

AI Assisted Coding

Assignment-8.5

2303A51670

Samshray

Batch-23

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.

```
Assignment-8.5.py X
Friday.py > Assignment-8.5.py > ...
207
208 def is_valid_username(username):
209     if len(username) < 5 or len(username) > 15:
210         return False
211     if not username[0].isalnum():
212         return False
213     for char in username:
214         if not char.isalnum() and char != '_':
215             return False
216     return True
217 # Test cases for the is_valid_username function
218 assert is_valid_username("user123") == True, "Test case 1 failed"
219 assert is_valid_username("1user") == True, "Test case 2 failed"
220 assert is_valid_username("user_name") == True, "Test case 3 failed"
221 assert is_valid_username("us") == False, "Test case 4 failed"
222 print("All test cases passed!")

PROBLEMS  DEBUG CONSOLE  OUTPUT  TERMINAL  PORTS
● PS C:\Users\Ganne\OneDrive\Desktop\Ai_Assisted_Coding> & "C:/Program Files/Python312/python.exe"
  nt-8.5.py"
  All test cases passed!
○ PS C:\Users\Ganne\OneDrive\Desktop\Ai_Assisted_Coding>
```

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

```

223
224 def classify_value(x):
225     if x < 0:
226         return "Negative"
227     elif x == 0:
228         return "Zero"
229     elif x%2==0:
230         return "Even"
231     else:
232         return "Odd"
233
234 # Test cases for the classify_value function:
235 assert classify_value(8) == "Even"
236 assert classify_value(-3) == "Negative"
237 assert classify_value(0) == "Zero"
238 assert classify_value("abc") == "Invalid input"
239 print("All test cases passed!")

```

PROBLEMS DEBUG CONSOLE OUTPUT TERMINAL PORTS

```

PS C:\Users\Ganne\OneDrive\Desktop\AI_Assisted_Coding> & "C:/Program Files/Python312/python.exe" "c:/Users/Ganne/OneDrive/Desktop/
assert classify_value("abc") == "Invalid input"
File "c:/Users/Ganne/OneDrive/Desktop/AI_Assisted_Coding\Friday.py\Assignment-8.5.py", line 225, in classify_value
    if x < 0:
TypeError: '<' not supported between instances of 'str' and 'int'
PS C:\Users\Ganne\OneDrive\Desktop\AI_Assisted_Coding>

```

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:

- o Ignore case, spaces, and punctuation.

- o 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

```
240
241 def is_palindrome(text):
242     cleaned_text = ''.join(char.lower() for char in text if char.isalnum())
243     return cleaned_text == cleaned_text[::-1]
244 # Test cases for the is_palindrome function
245 assert is_palindrome("A man, a plan, a canal panama") == True, "Test case 1 failed"
246 assert is_palindrome("Hello") == False, "Test case 2 failed"
247 assert is_palindrome("Hi") == False, "Test case 3 failed"
248 print("All test cases passed!")
```

PROBLEMS DEBUG CONSOLE OUTPUT **TERMINAL** PORTS

```
PS C:\Users\Garne\OneDrive\Desktop\AI_Assisted_Coding> & "C:/Program Files/Python312/python.exe" "c:/Users/Garne/OneDrive/Desktop/AI_Assisted_Coding/is_palindrome_test.py"
All test cases passed!
PS C:\Users\Garne\OneDrive\Desktop\AI_Assisted_Coding>
```

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.

```
Assignment-8.5.py X
```

```
Friday.py > Assignment-8.5.py > BankAccount > deposit
```

```
251 class BankAccount:
252     def __init__(self, account_number, balance=0):
253         self.account_number = account_number
254         self.balance = balance
255
256     def deposit(self, amount):
257         if amount > 0:
258             self.balance += amount
259             return True
260         return False
261
262     def withdraw(self, amount):
263         if 0 < amount <= self.balance:
264             self.balance -= amount
265             return True
266         return False
267     def get_balance(self):
268         return self.balance
269 # Test cases for the BankAccount class
270 acc = BankAccount(1000)
271 acc.deposit(500)
272 assert acc.get_balance() == 1500, "Deposit test failed"
273 acc.withdraw(1000)
274 assert acc.get_balance() == 500, "Withdraw test failed"
275 print("All test cases passed!")
```

```
PROBLEMS DEBUG CONSOLE OUTPUT TERMINAL PORTS
```

```
PS C:\Users\Ganne\OneDrive\Desktop\Ai_Assisted_Coding> & "C:/Program Files/Python312/python.exe" "c:/Users/Ganne/OneDrive/Desktop/Ai_Assisted_Coding/Friday.py"
IndentationError: unindent does not match any outer indentation level
PS C:\Users\Ganne\OneDrive\Desktop\Ai_Assisted_Coding> & "C:/Program Files/Python312/python.exe" "c:/Users/Ganne/OneDrive/Desktop/Ai_Assisted_Coding/Friday.py"
Traceback (most recent call last):
  File "c:/Users/Ganne/OneDrive/Desktop/Ai_Assisted_Coding\Friday.py\Assignment-8.5.py", line 272, in <module>
    assert acc.get_balance() == 1500, "Deposit test failed"
           ^^^^^^^^^^^^^^^^^
AssertionError: Deposit test failed
PS C:\Users\Ganne\OneDrive\Desktop\Ai_Assisted_Coding>
```

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.

```
276
277 def validate_email(email):
278     if '@' not in email or '.' not in email:
279         return False
280     at_index = email.index('@')
281     dot_index = email.rindex('.')
282     if at_index < 1 or dot_index < at_index + 2 or dot_index >= len(email) - 1:
283         return False
284     return True
285
286 # Test cases for the validate_email function
287 assert validate_email("user@example.com") == True, "Test case 1 failed"
288 assert validate_email("userexample.com") == False, "Test case 2 failed"
289 assert validate_email("user@.com") == False, "Test case 3 failed"
290 print("All test cases passed!")
```

PROBLEMS DEBUG CONSOLE OUTPUT TERMINAL PORTS

```
● PS C:\Users\Ganne\OneDrive\Desktop\Ai_Assisted_coding> & "C:/Program Files/Python312/python.exe" "c:/Users,
.
All test cases passed!
○ PS C:\Users\Ganne\OneDrive\Desktop\Ai_Assisted_coding>
```