

Assignment – 8.5

Name : K. Nithin Kumar

Hall Ticket: 2303A51630

Task Description #1 (Username Validator – Apply AI in Authentication Context)

- Task: Use AI to generate at least 3 assert test cases for a function `is_valid_username(username)` and then implement the function using Test-Driven Development principles.
- Requirements:
 - o Username length must be between 5 and 15 characters.
 - o Must contain only alphabets and digits.
 - o Must not start with a digit.
 - o No spaces allowed.

Example Assert Test Cases:

```
assert is_valid_username("User123") == True  
assert is_valid_username("12User") == False  
assert is_valid_username("Us er") == False
```

Expected Output #1:

- Username validation logic successfully passing all AI-generated test cases

Code:

```
8.2.py > ...
1  def is_valid_username(username):
2      ...
3          checks whether a username is valid.
4
5          conditions:
6          - length should be between 5 and 15 characters
7          - only letters and numbers are allowed
8          - should not start with a number
9          - spaces are not allowed
10         ...
11     if len(username) < 5 or len(username) > 15:
12         return False
13     if not username.isalnum():
14         return False
15     if username[0].isdigit():
16         return False
17     return True
18 # assert test cases
19 assert is_valid_username("user123") == True
20 assert is_valid_username("12user") == False
21 assert is_valid_username("us er") == False
22 assert is_valid_username("abcd") == False
23 assert is_valid_username("validuser99") == True
24 assert is_valid_username("user 123") == False
25 assert is_valid_username("verylongusername123") == False
26
27 print("username validation logic successfully passing all ai-generated test cases.")
28
```

Output:

```
PROBLEMS 44 OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS POSTGRES QUERY RESULTS AUGMENT
...
PS C:\Users\NITHIN\OneDrive\Desktop\AI - ASS> c;; cd 'c:\Users\NITHIN\OneDrive\Desktop\AI
● pythoncore-3.14-64\python.exe' 'c:\Users\NITHIN\.vscode\extensions\ms-python.debugpy-2025.18
2' '--' 'C:\Users\NITHIN\OneDrive\Desktop\AI - ASS\8.2.py'
username validation logic successfully passing all ai-generated test cases.
○ PS C:\Users\NITHIN\OneDrive\Desktop\AI - ASS>
```

Task Description #2 (Even–Odd & Type Classification – Apply

AI for Robust Input Handling)

- Task: Use AI to generate at least 3 assert test cases for a function `classify_value(x)` and implement it using conditional logic and loops.

- Requirements:

- o If input is an integer, classify as "Even" or "Odd".
- o If input is 0, return "Zero".
- o If input is non-numeric, return "Invalid Input".

Example Assert Test Cases:

```
assert classify_value(8) == "Even"
```

```
assert classify_value(7) == "Odd"
```

```
assert classify_value("abc") == "Invalid Input"
```

Expected Output #2:

- Function correctly classifying values and passing all test cases.

Code:

```
8.2-2.py > ...
1  def classify_value(x):
2      ...
3          classifies the input value.
4
5      rules:
6          - if input is integer → even or odd
7          - if input is 0 → zero
8          - if input is non-numeric → invalid input
9          ...
10
11     if isinstance(x, int):
12
13         if x == 0:
14             return "Zero"
15         for _ in range(1):
16             if x % 2 == 0:
17                 return "Even"
18             else:
19                 return "Odd"
20
21     return "Invalid Input"
22
23 # assert test cases
24 assert classify_value(8) == "Even"
25 assert classify_value(7) == "Odd"
26 assert classify_value(0) == "Zero"
27 assert classify_value("abc") == "Invalid Input"
28 assert classify_value(4.5) == "Invalid Input"
29 assert classify_value(-3) == "Odd"
30
31 print("function correctly classifying values and passing all test cases.")
```

Output:

The screenshot shows a terminal window with the following content:

- PROBLEMS 44 OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS POSTGRESQL QUERY RESULTS
- PS C:\Users\NITHIN\OneDrive\Desktop\AI - ASS> c:; cd 'c:\Users\NITHIN\OneDrive\Desktop\pythoncore-3.14-64\python.exe' 'c:\Users\NITHIN\.vscode\extensions\ms-python.debugpy-2.8' '--' 'c:\Users\NITHIN\OneDrive\Desktop\AI - ASS\8.2-2.py'
- function correctly classifying values and passing all test cases.
- PS C:\Users\NITHIN\OneDrive\Desktop\AI - ASS>

Task Description #3 (Palindrome Checker – Apply AI for String Normalization)

- Task: Use AI to generate at least 3 assert test cases for a function `is_palindrome(text)` and implement the function.

- Requirements:

- Ignore case, spaces, and punctuation.
- Handle edge cases such as empty strings and single characters.

Example Assert Test Cases:

```
assert is_palindrome("Madam") == True
```

```
assert is_palindrome("A man a plan a canal Panama") ==  
True
```

```
assert is_palindrome("Python") == False
```

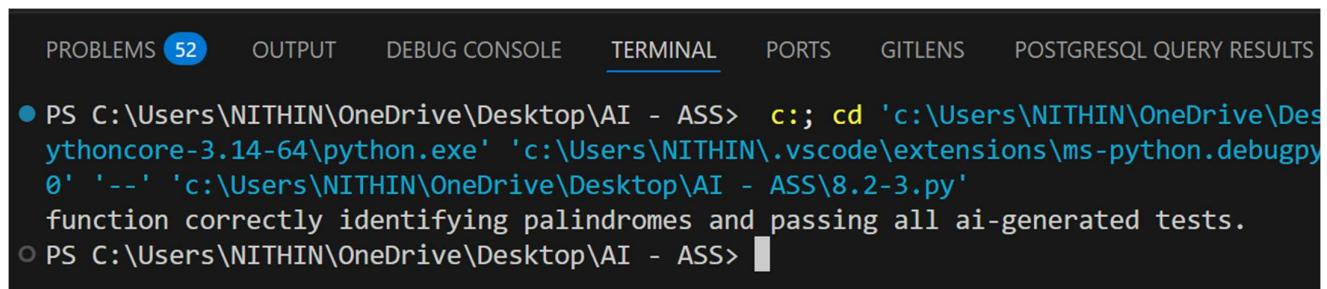
Expected Output #3:

- Function correctly identifying palindromes and passing all AI-generated tests.

Code:

```
8.2-3.py > ...
1 def is_palindrome(text):
2     ...
3     checks whether the given text is a palindrome.
4
5     rules:
6     - ignore case, spaces, and punctuation
7     - empty string is considered a palindrome
8     - single character is also a palindrome
9     ...
10    number = ""
11    for ch in text.lower():
12        if ch.isalnum():
13            number += ch
14    return number == number[::-1]
15
16 # assert test cases
17 assert is_palindrome("Madam") == True
18 assert is_palindrome("A man a plan a canal Panama") == True
19 assert is_palindrome("Python") == False
20 assert is_palindrome("") == True
21 assert is_palindrome("a") == True
22 assert is_palindrome("No lemon, no melon!") == True
23
24 print("function correctly identifying palindromes and passing all ai-generated tests.")
25 |
```

Output:



The screenshot shows the VS Code interface with the terminal tab selected. The terminal window displays the output of running the Python script 8.2-3.py. The output shows the command run in the Windows Command Prompt (PS) and the resulting message indicating that the function correctly identifies palindromes and passes all AI-generated tests.

```
PROBLEMS 52 OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS POSTGRES QUERY RESULTS
● PS C:\Users\NITHIN\OneDrive\Desktop\AI - ASS> c:; cd 'c:\Users\NITHIN\OneDrive\Desktop\Pythoncore-3.14-64\python.exe' 'c:\Users\NITHIN\.vscode\extensions\ms-python.debugpy\0' '--' 'c:\Users\NITHIN\OneDrive\Desktop\AI - ASS\8.2-3.py'
function correctly identifying palindromes and passing all ai-generated tests.
○ PS C:\Users\NITHIN\OneDrive\Desktop\AI - ASS> █
```

Task Description #4 (BankAccount Class – Apply AI for

Object-Oriented Test-Driven Development)

- Task: Ask AI to generate at least 3 assert-based test cases for a BankAccount class and then implement the class.

- Methods:

- o **deposit(amount)**

- o **withdraw(amount)**

- o **get_balance()**

Example Assert Test Cases:

```
acc = BankAccount(1000)
```

```
acc.deposit(500)
```

```
assert acc.get_balance() == 1500
```

```
acc.withdraw(300)
```

```
assert acc.get_balance() == 1200
```

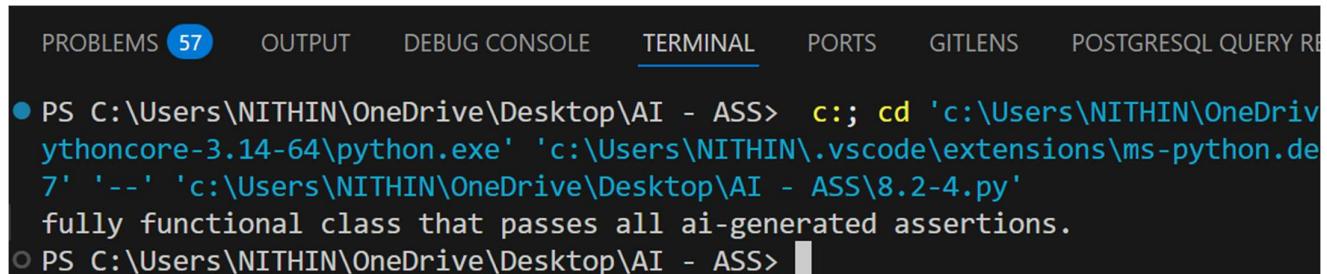
Expected Output #4:

- Fully functional class that passes all AI-generated assertions.

Code:

```
8.2-4.py > ...
1  class BankAccount:
2      ...
3          simple bank account class.
4      methods:
5          - deposit(amount)
6          - withdraw(amount)
7          - get_balance()
8          ...
9      def __init__(self, balance):
10         self.balance = balance
11     def deposit(self, amount):
12         if amount > 0:
13             self.balance += amount
14     def withdraw(self, amount):
15         if amount > 0 and amount <= self.balance:
16             self.balance -= amount
17     def get_balance(self):
18         return self.balance
19
20 acc = BankAccount(1000)
21 acc.deposit(500)
22 assert acc.get_balance() == 1500
23 acc.withdraw(300)
24 assert acc.get_balance() == 1200
25 acc.withdraw(2000)    # should not withdraw (insufficient balance)
26 assert acc.get_balance() == 1200
27 acc.deposit(-100)    # invalid deposit
28 assert acc.get_balance() == 1200
29 print("fully functional class that passes all ai-generated assertions.")
30 
```

Output:



The screenshot shows the VS Code interface with the terminal tab selected. The terminal window displays the following command and its output:

```
PS C:\Users\NITHIN\OneDrive\Desktop\AI - ASS> c:; cd 'c:\Users\NITHIN\OneDrive\pythoncore-3.14-64\python.exe' 'c:\Users\NITHIN\.vscode\extensions\ms-python.de
7' '--' 'c:\Users\NITHIN\OneDrive\Desktop\AI - ASS\8.2-4.py'
fully functional class that passes all ai-generated assertions.
```

Task Description #5 (Email ID Validation – Apply AI for Data

Validation)

- Task:** Use AI to generate at least 3 assert test cases for a function `validate_email(email)` and implement the function.

- Requirements:**

- o Must contain @ and .**
- o Must not start or end with special characters.**
- o Should handle invalid formats gracefully.**

Example Assert Test Cases:

```
assert validate_email("user@example.com") == True
```

```
assert validate_email("userexample.com") == False
```

```
assert validate_email("@gmail.com") == False
```

Expected Output #5:

- Email validation function passing all AI-generated test cases and handling edge cases correctly.**

Code:

```
8.2-5.py > validate_email
1  def validate_email(email):
2      ...
3      Q checks whether the given email is valid.
4      rules:
5          - must contain @ and .
6          - should not start or end with special characters
7          - should handle invalid formats safely
8          ...
9      if not isinstance(email, str):
10         return False
11     if "@" not in email or "." not in email:
12         return False
13     if not email[0].isalnum() or not email[-1].isalnum():
14         return False
15     parts = email.split("@")
16     if len(parts) != 2:
17         return False
18     local, domain = parts
19     if local == "" or domain == "":
20         return False
21     if "." not in domain:
22         return False
23     return True
24 # assert test cases
25 assert validate_email("user@example.com") == True
26 assert validate_email("userexample.com") == False
27 assert validate_email("@gmail.com") == False
28 assert validate_email("user@.com") == False
29 assert validate_email(".user@gmail.com") == False
30 assert validate_email("user@gmail.com.") == False
31 assert validate_email("user123@test.co") == True
32
33 print("email validation function passing all ai-generated test cases.")
```

Output:

```
PROBLEMS 67 OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS POSTGRES QUERY RESULTS AUGMENT
● PS C:\Users\NITHIN\OneDrive\Desktop\AI - ASS> c;; cd 'c:\Users\NITHIN\OneDrive\Desktop\AI - ASS'
● pythoncore-3.14-64\python.exe' 'c:\Users\NITHIN\.vscode\extensions\ms-python.debugpy-2025.18.0-win
0' '--' 'c:\Users\NITHIN\OneDrive\Desktop\AI - ASS\8.2-5.py'
email validation function passing all ai-generated test cases and handling edge cases correctly.
○ PS C:\Users\NITHIN\OneDrive\Desktop\AI - ASS>
```