Lab Assignment 8.5

Program	: B. Tech (CSE)
Course Title	: AI Assisted Coding
Course Code	: 23CS002PC304
Semester	: III
Academic Session	: 2025-2026
Name of Student	: Sruthi
Enrollment No.	: 2403A51L10
Batch No.	: 51
Date	: 30-01-2026

Task 1:

Prompt: Generate at least 3 Python assert test cases for a function
is_valid_username(username). Conditions: Username length must be between 5 and 15
characters, must contain only letters and digits, must not start with a digit, and must not
contain spaces. print the results if conditions satisfied true else false

Code:

```python
def is_valid_username(username):
    if not (5 <= len(username) <= 15):
        return False
    if not username.isalnum():
        return False
    if username[0].isdigit():
        return False
    if ' ' in username:
        return False
    return True

# Test cases
print(is_valid_username("User123"))  # Should print True
print(is_valid_username("1User"))    # Should print False
print(is_valid_username("User Name")) # Should print False
print(is_valid_username("Us"))        # Should print False
print(is_valid_username("ValidUser15")) # Should print True
print(is_valid_username("ThisUsernameIsWayTooLong")) # Should print False
```

Output:

```
True
False
False
False
True
False
```

Explanation:

The function is_valid_username checks if a given username meets specific criteria:

1. Length: The username must be between 5 and 15 characters long.

2. Characters: The username must contain only letters and digits (no special characters).

3. Starting Character: The username must not start with a digit.

4. Spaces: The username must not contain any spaces.

The function returns True if all conditions are satisfied, otherwise it returns False.

Task 2:

Prompt: Write a Python function named classify_value(x) that classifies the input based on the following rules:

1. If the input is not a number, return "Invalid Input".

2. If the input is 0, return "Zero".

3. If the input is an integer: Return "Even" if the number is divisible by 2, Return "Odd" if the number is not divisible by 2

Also generate at least 3 assert test cases to verify the function works correctly.

Code:

```python
def classify_value(x):
    if not isinstance(x, (int, float)):
        return "Invalid Input"
    if x == 0:
        return "Zero"
    if isinstance(x, int):
        if x % 2 == 0:
            return "Even"
        else:
            return "Odd"
    return "Invalid Input"
# Test cases
print(classify_value(0))            # Should print "Zero"
print(classify_value(4))            # Should print "Even"
print(classify_value(7))            # Should print "Odd"
print(classify_value("Hello"))      # Should print "Invalid Input"
print(classify_value(3.5))          # Should print "Invalid Input"
```

Output:

```
Zero
Even
Odd
Invalid Input
Invalid Input
```

Explanation:

The function classify_value takes an input x and classifies it based on the specified rules:

1. It first checks if x is a number (either int or float). If not, it returns "Invalid Input".

2. If x is 0, it returns "Zero".

3. If x is an integer, it checks if it is even or odd and returns "Even" or "Odd" accordingly.

4. If x is a float, it returns "Invalid Input" since the classification only applies to integers.

Task 3:

Prompt: Write a Python function called is_palindrome(text) that checks whether a string is a palindrome.

Requirements: Ignore uppercase/lowercase differences, ignore spaces, Ignore punctuation and special characters, Handle edge cases like empty strings and single characters
Also generate at least 3 assert test cases to verify the function.

Code:

```python
import string
def is_palindrome(text):
    # Normalize the text: convert to lowercase, remove spaces and punctuation
    normalized_text = ''.join(char.lower() for char in text if char.isalnum())

    # Check if the normalized text is the same forwards and backwards
    return normalized_text == normalized_text[::-1]
# Test cases
print(is_palindrome("A man, a plan, a canal: Panama"))  # Should print True
print(is_palindrome("racecar"))                         # Should print True
print(is_palindrome("Hello"))                           # Should print False
print(is_palindrome(""))                                # Should print True
print(is_palindrome("A"))                               # Should print True
```

Output:

```
True
True
False
True
True
```

Explanation:

The function is_palindrome checks if a given string is a palindrome by:

1. Normalizing the input string by converting it to lowercase and removing all non-alphanumeric characters (including spaces and punctuation).

2. Comparing the normalized string to its reverse.

If they are the same, it returns True, indicating that the string is a palindrome; otherwise, it returns False.

Task 4:

Prompt: Create a Python class called BankAccount.
The class should: Be initialized with a starting balance, have a method deposit(amount) that adds money to the balance, have a method withdraw(amount) that subtracts money only if sufficient funds exist, have a method get_balance () that returns the current balance. Also generate at least 3 assert-based test cases to verify the class works correctly. Keep the implementation simple and beginner-friendly.

Code:

```python
class BankAccount:
    def __init__(self, starting_balance):
        self.balance = starting_balance
    def deposit(self, amount):
        if amount > 0:
            self.balance += amount
    def withdraw(self, amount):
        if 0 < amount <= self.balance:
            self.balance -= amount
    def get_balance(self):
        return self.balance
# Test cases
account = BankAccount(100)
account.deposit(50)
print(account.get_balance())  # Should print 150
account.withdraw(30)
print(account.get_balance())  # Should print 120
account.withdraw(200)          # Should not withdraw, insufficient funds
print(account.get_balance())  # Should still print 120
```

Output:

```
150
120
120
```

Explanation:

The BankAccount class simulates a simple bank account with the following features:

1. Initialization: The account is initialized with a starting balance.

2. Deposit Method: The deposit method allows adding money to the account balance, but only if the amount is positive.

3. Withdraw Method: The withdraw method allows subtracting money from the balance, but only if there are sufficient funds.

4. Get Balance Method: The get_balance method returns the current balance of the account.

Task 5:

Prompt: Write a Python function called validate_email(email) to check whether an email address is valid. Requirements: Must contain exactly one "@" symbol Must contain at least one "." after the "@" Must not start or end with special characters like "@", ".", or spaces, should return False for clearly invalid formats, Keep the logic simple using basic string checks (beginner-friendly). Also generate at least 3 assert test cases to verify the function works correctly.

Code:

```python
def validate_email(email):
    # Check for exactly one "@" symbol
    if email.count('@') != 1:
        return False
    # Split the email into local and domain parts
    local, domain = email.split('@')
    # Check if local or domain parts are empty
    if not local or not domain:
        return False
    # Check if email starts or ends with special characters or spaces
    if email[0] in '@. ' or email[-1] in '@. ':
        return False
    # Check for at least one "." in the domain part
    if '.' not in domain:
        return False
    return True
# Test cases
print(validate_email("user@example.com"))      # Should print True
print(validate_email("userexample.com"))       # Should print False
print(validate_email("user@.com"))             # Should print False
print(validate_email("@example.com"))          # Should print False
print(validate_email("user@domain"))           # Should print False
```

Output:

```
True
False
True
False
False
```

Explanation:

The function validate_email checks if a given email address is valid based on specific criteria:

1. It ensures there is exactly one "@" symbol in the email.

2. It splits the email into local and domain parts and checks that neither is empty.

3. It checks that the email does not start or end with special characters like "@", ".", or spaces.

4. It verifies that there is at least one "." in the domain part of the email.