2403A52L05 – B50

E. Sai Nithin Ramanujan

LAB ASS – 8.3

Task 1: Email Validation using TDD

Prompt and Code:

```python
#Task 1: Email Validation using TDD
"""Create a Python email validator using TDD. Write pytest test cases for valid and invalid emails, then implement `is_valid_email(email)` to pass them.
Rules: exactly one `@`, at least one `.` after `@`, no starting/ending special character, reject multiple `@`.
"""
import re

def is_valid_email(email: str) -> bool:
    if not isinstance(email, str) or not email.strip():
        return False
    if " " in email:
        return False

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

    # Must have exactly one @
    if email.count("@") != 1:
        return False

    local, domain = email.split("@")
    if not local or not domain:
        return False

    # Must contain . after @
    if "." not in domain:
        return False

    pattern = r"^[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,}$"
    return re.fullmatch(pattern, email) is not None


def run_tests():
    valid_emails = [
        "user@example.com",
        "john.doe123@mail-domain.org",
        "a_b+c@test.co",
        "user1@sub.domain.com",
    ]
```

```python
    invalid_emails = [
        "",
        "plainaddress",
        "@example.com",
        "user@",
        "user@@example.com",
        "userexample.com",
        ".user@example.com",
        "user@example.com!",
        "user@domain",
        "user name@example.com",
        "user@.com",
        "user@com.",
    ]

    passed = 0
    failed = 0

    print("Running test cases...\n")

    for email in valid_emails:
        result = is_valid_email(email)
        if result is True:
            print(f"[PASS] Valid email accepted:   {email}")
            passed += 1
        else:
            print(f"[FAIL] Valid email rejected:   {email}")
            failed += 1

    for email in invalid_emails:
        result = is_valid_email(email)
        if result is False:
            print(f"[PASS] Invalid email rejected: {email}")
            passed += 1
        else:
            print(f"[FAIL] Invalid email accepted: {email}")
            failed += 1
```

```
    total = passed + failed
    print("\n-----------------------------")
    print(f"Total:  {total}")
    print(f"Passed: {passed}")
    print(f"Failed: {failed}")
    print("-----------------------------")

    if failed == 0:
        print("All test cases passed successfully.")
    else:
        print("Some test cases failed.")


if __name__ == "__main__":
    run_tests()
```

OUTPUT:

```
PS D:\collage\AI-AC> python ass-8.3.py
Running test cases...

[PASS] Valid email accepted:   user@example.com
[PASS] Valid email accepted:   john.doe123@mail-domain.org
[PASS] Valid email accepted:   a_b+c@test.co
[PASS] Valid email accepted:   user1@sub.domain.com
[PASS] Invalid email rejected:
[PASS] Invalid email rejected: plainaddress
[PASS] Invalid email rejected: @example.com
[PASS] Invalid email rejected: user@
[PASS] Invalid email rejected: user@@example.com
[PASS] Invalid email rejected: userexample.com
[PASS] Invalid email rejected: .user@example.com
[PASS] Invalid email rejected: user@example.com!
[PASS] Invalid email rejected: user@domain
[PASS] Invalid email rejected: user name@example.com
[PASS] Invalid email rejected: user@.com
[PASS] Invalid email rejected: user@com.

-----------------------------
Total:  16
Passed: 16
Failed: 0
-----------------------------
All test cases passed successfully.
PS D:\collage\AI-AC>
```

Task 2: Grade Assignment using Loops

Prompt and Code:

```
#Task 2: Grade Assignment using Loops
"""Write a Python code that takes a list of student scores and assigns letter grades based on the following scale:
90-100: A; 80-89: B; 70-79: C; 60-69: D and Below 60: F
The code should loop through the scores and print each score along with its corresponding letter grade."""
def assign_grades(scores):
    for score in scores:
        if 90 <= score <= 100:
            grade = 'A'
        elif 80 <= score < 90:
            grade = 'B'
        elif 70 <= score < 80:
            grade = 'C'
        elif 60 <= score < 70:
            grade = 'D'
        else:
            grade = 'F'
        print(f"Score: {score} - Grade: {grade}")
# Example usage
student_scores = [95, 82, 76, 65, 54]
assign_grades(student_scores)
```

OUTPUT:

```
PS D:\collage\AI-AC> python ass-8.3.py
Score: 95 - Grade: A
Score: 82 - Grade: B
Score: 76 - Grade: C
Score: 65 - Grade: D
Score: 54 - Grade: F
```

Task 3: Sentence Palindrome Checker

Prompt and Code:

```python
#Task 3: Sentence Palindrome Checker
"""Write a Python function that checks if a given sentence is a palindrome, ignoring spaces, punctuation, and case sensitivity.
Examples:
Input: "A man, a plan, a canal, Panama!" → Output: True
Input: "Hello, World!" → Output: False"""
import re
def is_palindrome(sentence: str) -> bool:
    cleaned = re.sub(r'[^A-Za-z0-9]', '', sentence).lower()
    return cleaned == cleaned[::-1]
# Example usage
sentence1 = "A man, a plan, a canal, Panama!"
sentence2 = "Hello, World!"
print(f"'{sentence1}' is a palindrome: {is_palindrome(sentence1)}")
print(f"'{sentence2}' is a palindrome: {is_palindrome(sentence2)}")
```

OUTPUT:

```
PS D:\collage\AI-AC> python ass-8.3.py
'A man, a plan, a canal, Panama!' is a palindrome: True
'Hello, World!' is a palindrome: False
```

Task 4: Shopping Cart Class

Prompt and Code:

```python
#Task 4: ShoppingCart Class
"""Create a Python class `ShoppingCart` that allows users to add items with their prices, calculate the total cost,
and apply a discount code for a percentage off the total. The class should have methods to add items, calculate the total, and apply discounts.
"""
class ShoppingCart:
    def __init__(self):
        self.items = []
        self.total = 0.0

    def add_item(self, name: str, price: float):
        self.items.append((name, price))
        self.total += price

    def calculate_total(self) -> float:
        return self.total

    def apply_discount(self, discount_code: str) -> float:
        discounts = {
            "SAVE10": 0.10,
            "SAVE20": 0.20,
            "SAVE30": 0.30,
        }
        discount = discounts.get(discount_code.upper(), 0)
        discounted_total = self.total * (1 - discount)
        return round(discounted_total, 2)


# Example usage
cart = ShoppingCart()
cart.add_item("Book", 12.99)
cart.add_item("Headphones", 59.99)

print(f"Total before discount: ${cart.calculate_total():.2f}")

discounted_total = cart.apply_discount("SAVE20")
print(f"Total after discount: ${discounted_total:.2f}")
```

OUTPUT:

```
PS D:\collage\AI-AC> python ass-8.3.py
Total before discount: $72.98
Total after discount: $58.38
```

Task 5: Date Format Conversion

Prompt and Code:

```python
#Task 5: Date Format Conversion
"""Write a Python code that converts a date from "DD/MM/YYYY" format to "YYYY-MM-DD" format.
Example:
Input: "25/12/2024"
Output: "2024-12-25"
"""
def convert_date_format(date_str: str) -> str:
    try:
        day, month, year = date_str.split('/')
        return f"{year}-{month.zfill(2)}-{day.zfill(2)}"
    except ValueError:
        return "Invalid date format. Please use 'DD/MM/YYYY'."
# Example usage
date_input = "25/12/2024"
converted_date = convert_date_format(date_input)
print(f"Converted date: {converted_date}")
```

OUTPUT:

```
PS D:\collage\AI-AC> python ass-8.3.py
Converted date: 2024-12-25
```