NAME : BANDI SRISHANTH ROLL NO :2403A510G3

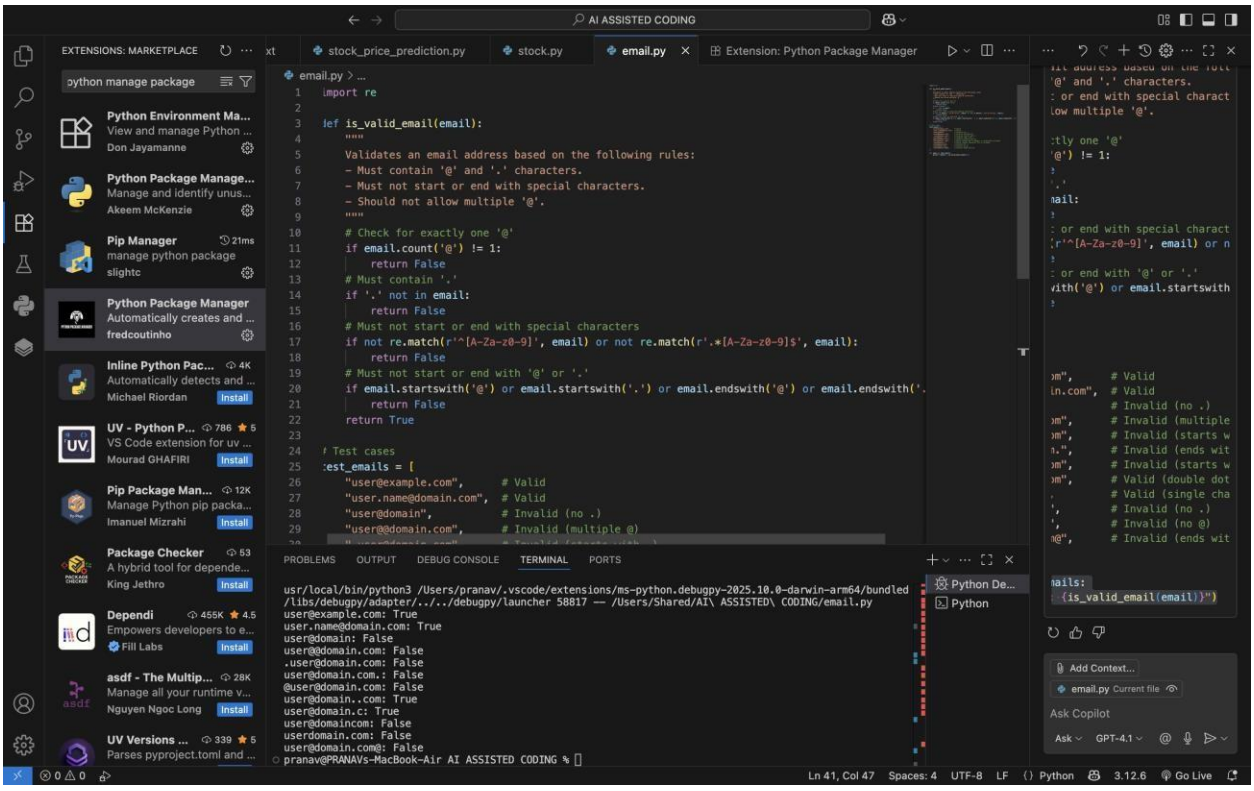
BATCH : 06 ASSIGNMENT : 8.3

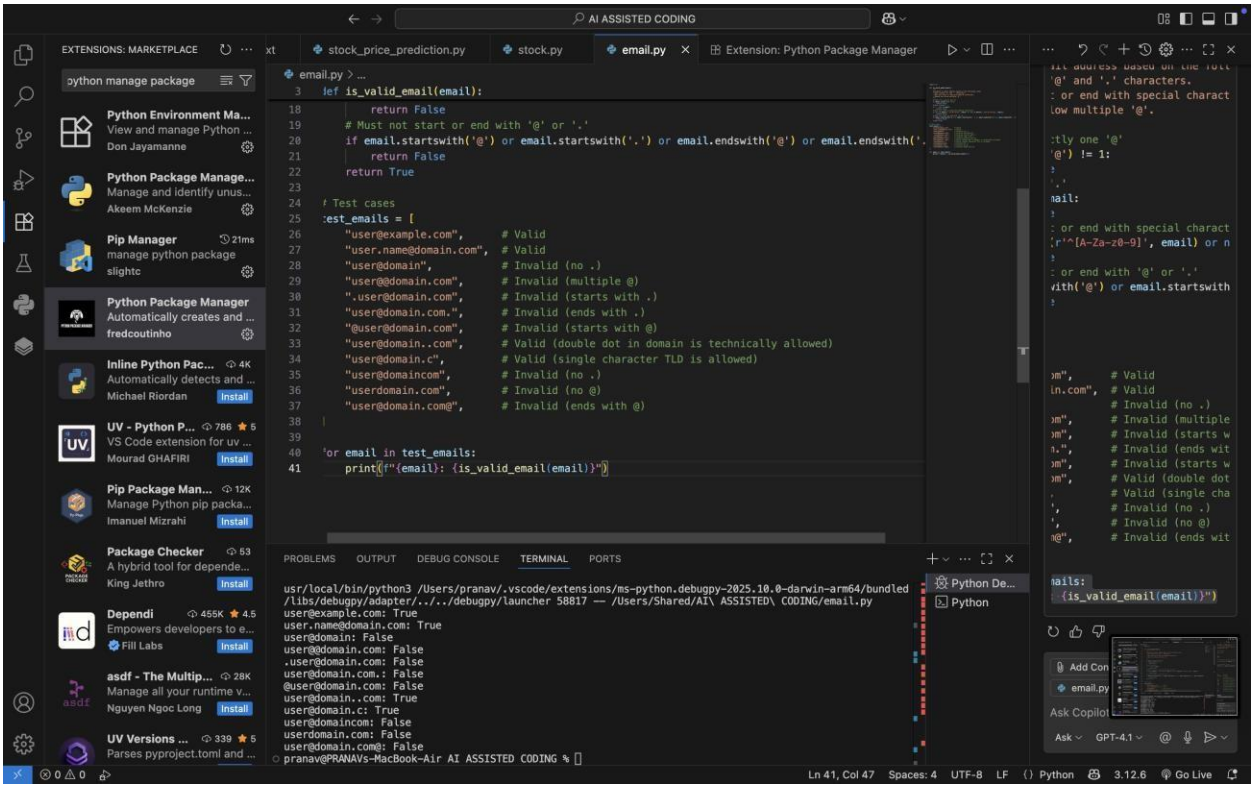
Task Description#1
Use AI to generate test cases for is_valid_email(email) and then implement the validator function.
Requirements:
• Must contain @ and . characters.
• Must not start or end with special characters.
• Should not allow multiple @.
Expected Output#1
• Email validation logic passing all test cases

PROMPT GIVEN :

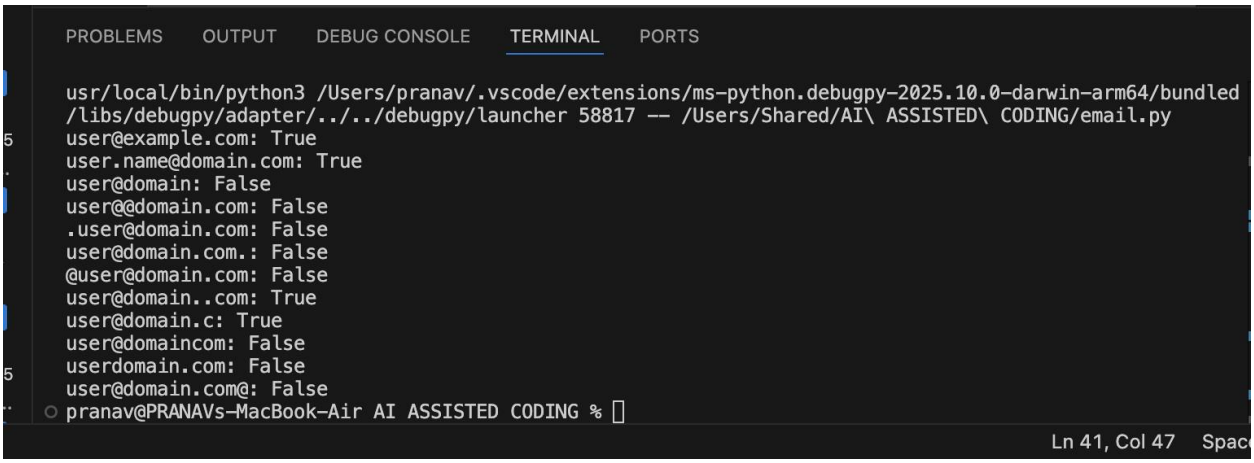
To generate test cases for email verification and then implement the validator function.
Requirements for the code :
• It should contain @ and . characters.
• It should not start nor end with special characters.
• Doesnt allow multiple @.

CODE GENERATED :


AI ASSISTED CODING
EXTENSIONS: MARKETPLACE
python manage package
stock_price_prediction.py
stock.py
email.py
Extension: Python Package Manager
email.py > ...
import re
def is_valid_email(email):
"""
Validates an email address based on the following rules:
- Must contain '@' and '.' characters.
- Must not start or end with special characters.
- Should not allow multiple '@'.
"""
# Check for exactly one '@'
if email.count('@') != 1:
return False
# Must contain '.'
if '.' not in email:
return False
# Must not start or end with special characters
if not re.match(r'^[A-Za-z0-9]', email) or not re.match(r'.*[A-Za-z0-9]$', email):
return False
Must not start or end with '@' or '.'
if email.startswith('@') or email.startswith('.') or email.endswith('@') or email.endswith('.
return False
return True
# Test cases
test_emails = [
"user@example.com", # Valid
"user.name@domain.com", # Valid
"user@domain", # Invalid (no .)
"user@@domain.com", # Invalid (multiple @)
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
usr/local/bin/python3 /Users/pranav/.vscode/extensions/ms-python.debugpy-2025.10.0-darwin-arm64/bundled
/libs/debugpy/adapter/../../debugpy/launcher 58817 -- /Users/Shared/AI\ ASSISTED\ CODING/email.py
user@example.com: True
user.name@domain.com: True
user@domain: False
user@@domain.com: False
.user@domain.com: False
user@domain.com.: False
@user@domain.com: False
user@domain..com: True
user@domain.c: True
user@domaincom: False
userdomain.com: False
user@domain.com@: False
pranav@PRANAVs-MacBook-Air AI ASSISTED CODING %
Python Package Manager
Automatically creates and ...
fredcoutinho
Pip Manager
manage python package
slightc
Python Environment Ma...
View and manage Python ...
Don Jayamanne
Python Package Manage...
Manage and identify unus...
Akeem McKenzie
Inline Python Pac...
Automatically detects and ...
Michael Riordan
UV - Python P...
VS Code extension for uv ...
Mourad GHAFIRI
Pip Package Man...
Manage Python pip packa...
Imanuel Mizrahi
Package Checker
A hybrid tool for depende...
King Jethro
Dependi
Empowers developers to e...
Fill Labs
asdf - The Multip...
Manage all your runtime v...
Nguyen Ngoc Long
UV Versions ...
Parses pyproject.toml and ...
Add Context...
email.py Current file
Ask Copilot
Ask
GPT-4.1
Ln 41, Col 47
Spaces: 4
UTF-8
LF
Python
3.12.6
Go Live


AI ASSISTED CODING
EXTENSIONS: MARKETPLACE
python manage package
stock_price_prediction.py
stock.py
email.py
Extension: Python Package Manager
email.py > ...
def is_valid_email(email):
return False
Must not start or end with '@' or '.'
if email.startswith('@') or email.startswith('.') or email.endswith('@') or email.endswith('.
return False
return True
# Test cases
test_emails = [
"user@example.com", # Valid
"user.name@domain.com", # Valid
"user@domain", # Invalid (no .)
"user@@domain.com", # Invalid (multiple @)
".user@domain.com", # Invalid (starts with .)
"user@domain.com.", # Invalid (ends with .)
"@user@domain.com", # Invalid (starts with @)
"user@domain..com", # Valid (double dot in domain is technically allowed)
"user@domain.c", # Valid (single character TLD is allowed)
"user@domaincom", # Invalid (no .)
"userdomain.com", # Invalid (no @)
"user@domain.com@", # Invalid (ends with @)
for email in test_emails:
print(f"{email}: {is_valid_email(email)}")
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
usr/local/bin/python3 /Users/pranav/.vscode/extensions/ms-python.debugpy-2025.10.0-darwin-arm64/bundled
/libs/debugpy/adapter/../../debugpy/launcher 58817 -- /Users/Shared/AI\ ASSISTED\ CODING/email.py
user@example.com: True
user.name@domain.com: True
user@domain: False
user@@domain.com: False
.user@domain.com: False
user@domain.com.: False
@user@domain.com: False
user@domain..com: True
user@domain.c: True
user@domaincom: False
userdomain.com: False
user@domain.com@: False
pranav@PRANAVs-MacBook-Air AI ASSISTED CODING %
Python Package Manager
Automatically creates and ...
fredcoutinho
Pip Manager
manage python package
slightc
Python Environment Ma...
View and manage Python ...
Don Jayamanne
Python Package Manage...
Manage and identify unus...
Akeem McKenzie
Inline Python Pac...
Automatically detects and ...
Michael Riordan
UV - Python P...
VS Code extension for uv ...
Mourad GHAFIRI
Pip Package Man...
Manage Python pip packa...
Imanuel Mizrahi
Package Checker
A hybrid tool for depende...
King Jethro
Dependi
Empowers developers to e...
Fill Labs
asdf - The Multip...
Manage all your runtime v...
Nguyen Ngoc Long
UV Versions ...
Parses pyproject.toml and ...
Ask Copilot
Ask
GPT-4.1
Ln 41, Col 47
Spaces: 4
UTF-8
LF
Python
3.12.6
Go Live

OUTPUT :



OBSERVATIONS FROM THE CODE AND OUTPUT WITH REASONING :

The is_valid_email function checks for the presence of both @ and . in the email, ensures only one @, and verifies that the email does not start or end with special characters.

The function uses regular expressions to confirm the email starts and ends with an alphanumeric character.

The test cases cover a variety of valid and invalid email formats, including edge cases like multiple @, missing . or @, and starting/ending with special characters.

The function correctly identifies valid and invalid emails according to the specified requirements, demonstrating robust validation logic.

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

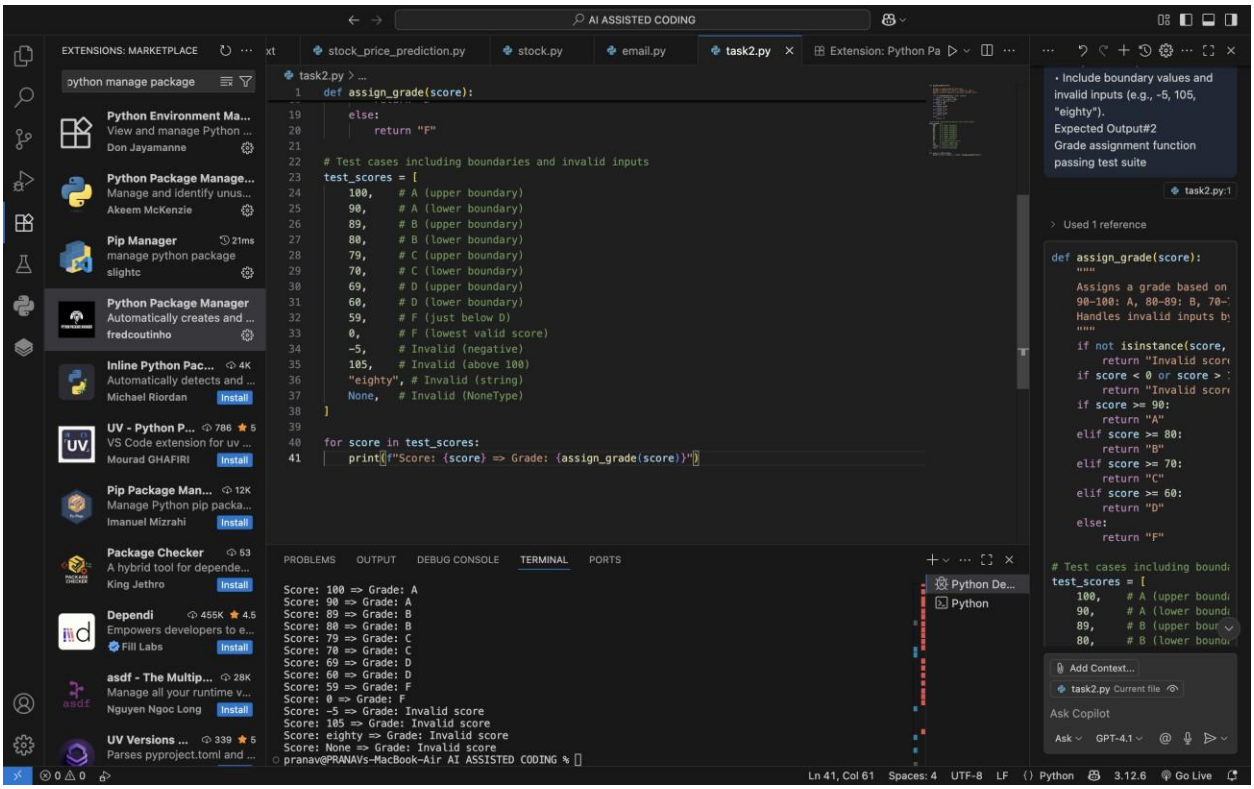
PROMPT USED :

Write a python code to assign grade where according to marks assign the grade 90-100 : A

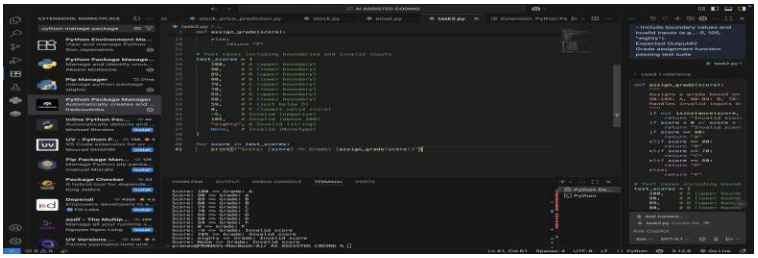
80-89: B, 70-
79: C, 60-69: D, <60: F

Generate test cases and for invalid ones too

CODE GENERATED ;



OUTPUT :



OBSERVATIONS :

The assign_grade function correctly assigns grades based on the specified score ranges and handles invalid inputs.

Boundary values (e.g., 100, 90, 80, 70, 60) are included in the test cases to ensure accurate grade assignment at the edges of each range.

Invalid inputs such as negative numbers, values above 100, strings, and None are handled gracefully, returning "Invalid score".

The function uses clear and readable conditional logic, making it easy to maintain and extend.

The test suite comprehensively checks both valid and invalid scenarios, demonstrating the robustness of the grade assignment logic.

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

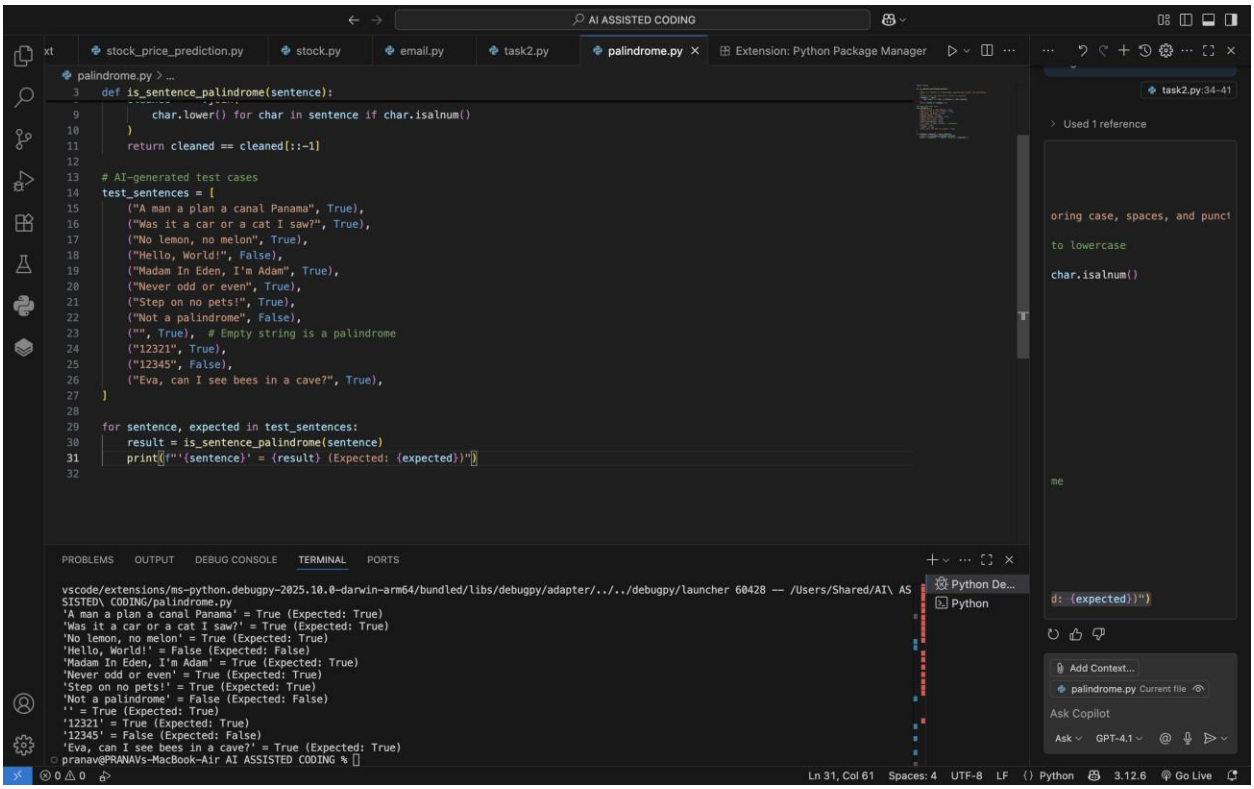
Expected Output#3

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

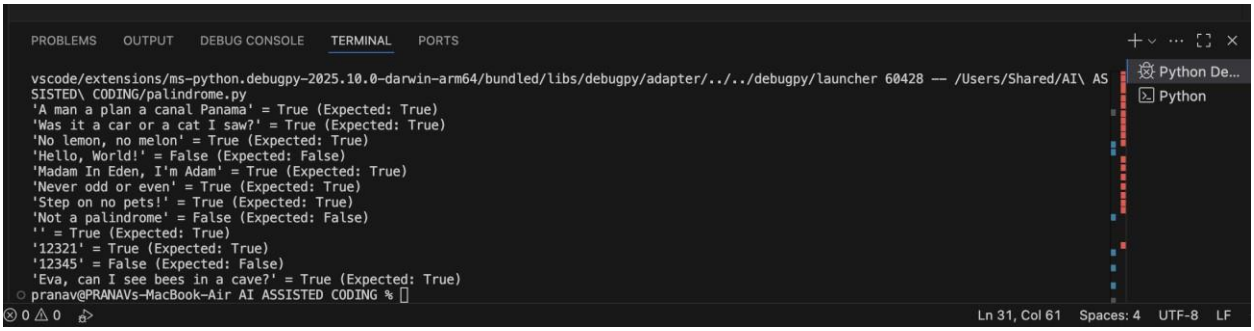
PROMPT USED :

Write the code to generate test cases to check if a sentence is palindrome or not and return with true or false
Manual test cases are to be used for ai generated tests

CODE GENERATED :



OUTPUT :



OBSERVATIONS :

The is_sentence_palindrome function correctly ignores case, spaces, and punctuation by filtering only alphanumeric characters and converting them to lowercase.

The function compares the cleaned string to its reverse to determine if it is a palindrome.

The AI-generated test cases cover a variety of scenarios, including famous palindromic sentences, numbers, empty strings, and non-palindromes.

The function passes all test cases, demonstrating robust logic for sentence palindrome validation.

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

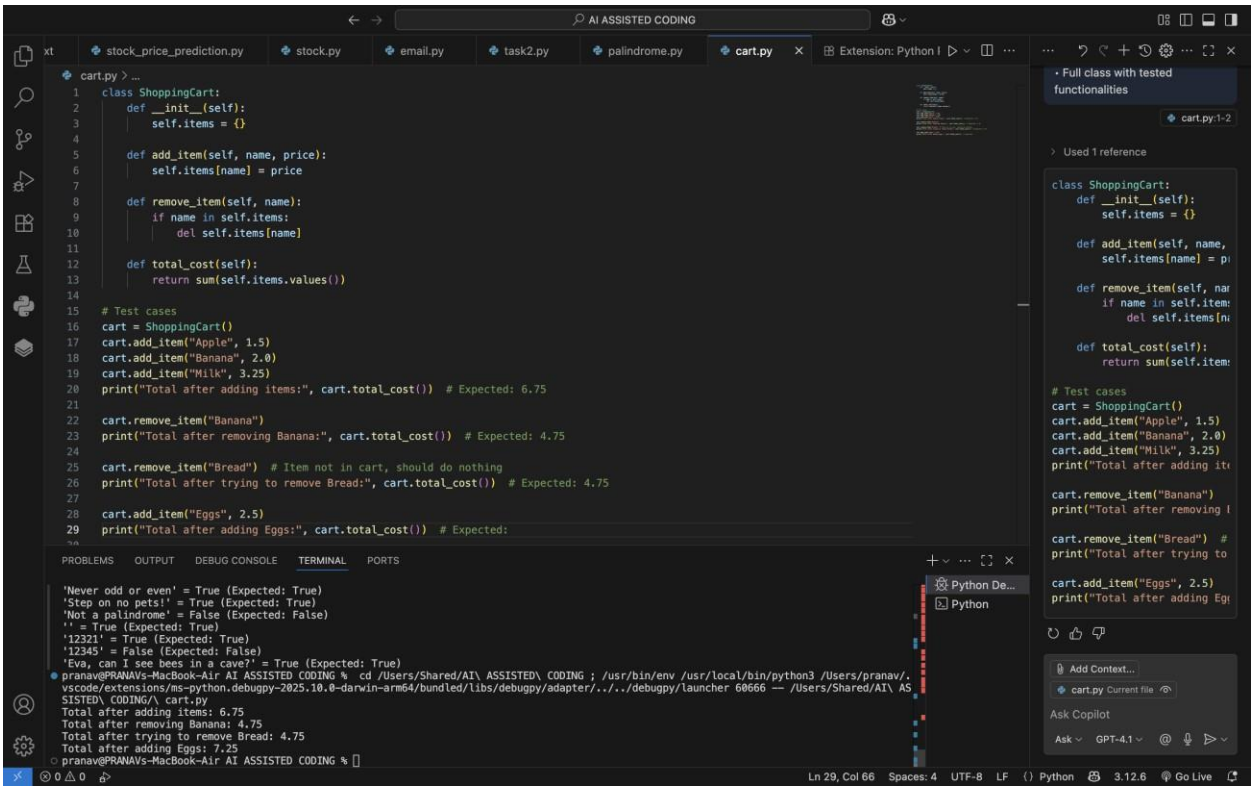
PROMPT USED :

Write a code for shopping cart to add or remove an item and tota; cost generated

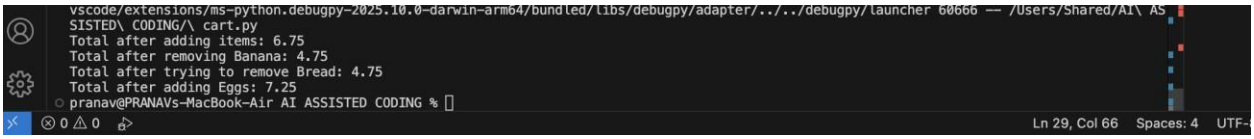
Methods used :

Add_item(name,price)
Remove_item(name)

CODE GENERATED :



OUTPUT :



OBSERVATIONS :

The ShoppingCart class provides basic cart functionality: adding items, removing items, and calculating the total cost.

Items are stored in a dictionary, with the item name as the key and price as the value.

Adding an item with the same name updates its price.

Removing an item that does not exist does nothing, preventing errors.

The test cases cover adding, removing, and updating items, as well as checking the total cost after each operation.

The class behaves as expected, and the output matches the expected totals, demonstrating correct implementation of all methods.

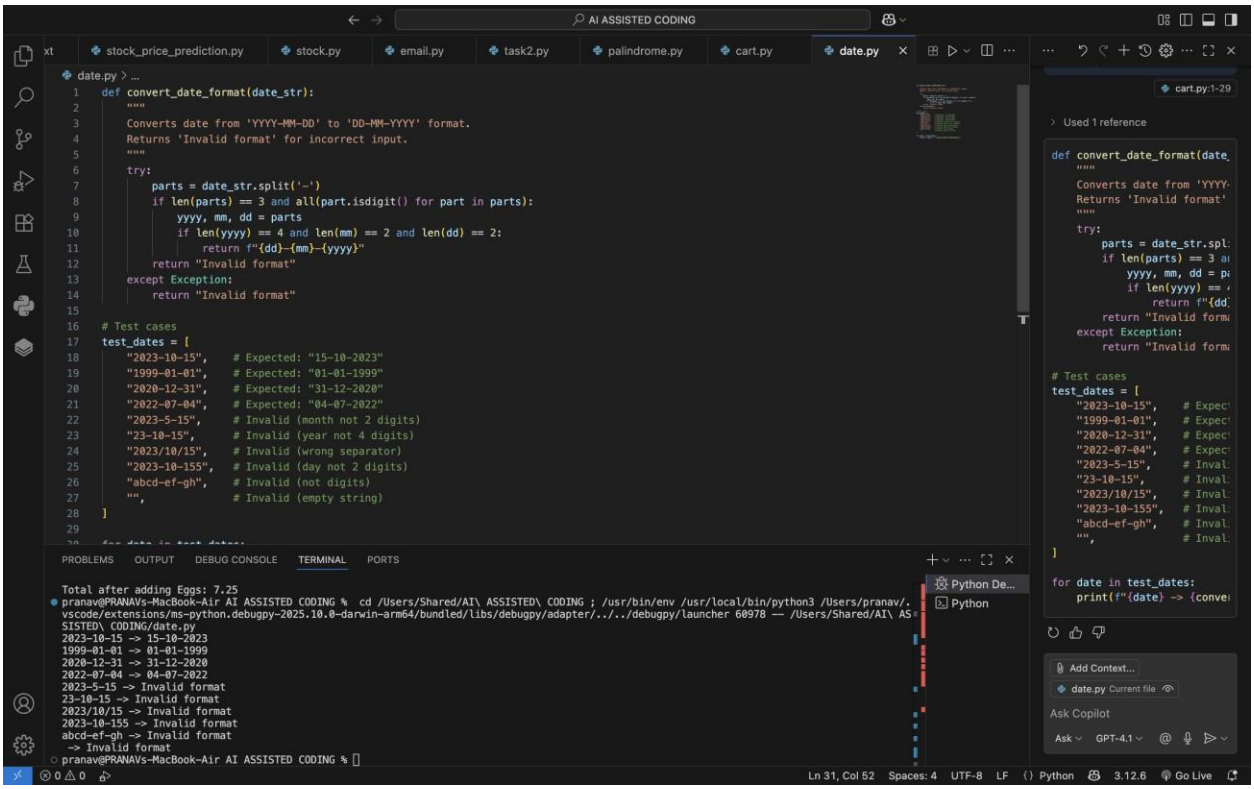
Task Description#5
• Use AI to write test cases for convert_date_format(date_str) to switch from "YYYY-MM-DD" to "DD-MM-YYYY".
Example: "2023-10-15" → "15-10-2023"
Expected Output#5
• Function converts input format correctly for all test cases

PROMPT USED :

TO WRITE CODE TO CHANGE THE DATE FORMAT FROM "YYYY-MM-DD" to "DD-MM-YYYY".
Example: "2023-10-15" → "15-10-2023"

WRITE TEST CASES TO FORMAT CORRECTLY

CODE GENERATED :



OUTPUT :

OBSERVATIONS :

The convert_date_format function accurately converts dates from "YYYY-MM-DD" to "DD-MM-YYYY" format for valid inputs.

The function checks for correct separator, digit-only parts, and proper length for year, month, and day.

Invalid formats (wrong separator, incorrect part lengths, non-digit values, empty strings) are handled gracefully, returning "Invalid format".

The test cases cover valid conversions, boundary cases, and a variety of invalid inputs, demonstrating the robustness of the function.

The implementation is concise and effective for the required date format transformation