ASSIGNMENT - 8.1

Name : Aravind Reddy

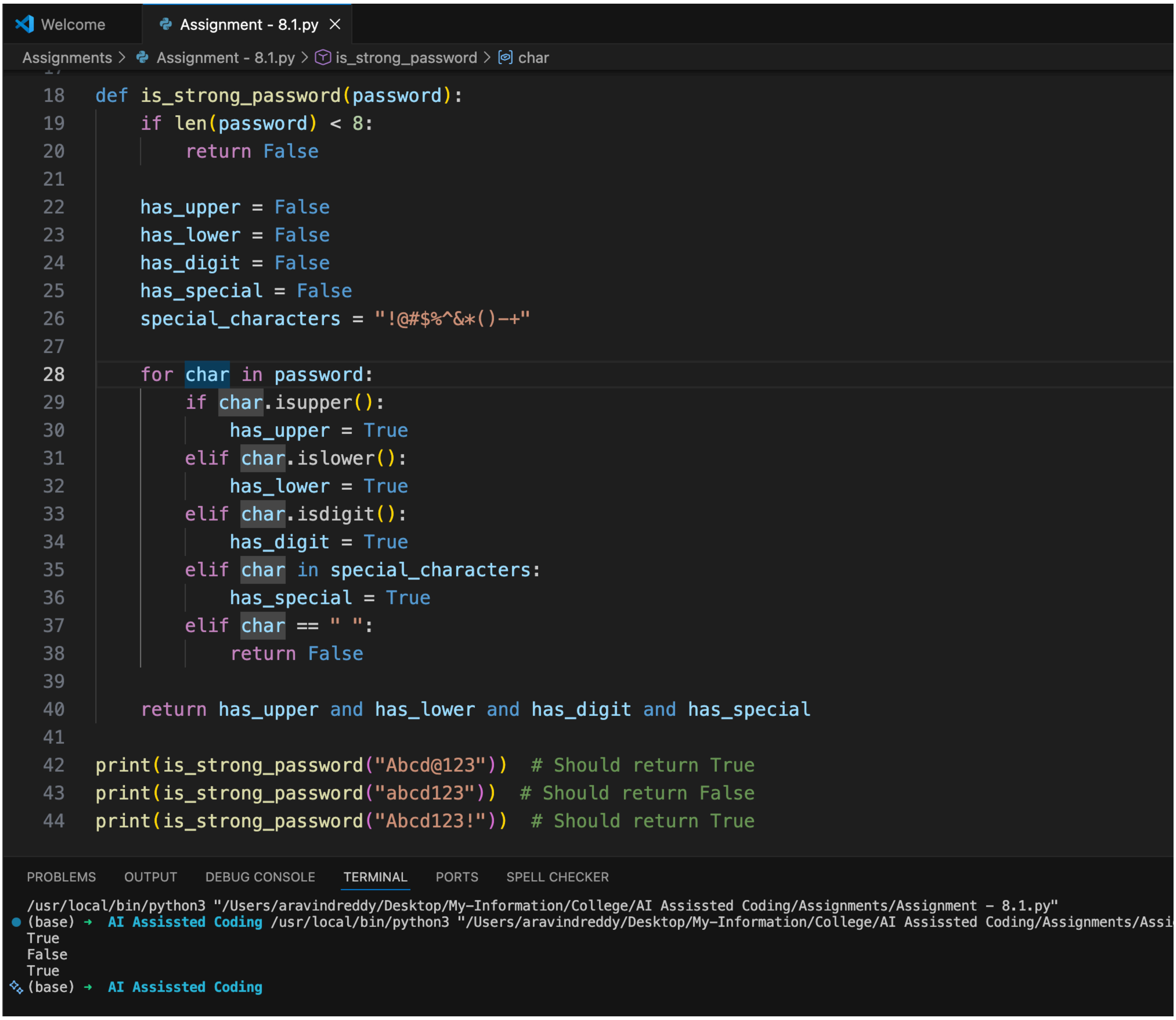
Hall Ticket No : 2303a51027

Batch No : 01

Course Title : AI Assisted Coding

Instructor's Name : Mr. S Naresh Kumar

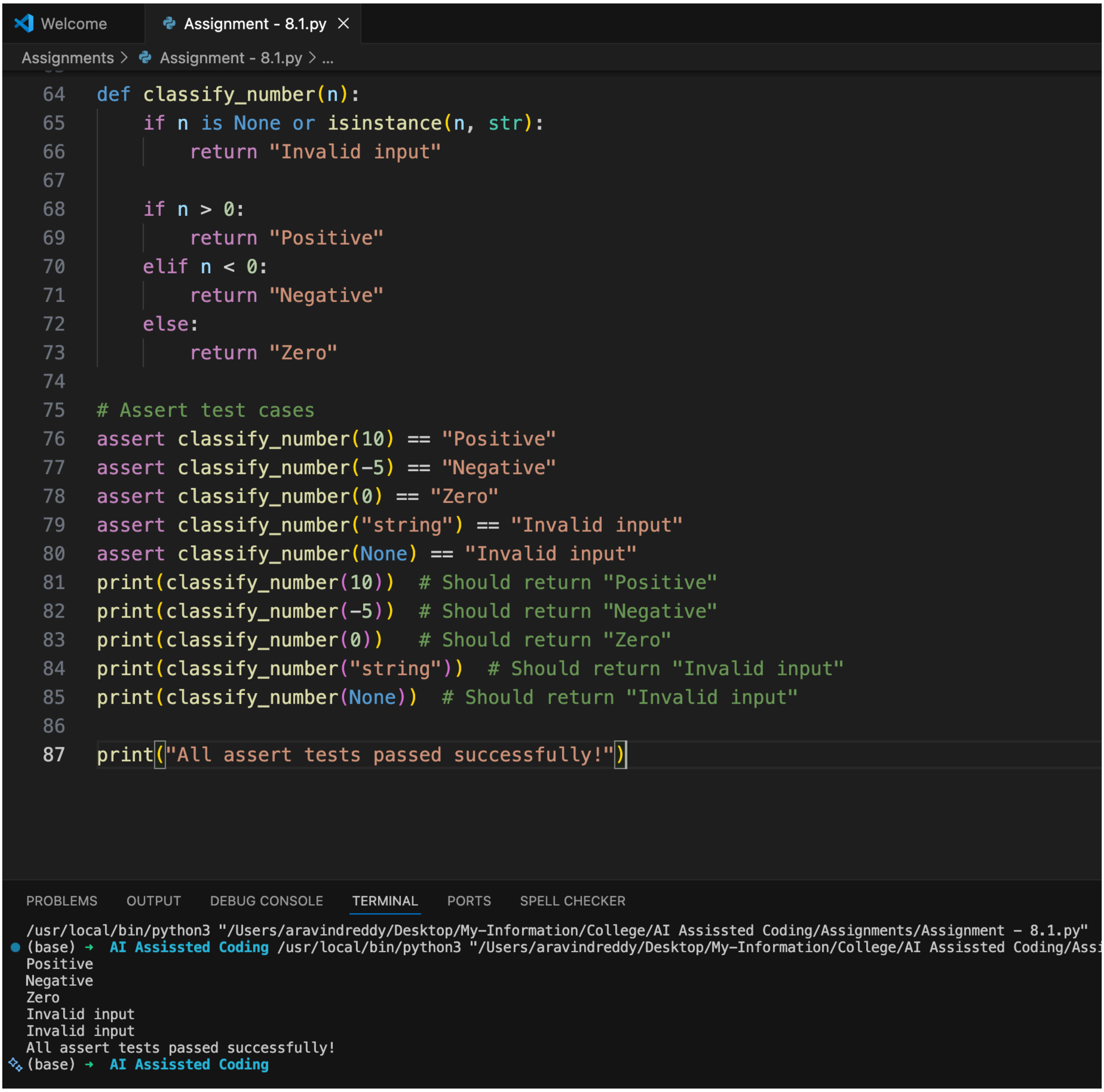
Task 1 : Password Strength Validator – Apply AI in Security Context



```python
def is_strong_password(password):
    if len(password) < 8:
        return False

    has_upper = False
    has_lower = False
    has_digit = False
    has_special = False
    special_characters = "!@#$%^&*()-+"

    for char in password:
        if char.isupper():
            has_upper = True
        elif char.islower():
            has_lower = True
        elif char.isdigit():
            has_digit = True
        elif char in special_characters:
            has_special = True
        elif char == " ":
            return False

    return has_upper and has_lower and has_digit and has_special

print(is_strong_password("Abcd@123"))  # Should return True
print(is_strong_password("abcd123"))  # Should return False
print(is_strong_password("Abcd123!"))  # Should return True
```

```
/usr/local/bin/python3 "/Users/aravindreddy/Desktop/My-Information/College/AI Assisted Coding/Assignments/Assignment - 8.1.py"
● (base) → AI Assisted Coding /usr/local/bin/python3 "/Users/aravindreddy/Desktop/My-Information/College/AI Assisted Coding/Assignments/Assi
True
False
True
(base) → AI Assisted Coding
```

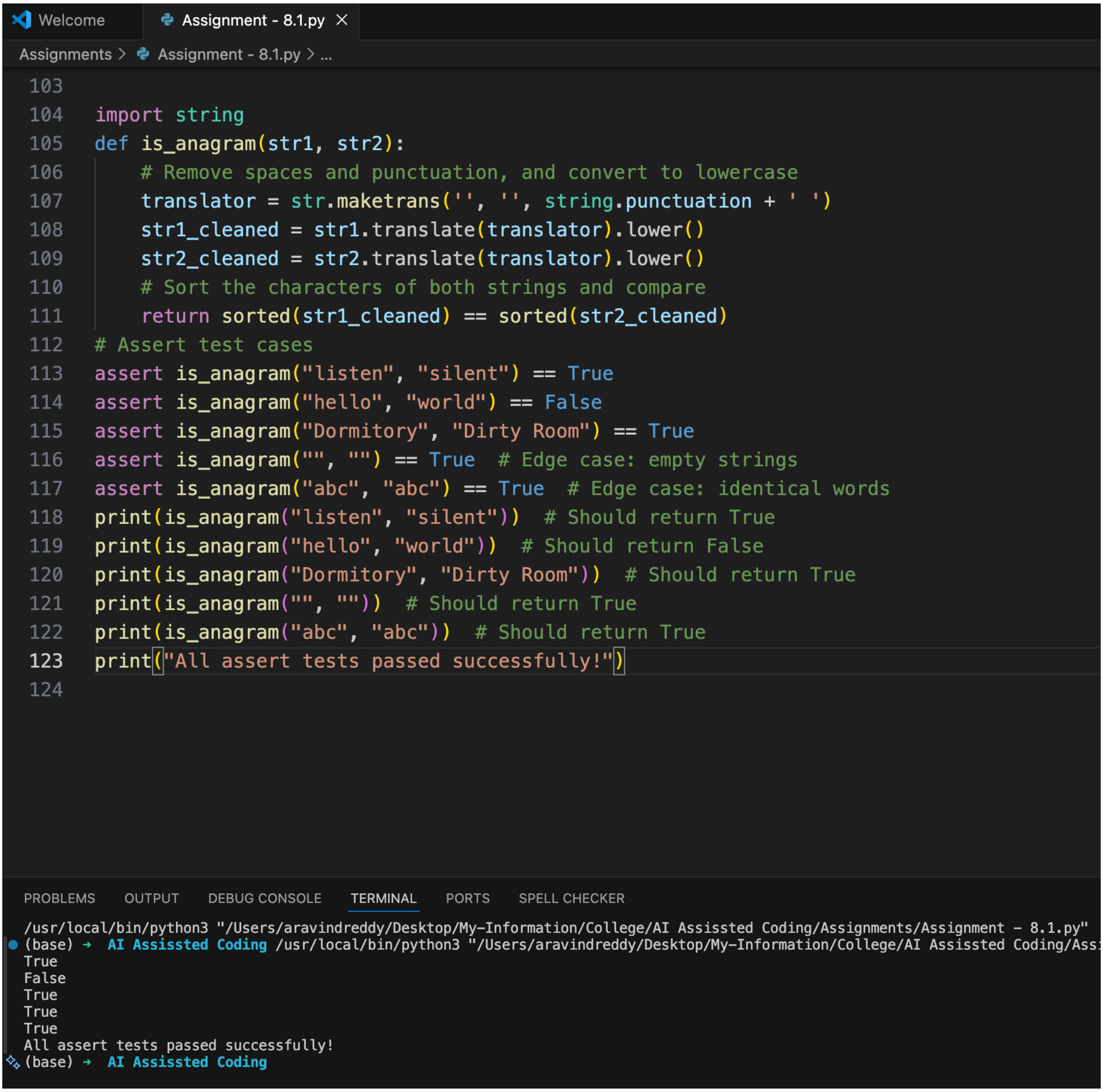
Task 2 : Number Classification with Loops – Apply AI for Edge Case Handling



```python
def classify_number(n):
    if n is None or isinstance(n, str):
        return "Invalid input"

    if n > 0:
        return "Positive"
    elif n < 0:
        return "Negative"
    else:
        return "Zero"

# Assert test cases
assert classify_number(10) == "Positive"
assert classify_number(-5) == "Negative"
assert classify_number(0) == "Zero"
assert classify_number("string") == "Invalid input"
assert classify_number(None) == "Invalid input"
print(classify_number(10))  # Should return "Positive"
print(classify_number(-5))  # Should return "Negative"
print(classify_number(0))   # Should return "Zero"
print(classify_number("string"))  # Should return "Invalid input"
print(classify_number(None))  # Should return "Invalid input"

print("All assert tests passed successfully!")
```

```
/usr/local/bin/python3 "/Users/aravindreddy/Desktop/My-Information/College/AI Assissted Coding/Assignments/Assignment - 8.1.py"
(base) → AI Assissted Coding /usr/local/bin/python3 "/Users/aravindreddy/Desktop/My-Information/College/AI Assissted Coding/Assi
Positive
Negative
Zero
Invalid input
Invalid input
All assert tests passed successfully!
(base) → AI Assissted Coding
```

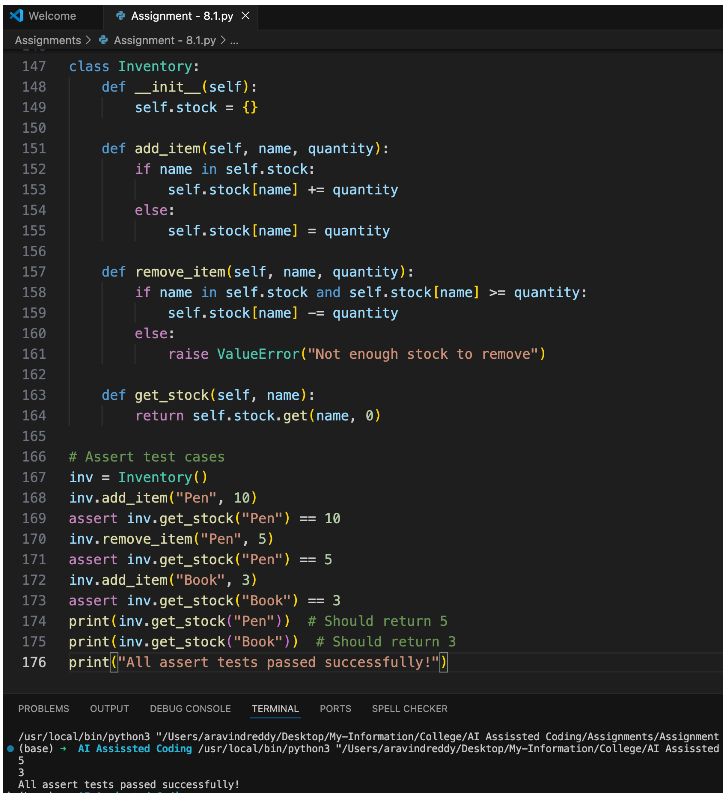
Task 3 : Anagram Checker – Apply AI for String Analysis



```python

import string
def is_anagram(str1, str2):
    # Remove spaces and punctuation, and convert to lowercase
    translator = str.maketrans('', '', string.punctuation + ' ')
    str1_cleaned = str1.translate(translator).lower()
    str2_cleaned = str2.translate(translator).lower()
    # Sort the characters of both strings and compare
    return sorted(str1_cleaned) == sorted(str2_cleaned)
# Assert test cases
assert is_anagram("listen", "silent") == True
assert is_anagram("hello", "world") == False
assert is_anagram("Dormitory", "Dirty Room") == True
assert is_anagram("", "") == True  # Edge case: empty strings
assert is_anagram("abc", "abc") == True  # Edge case: identical words
print(is_anagram("listen", "silent"))  # Should return True
print(is_anagram("hello", "world"))  # Should return False
print(is_anagram("Dormitory", "Dirty Room"))  # Should return True
print(is_anagram("", ""))  # Should return True
print(is_anagram("abc", "abc"))  # Should return True
print("All assert tests passed successfully!")

```

```
/usr/local/bin/python3 "/Users/aravindreddy/Desktop/My-Information/College/AI Assissted Coding/Assignments/Assignment - 8.1.py"
(base) → AI Assissted Coding /usr/local/bin/python3 "/Users/aravindreddy/Desktop/My-Information/College/AI Assissted Coding/Ass
True
False
True
True
True
All assert tests passed successfully!
(base) → AI Assissted Coding
```

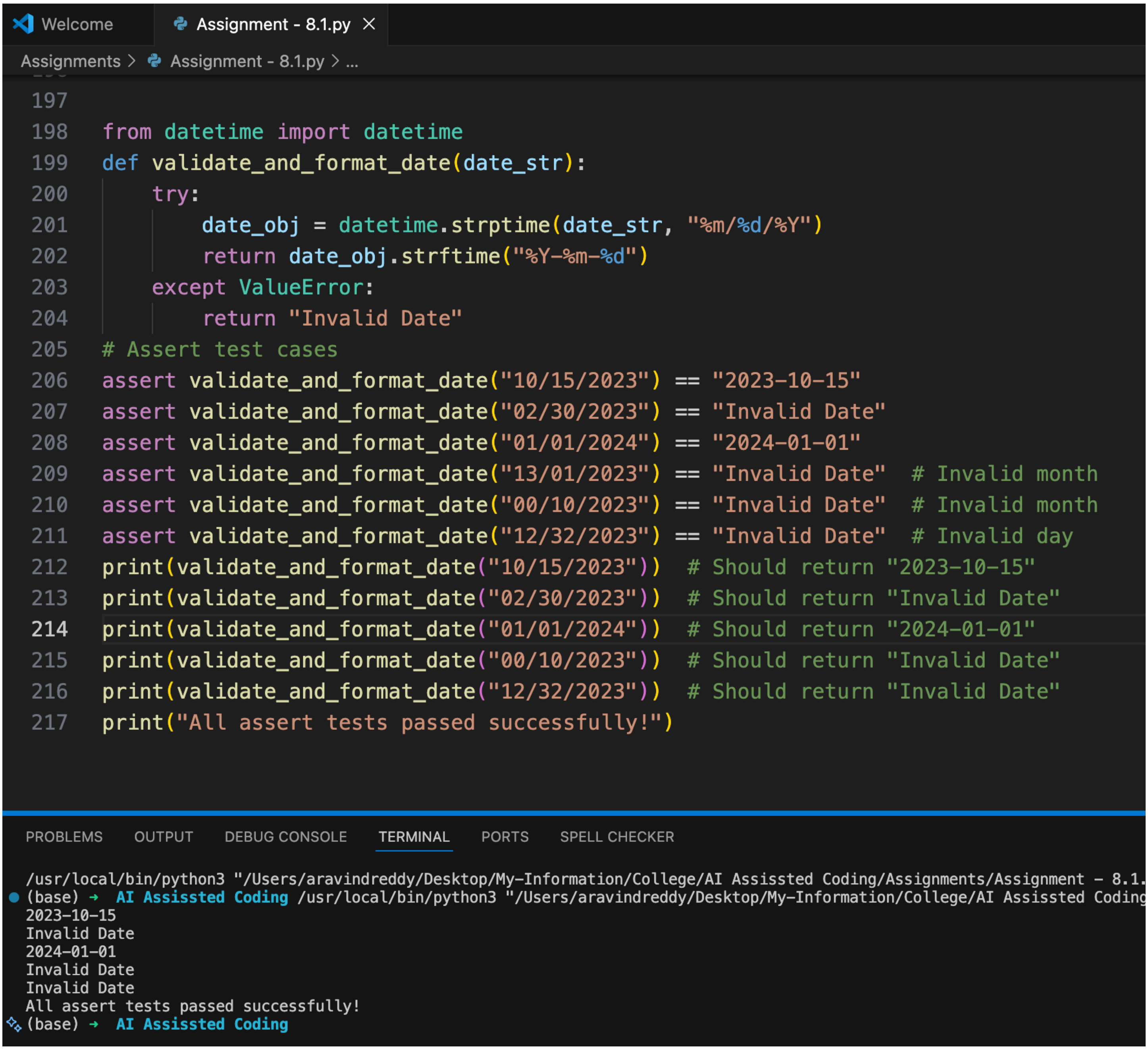
Task 4 : Inventory Class – Apply AI to Simulate Real- World Inventory System



```python
class Inventory:
    def __init__(self):
        self.stock = {}

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

    def remove_item(self, name, quantity):
        if name in self.stock and self.stock[name] >= quantity:
            self.stock[name] -= quantity
        else:
            raise ValueError("Not enough stock to remove")

    def get_stock(self, name):
        return self.stock.get(name, 0)

# Assert test cases
inv = Inventory()
inv.add_item("Pen", 10)
assert inv.get_stock("Pen") == 10
inv.remove_item("Pen", 5)
assert inv.get_stock("Pen") == 5
inv.add_item("Book", 3)
assert inv.get_stock("Book") == 3
print(inv.get_stock("Pen"))  # Should return 5
print(inv.get_stock("Book"))  # Should return 3
print("All assert tests passed successfully!")
```

```
 /usr/local/bin/python3 "/Users/aravindreddy/Desktop/My-Information/College/AI Assissted Coding/Assignments/Assignment
● (base) ➜  AI Assissted Coding /usr/local/bin/python3 "/Users/aravindreddy/Desktop/My-Information/College/AI Assissted
 5
 3
 All assert tests passed successfully!
```

Task 5 : Date Validation & Formatting – Apply AI for Data Validation



```python

from datetime import datetime
def validate_and_format_date(date_str):
    try:
        date_obj = datetime.strptime(date_str, "%m/%d/%Y")
        return date_obj.strftime("%Y-%m-%d")
    except ValueError:
        return "Invalid Date"
# Assert test cases
assert validate_and_format_date("10/15/2023") == "2023-10-15"
assert validate_and_format_date("02/30/2023") == "Invalid Date"
assert validate_and_format_date("01/01/2024") == "2024-01-01"
assert validate_and_format_date("13/01/2023") == "Invalid Date"  # Invalid month
assert validate_and_format_date("00/10/2023") == "Invalid Date"  # Invalid month
assert validate_and_format_date("12/32/2023") == "Invalid Date"  # Invalid day
print(validate_and_format_date("10/15/2023"))  # Should return "2023-10-15"
print(validate_and_format_date("02/30/2023"))  # Should return "Invalid Date"
print(validate_and_format_date("01/01/2024"))  # Should return "2024-01-01"
print(validate_and_format_date("00/10/2023"))  # Should return "Invalid Date"
print(validate_and_format_date("12/32/2023"))  # Should return "Invalid Date"
print("All assert tests passed successfully!")
```

```
/usr/local/bin/python3 "/Users/aravindreddy/Desktop/My-Information/College/AI Assissted Coding/Assignments/Assignment - 8.1.
(base) → AI Assissted Coding /usr/local/bin/python3 "/Users/aravindreddy/Desktop/My-Information/College/AI Assissted Coding
2023-10-15
Invalid Date
2024-01-01
Invalid Date
Invalid Date
All assert tests passed successfully!
(base) → AI Assissted Coding
```