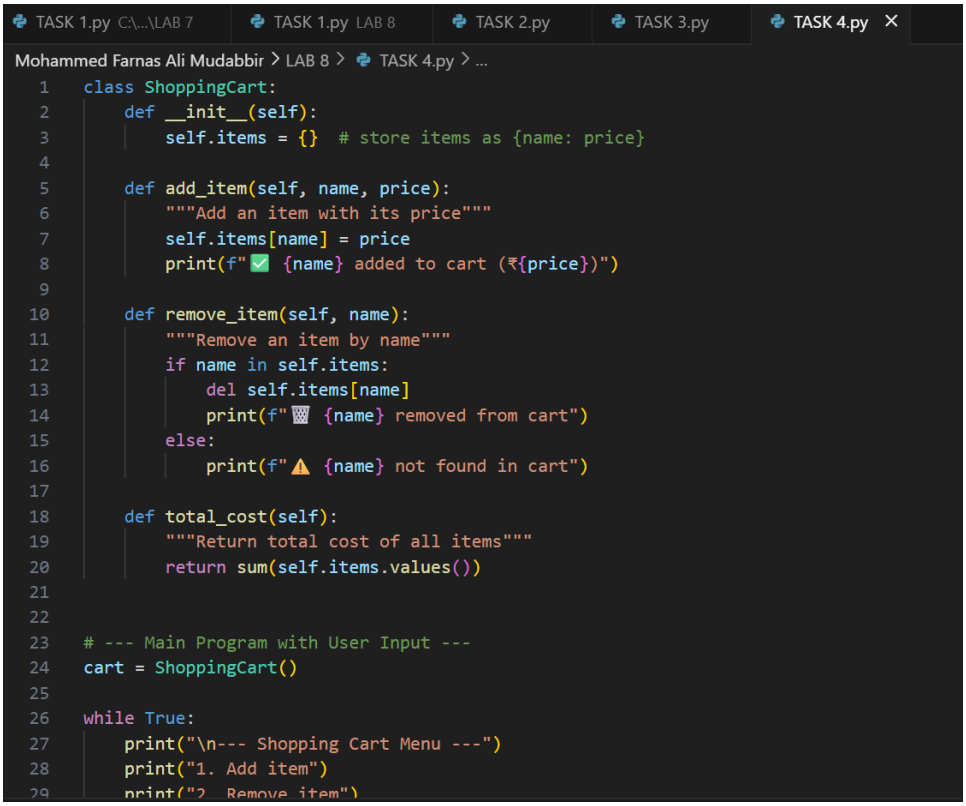
```
PS C:\Users\rimsha\OneDrive\Desktop\Mohammed Farnas Ali Mudabbir> python -u "c:\Users\rimsha\OneDrive\Desktop\Mohammed Farnas Ali Mudabbir\tempCodeRunnerFile.python"
Enter a sentence: LEVEL
✅ It's a palindrome!
PS C:\Users\rimsha\OneDrive\Desktop\Mohammed Farnas Ali Mudabbir> python -u "c:\Users\rimsha\OneDrive\Desktop\Mohammed Farnas Ali Mudabbir\tempCodeRunnerFile.python"
Enter a sentence: HELLO WORLD
❌ Not a palindrome.
PS C:\Users\rimsha\OneDrive\Desktop\Mohammed Farnas Ali Mudabbir> python -u "c:\Users\rimsha\OneDrive\Desktop\Mohammed Farnas Ali Mudabbir\tempCodeRunnerFile.python"
Enter a sentence: Never odd or even
✅ It's a palindrome!
PS C:\Users\rimsha\OneDrive\Desktop\Mohammed Farnas Ali Mudabbir>
```

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()

PROMPT: Write a Python program for a ShoppingCart class with methods add_item(name, price), remove_item(name), and total_cost()



TASK 1.py C:\...\LAB 7 | TASK 1.py LAB 8 | TASK 2.py | TASK 3.py | TASK 4.py

Mohammed Farnas Ali Mudabbir > LAB 8 > TASK 4.py > ...

```
28      print("1. Add item")
29      print("2. Remove item")
30      print("3. View total cost")
31      print("4. Exit")
32
33      choice = input("Enter your choice (1-4): ")
34
35      if choice == "1":
36          name = input("Enter item name: ")
37          try:
38              price = float(input("Enter item price: "))
39              cart.add_item(name, price)
40          except:
41              print("❌ Invalid price! Please enter a number.")
42
43      elif choice == "2":
44          name = input("Enter item name to remove: ")
45          cart.remove_item(name)
46
47      elif choice == "3":
48          print(f"💰 Total cost of items in cart: ₹{cart.total_cost()}")
49
50      elif choice == "4":
51          print("🛒 Thank you for shopping! Goodbye 👋")
52          break
53
54      else:
55          print("⚠️ Invalid choice, please enter 1-4.")
```

Expected Output#4

- Full class with tested functionalities

Practical output:

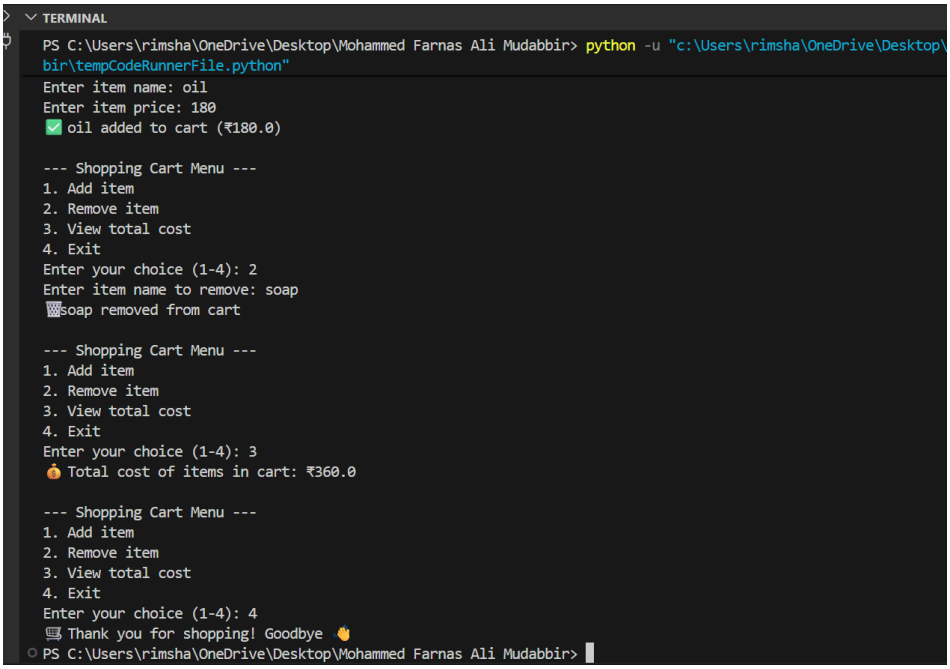
TERMINAL

```
PS C:\Users\rimsha\OneDrive\Desktop\Mohammed Farnas Ali Mudabbir> python -u "c:\Users\rimsha\OneDrive\D
bir\tempCodeRunnerFile.python"

--- Shopping Cart Menu ---
1. Add item
2. Remove item
3. View total cost
4. Exit
Enter your choice (1-4): 1
Enter item name: fruits
Enter item price: 180
✅ fruits added to cart (₹180.0)

--- Shopping Cart Menu ---
1. Add item
2. Remove item
3. View total cost
4. Exit
Enter your choice (1-4): 1
Enter item name: soap
Enter item price: 80
✅ soap added to cart (₹80.0)

--- Shopping Cart Menu ---
1. Add item
2. Remove item
3. View total cost
4. Exit
Enter your choice (1-4): 1
Enter item name: oil
Enter item price: 180
```



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"

PROMPT: Write a Python program to convert date from "YYYY-MM-DD" to "DD-MM-YYYY"

C:\...\LAB 7 | TASK 1.py LAB 8 | TASK 2.py | TASK 3.py | TASK 4.py | def convert

```python
def convert_date_format(date_str):
    # Split the date into parts
    parts = date_str.split('-')

    # Ensure 3 parts: year, month, day
    if len(parts) != 3:
        return "Invalid format ❌"

    year, month, day = parts

    # Validate numeric parts
    if not (year.isdigit() and month.isdigit() and day.isdigit()):
        return "Invalid date ❌"

    # Check valid lengths
    if len(year) != 4 or len(month) != 2 or len(day) != 2:
        return "Invalid format ❌"

    # Return converted format
    return f"{day}-{month}-{year}"


# --- User Input ---
date_str = input("Enter date in YYYY-MM-DD format: ")
print("Converted Date Format:", convert_date_format(date_str))
```

Expected Output#5

- Function converts input format correctly for all test cases

Practical output:

```
PS C:\Users\rimsha\OneDrive\Desktop\Mohammed Farnas Ali Mudabbir> python -u "c:\Users\rimsha\OneDrive\Desktop\Mohammed
bir\tempCodeRunnerFile.python"
Enter date in YYYY-MM-DD format: 2003-04-21
Converted Date Format: 21-04-2003
PS C:\Users\rimsha\OneDrive\Desktop\Mohammed Farnas Ali Mudabbir> python -u "c:\Users\rimsha\OneDrive\Desktop\Mohammed
bir\tempCodeRunnerFile.python"
Enter date in YYYY-MM-DD format: 2003-02-10
Converted Date Format: 10-02-2003
PS C:\Users\rimsha\OneDrive\Desktop\Mohammed Farnas Ali Mudabbir>
```