LAB-8.3

2303A51525

BATCH-10

Task 1: Email Validation using TDD

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

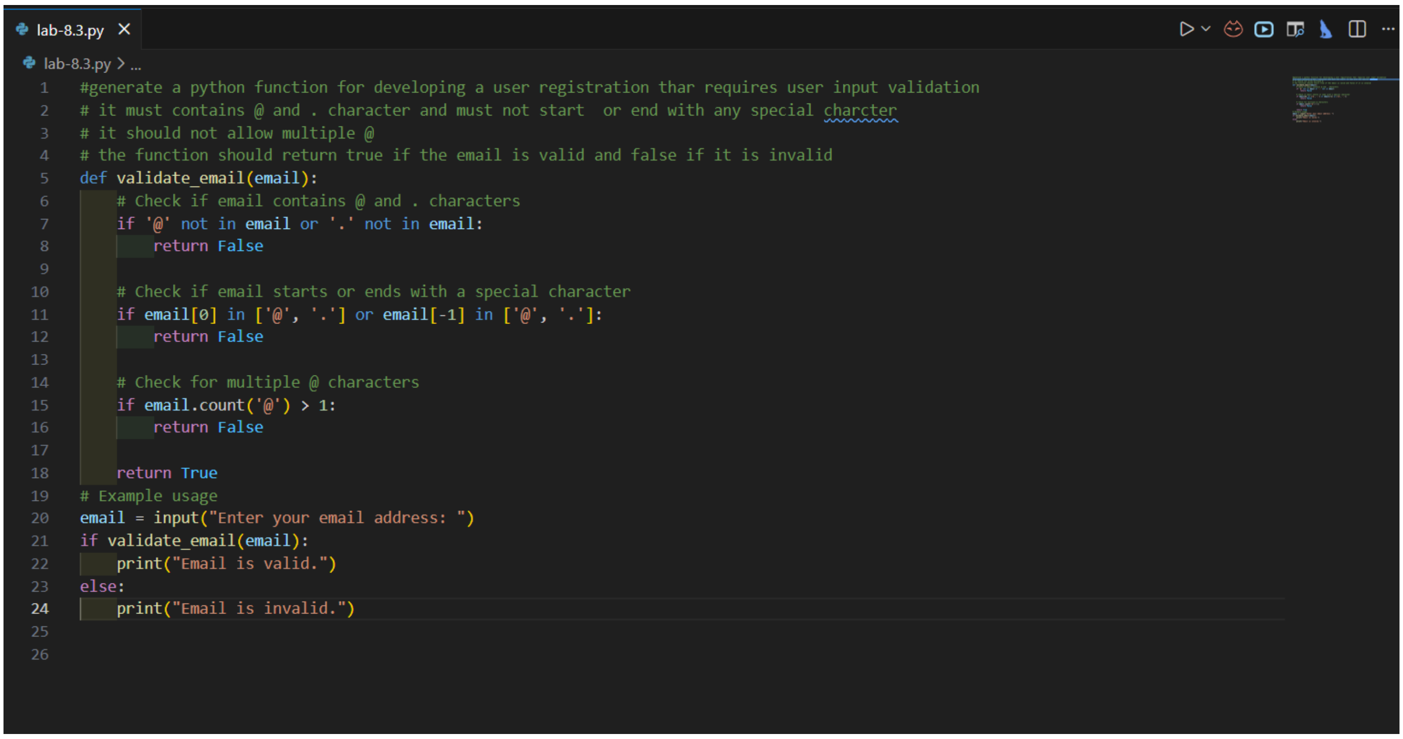
Prompt used:

# generate a python function for developing a user registration thar requires user input validation

# it must contains @ and . character and must not start or end with any special charcter

# it should not allow multiple @

# the function should return true if the email is valid and false if it is invalid.

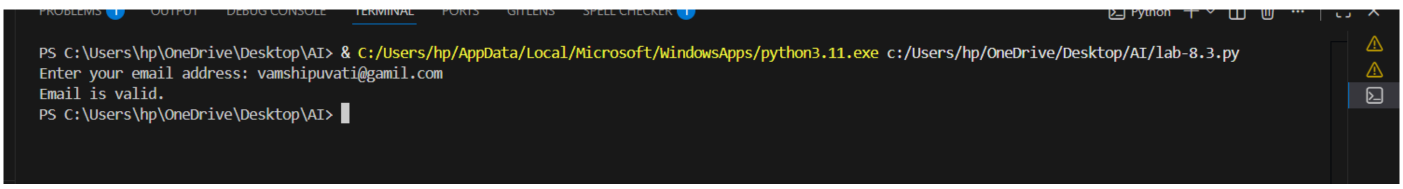


```python
#generate a python function for developing a user registration thar requires user input validation
# it must contains @ and . character and must not start  or end with any special charcter
# it should not allow multiple @
# the function should return true if the email is valid and false if it is invalid
def validate_email(email):
    # Check if email contains @ and . characters
    if '@' not in email or '.' not in email:
        return False

    # Check if email starts or ends with a special character
    if email[0] in ['@', '.'] or email[-1] in ['@', '.']:
        return False

    # Check for multiple @ characters
    if email.count('@') > 1:
        return False

    return True
# Example usage
email = input("Enter your email address: ")
if validate_email(email):
    print("Email is valid.")
else:
    print("Email is invalid.")
```



```
PS C:\Users\hp\OneDrive\Desktop\AI> & C:/Users/hp/AppData/Local/Microsoft/WindowsApps/python3.11.exe c:/Users/hp/OneDrive/Desktop/AI/lab-8.3.py
Enter your email address: vamshipuvati@gamil.com
Email is valid.
PS C:\Users\hp\OneDrive\Desktop\AI>
```

Task 2: Grade Assignment using Loops

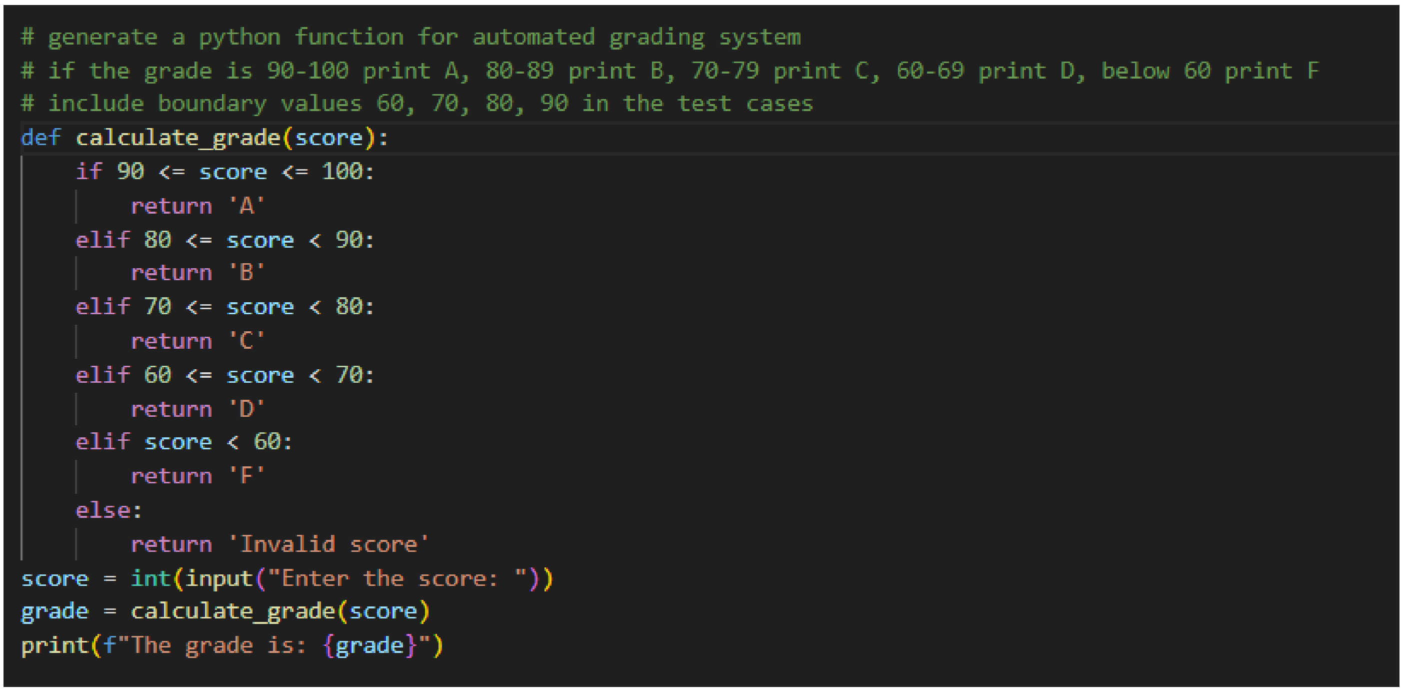
Scenario: You are building an automated grading system for an online examination platform.

Prompt used:

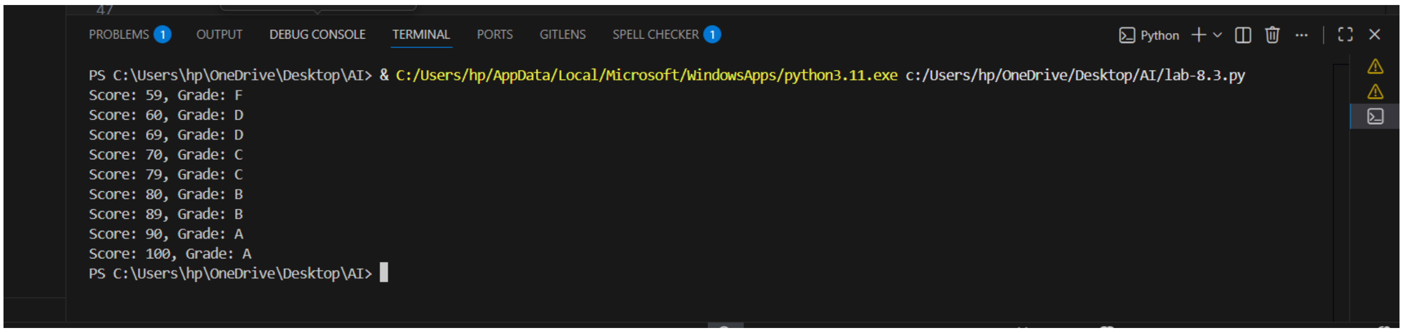
# generate a python function for automated grading system

# if the grade is 90-100 print A, 80-89 print B, 70-79 print C, 60-69 print D, below 60 print F

# include boundary values 60, 70, 80, 90 in the test cases



```python
# generate a python function for automated grading system
# if the grade is 90-100 print A, 80-89 print B, 70-79 print C, 60-69 print D, below 60 print F
# include boundary values 60, 70, 80, 90 in the test cases
def calculate_grade(score):
    if 90 <= score <= 100:
        return 'A'
    elif 80 <= score < 90:
        return 'B'
    elif 70 <= score < 80:
        return 'C'
    elif 60 <= score < 70:
        return 'D'
    elif score < 60:
        return 'F'
    else:
        return 'Invalid score'
score = int(input("Enter the score: "))
grade = calculate_grade(score)
print(f"The grade is: {grade}")
```



```
PS C:\Users\hp\OneDrive\Desktop\AI> & C:/Users/hp/AppData/Local/Microsoft/WindowsApps/python3.11.exe c:/Users/hp/OneDrive/Desktop/AI/lab-8.3.py
Score: 59, Grade: F
Score: 60, Grade: D
Score: 69, Grade: D
Score: 70, Grade: C
Score: 79, Grade: C
Score: 80, Grade: B
Score: 89, Grade: B
Score: 90, Grade: A
Score: 100, Grade: A
PS C:\Users\hp\OneDrive\Desktop\AI>
```

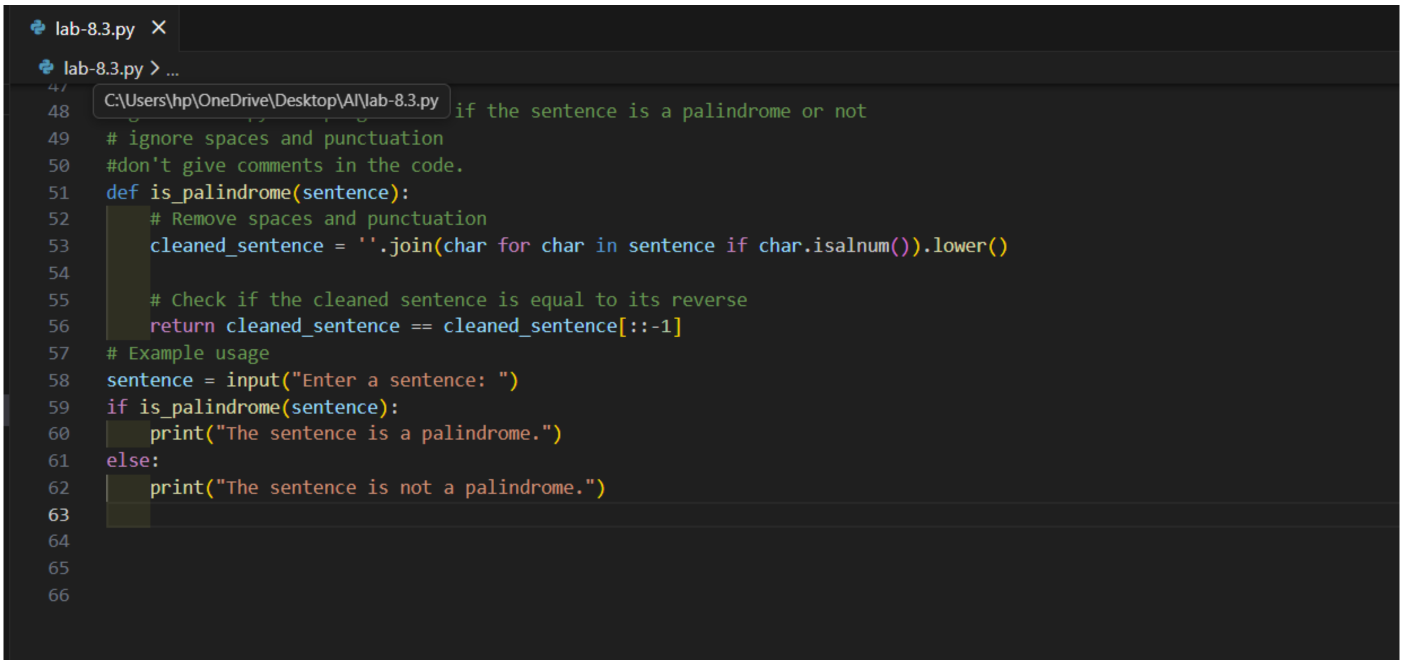
Task 3: Sentence Palindrome Checker

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

Prompt used:

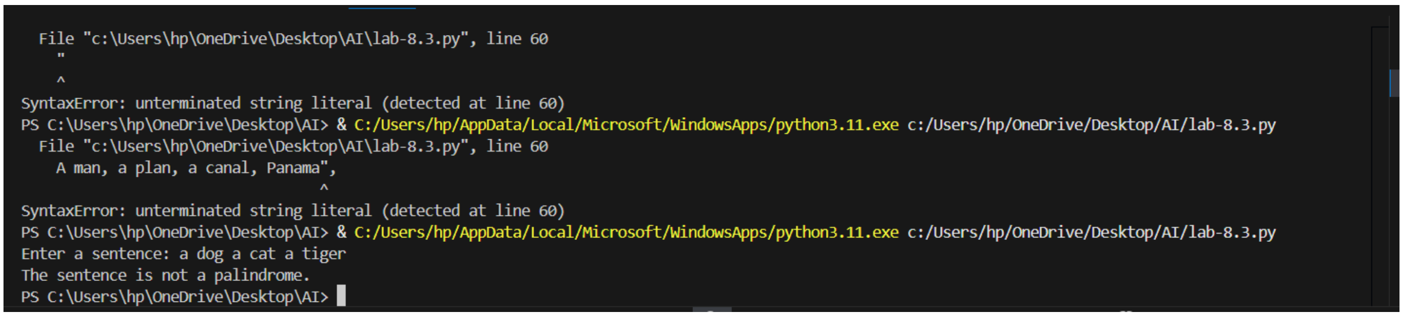
# generate a python program for if the sentence is a palindrome or not

ignore spaces and punctuation



```
lab-8.3.py ×
lab-8.3.py > ...
C:\Users\hp\OneDrive\Desktop\AI\lab-8.3.py  if the sentence is a palindrome or not
# ignore spaces and punctuation
#don't give comments in the code.
def is_palindrome(sentence):
    # Remove spaces and punctuation
    cleaned_sentence = ''.join(char for char in sentence if char.isalnum()).lower()

    # Check if the cleaned sentence is equal to its reverse
    return cleaned_sentence == cleaned_sentence[::-1]
# Example usage
sentence = input("Enter a sentence: ")
if is_palindrome(sentence):
    print("The sentence is a palindrome.")
else:
    print("The sentence is not a palindrome.")
```



```
  File "c:\Users\hp\OneDrive\Desktop\AI\lab-8.3.py", line 60
    "
    ^
SyntaxError: unterminated string literal (detected at line 60)
PS C:\Users\hp\OneDrive\Desktop\AI> & C:/Users/hp/AppData/Local/Microsoft/WindowsApps/python3.11.exe c:/Users/hp/OneDrive/Desktop/AI/lab-8.3.py
  File "c:\Users\hp\OneDrive\Desktop\AI\lab-8.3.py", line 60
    A man, a plan, a canal, Panama",
                                  ^
SyntaxError: unterminated string literal (detected at line 60)
PS C:\Users\hp\OneDrive\Desktop\AI> & C:/Users/hp/AppData/Local/Microsoft/WindowsApps/python3.11.exe c:/Users/hp/OneDrive/Desktop/AI/lab-8.3.py
Enter a sentence: a dog a cat a tiger
The sentence is not a palindrome.
PS C:\Users\hp\OneDrive\Desktop\AI> 
```

Task 4: ShoppingCart Class

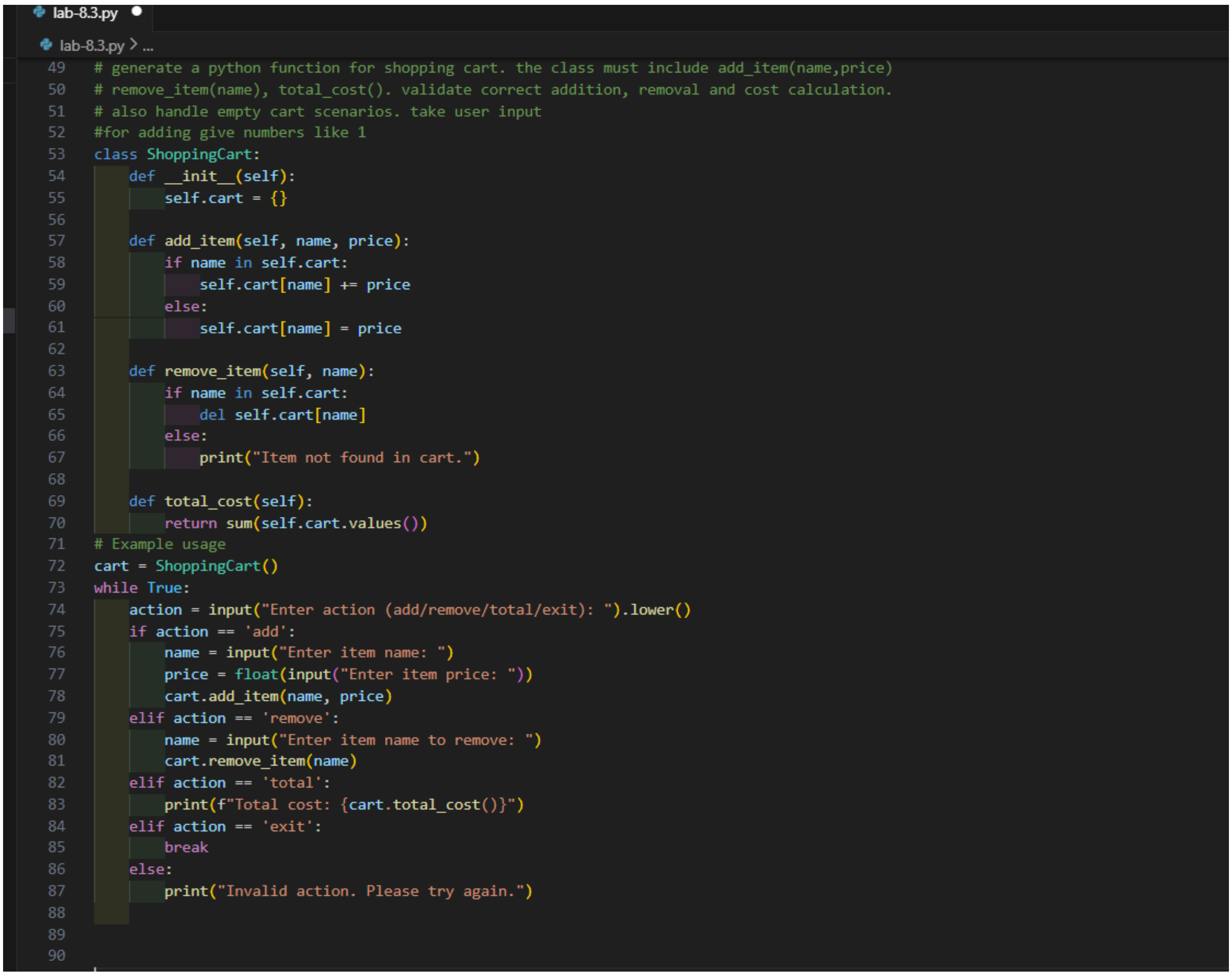
Scenario: You are designing a basic shopping cart module for an e-commerce application.

Prompt used:

# generate a python function for shopping cart. the class must include add_item(name,price)

# remove_item(name), total_cost(). validate correct addition, removal and cost calculation.

# also handle empty cart scenarios. take user input



lab-8.3.py ●

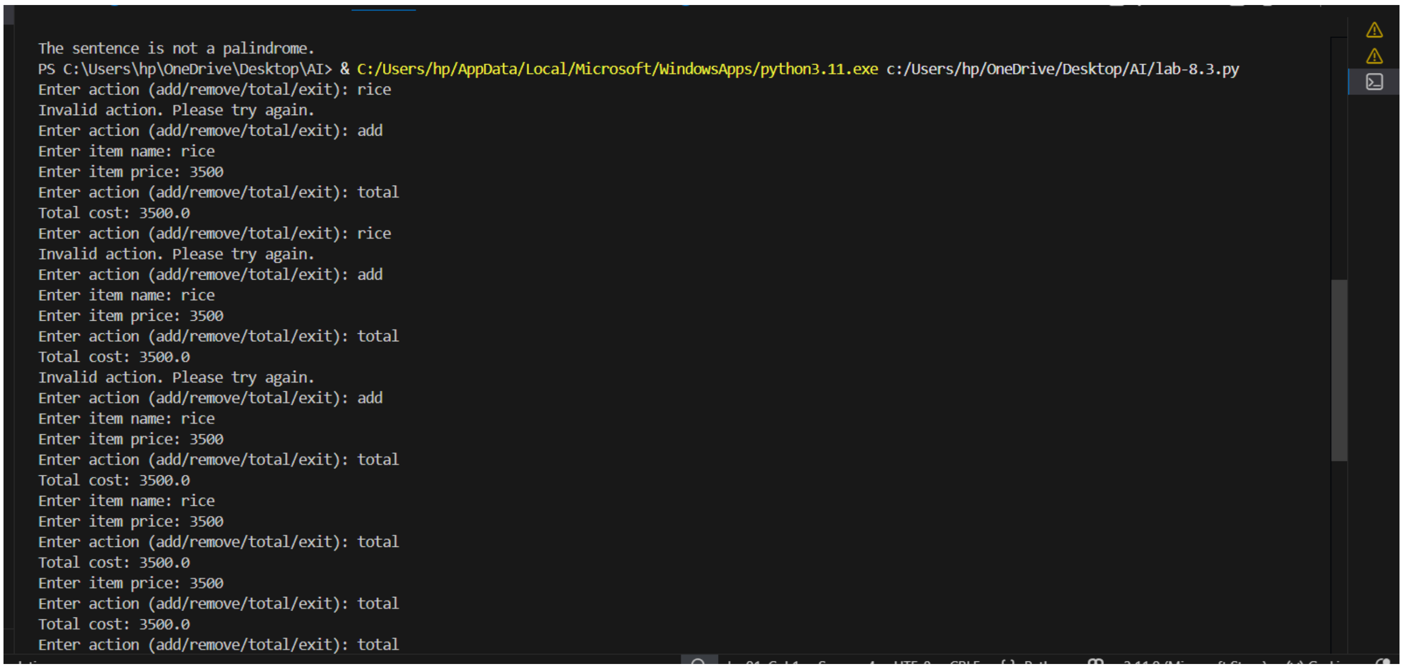
lab-8.3.py > ...

```python
# generate a python function for shopping cart. the class must include add_item(name,price)
# remove_item(name), total_cost(). validate correct addition, removal and cost calculation.
# also handle empty cart scenarios. take user input
#for adding give numbers like 1
class ShoppingCart:
    def __init__(self):
        self.cart = {}

    def add_item(self, name, price):
        if name in self.cart:
            self.cart[name] += price
        else:
            self.cart[name] = price

    def remove_item(self, name):
        if name in self.cart:
            del self.cart[name]
        else:
            print("Item not found in cart.")

    def total_cost(self):
        return sum(self.cart.values())
# Example usage
cart = ShoppingCart()
while True:
    action = input("Enter action (add/remove/total/exit): ").lower()
    if action == 'add':
        name = input("Enter item name: ")
        price = float(input("Enter item price: "))
        cart.add_item(name, price)
    elif action == 'remove':
        name = input("Enter item name to remove: ")
        cart.remove_item(name)
    elif action == 'total':
        print(f"Total cost: {cart.total_cost()}")
    elif action == 'exit':
        break
    else:
        print("Invalid action. Please try again.")
```



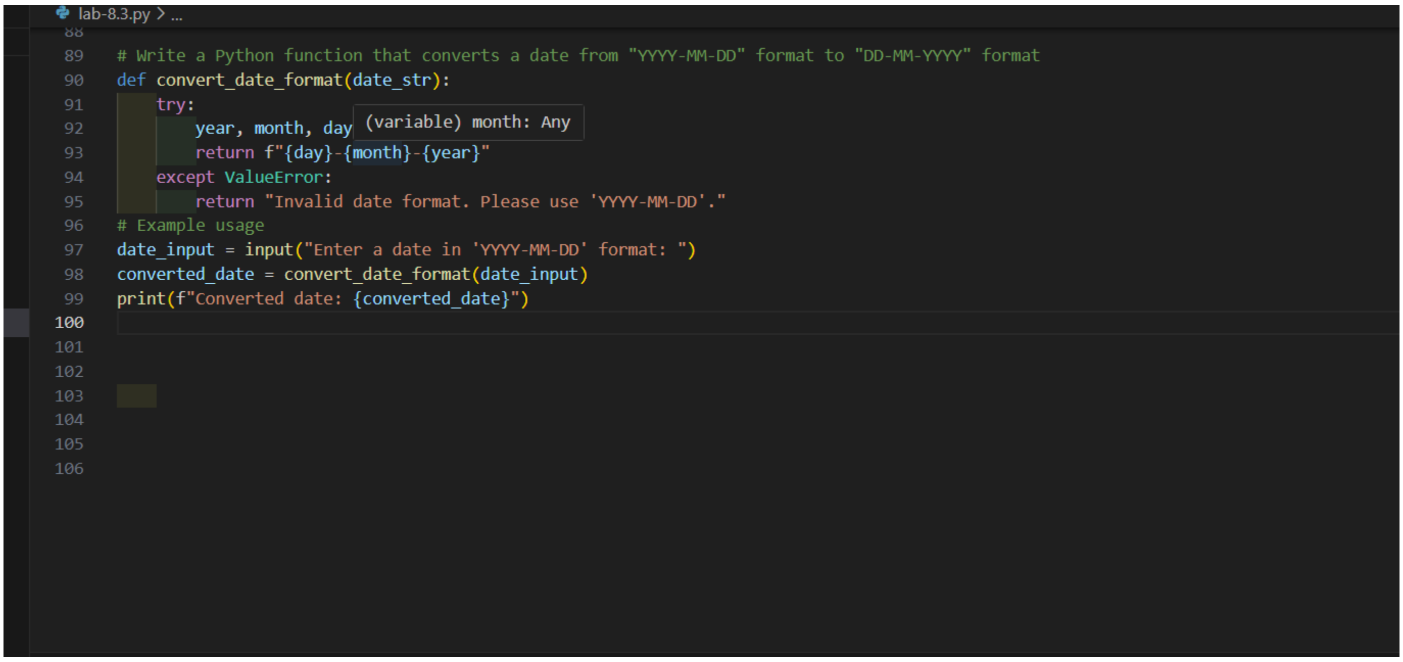
```
The sentence is not a palindrome.
PS C:\Users\hp\OneDrive\Desktop\AI> & C:/Users/hp/AppData/Local/Microsoft/WindowsApps/python3.11.exe c:/Users/hp/OneDrive/Desktop/AI/lab-8.3.py
Enter action (add/remove/total/exit): rice
Invalid action. Please try again.
Enter action (add/remove/total/exit): add
Enter item name: rice
Enter item price: 3500
Enter action (add/remove/total/exit): total
Total cost: 3500.0
Enter action (add/remove/total/exit): rice
Invalid action. Please try again.
Enter action (add/remove/total/exit): add
Enter item name: rice
Enter item price: 3500
Enter action (add/remove/total/exit): total
Total cost: 3500.0
Invalid action. Please try again.
Enter action (add/remove/total/exit): add
Enter item name: rice
Enter item price: 3500
Enter action (add/remove/total/exit): total
Total cost: 3500.0
Enter item name: rice
Enter item price: 3500
Enter action (add/remove/total/exit): total
Total cost: 3500.0
Enter item price: 3500
Enter action (add/remove/total/exit): total
Total cost: 3500.0
Enter action (add/remove/total/exit): total
```

Task 5: Date Format Conversion

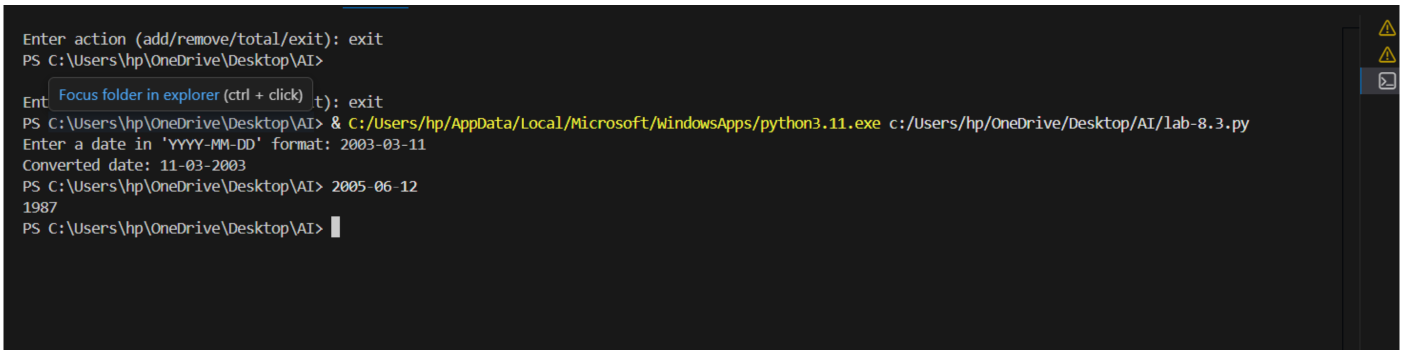
Scenario: You are creating a utility function to convert date formats for reports.

Prompt used:

# Write a Python function that converts a date from "YYYY-MM-DD" format to "DD-MM-YYYY" format



```
# Write a Python function that converts a date from "YYYY-MM-DD" format to "DD-MM-YYYY" format
def convert_date_format(date_str):
    try:
        year, month, day  (variable) month: Any
        return f"{day}-{month}-{year}"
    except ValueError:
        return "Invalid date format. Please use 'YYYY-MM-DD'."
# Example usage
date_input = input("Enter a date in 'YYYY-MM-DD' format: ")
converted_date = convert_date_format(date_input)
print(f"Converted date: {converted_date}")
```



```
Enter action (add/remove/total/exit): exit
PS C:\Users\hp\OneDrive\Desktop\AI>

Ent Focus folder in explorer (ctrl + click) t): exit
PS C:\Users\hp\OneDrive\Desktop\AI> & C:/Users/hp/AppData/Local/Microsoft/WindowsApps/python3.11.exe c:/Users/hp/OneDrive/Desktop/AI/lab-8.3.py
Enter a date in 'YYYY-MM-DD' format: 2003-03-11
Converted date: 11-03-2003
PS C:\Users\hp\OneDrive\Desktop\AI> 2005-06-12
1987
PS C:\Users\hp\OneDrive\Desktop\AI> █
```

Explanation:

1. The first snippet is a function to validate email addresses based on specific criteria.

2. The second snippet is a function to calculate letter grades based on numerical scores.

3. The third snippet checks if a given sentence is a palindrome, ignoring spaces and punctuation.

4. The fourth snippet defines a ShoppingCart class that allows users to add and remove items, calculate total cost, and display the cart contents.

5. The fifth snippet is a function that converts a date from "YYYY-MM-DD" format to "DD-MM-YYYY" format.