

## School of Computer Science and Artificial Intelligence

### Lab Assignment # 8.5

<b>Program</b>	<b>:</b> B. Tech (CSE)
<b>Specialization</b>	<b>:</b> -
<b>Course Title</b>	<b>:</b> AI Assisted Coding
<b>Course Code</b>	<b>:</b> 23CS002PC304
<b>Semester</b>	<b>II</b>
<b>Academic Session</b>	<b>:</b> 2025-2026
<b>Name of Student</b>	<b>:</b> R.Akash Reddy
<b>Enrollment No.</b>	<b>:</b> 2403A51L30
<b>Batch No.</b>	<b>51</b>
<b>Date</b>	<b>:</b> 30/01/26

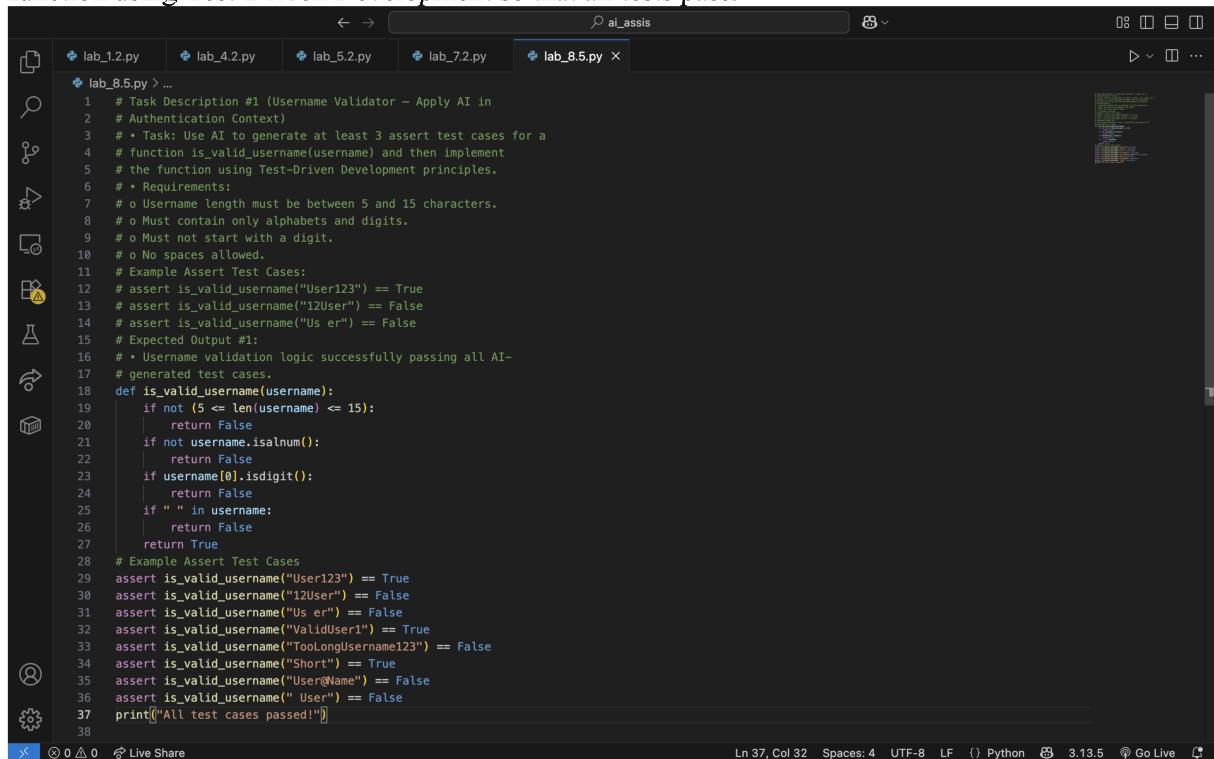
### Submission Starts here

#### Screenshots:

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

#### Prompt:

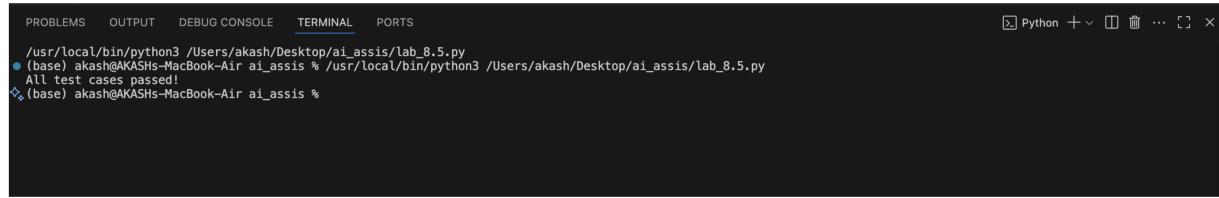
Generate at least 3 assert-based test cases for a Python function `is_valid_username(username)` with the following rules: - Username length must be between 5 and 15 characters. - Only alphabets and digits allowed. - Username must not start with a digit. - No spaces allowed. Then implement the function using Test-Driven Development so that all tests pass.



```

lab_8.5.py > ...
1  # Task Description #1 (Username Validator – Apply AI in
2  # Authentication Context)
3  # * Task: Use AI to generate at least 3 assert test cases for a
4  # function is_valid_username(username) and then implement
5  # the function using Test-Driven Development principles.
6  # * Requirements:
7  # o Username length must be between 5 and 15 characters.
8  # o Must contain only alphabets and digits.
9  # o Must not start with a digit.
10 # o No spaces allowed.
11 # Example Assert Test Cases:
12 # assert is_valid_username("User123") == True
13 # assert is_valid_username("12User") == False
14 # assert is_valid_username("Us er") == False
15 # Expected Output #1:
16 # * Username validation logic successfully passing all AI-
17 # generated test cases.
18 def is_valid_username(username):
19     if not (5 <= len(username) <= 15):
20         return False
21     if not username.isalnum():
22         return False
23     if username[0].isdigit():
24         return False
25     if " " in username:
26         return False
27     return True
28 # Example Assert Test Cases
29 assert is_valid_username("User123") == True
30 assert is_valid_username("12User") == False
31 assert is_valid_username("Us er") == False
32 assert is_valid_username("ValidUser1") == True
33 assert is_valid_username("TooLongUsername123") == False
34 assert is_valid_username("Short") == True
35 assert is_valid_username("User@Name") == False
36 assert is_valid_username(" User") == False
37 print("All test cases passed!")
38

```



```
/usr/local/bin/python3 /Users/akash/Desktop/ai_assis/lab_8.5.py
● (base) akash@AKASH-MacBook-Air ai_assis % /usr/local/bin/python3 /Users/akash/Desktop/ai_assis/lab_8.5.py
All test cases passed!
● (base) akash@AKASH-MacBook-Air ai_assis %
```

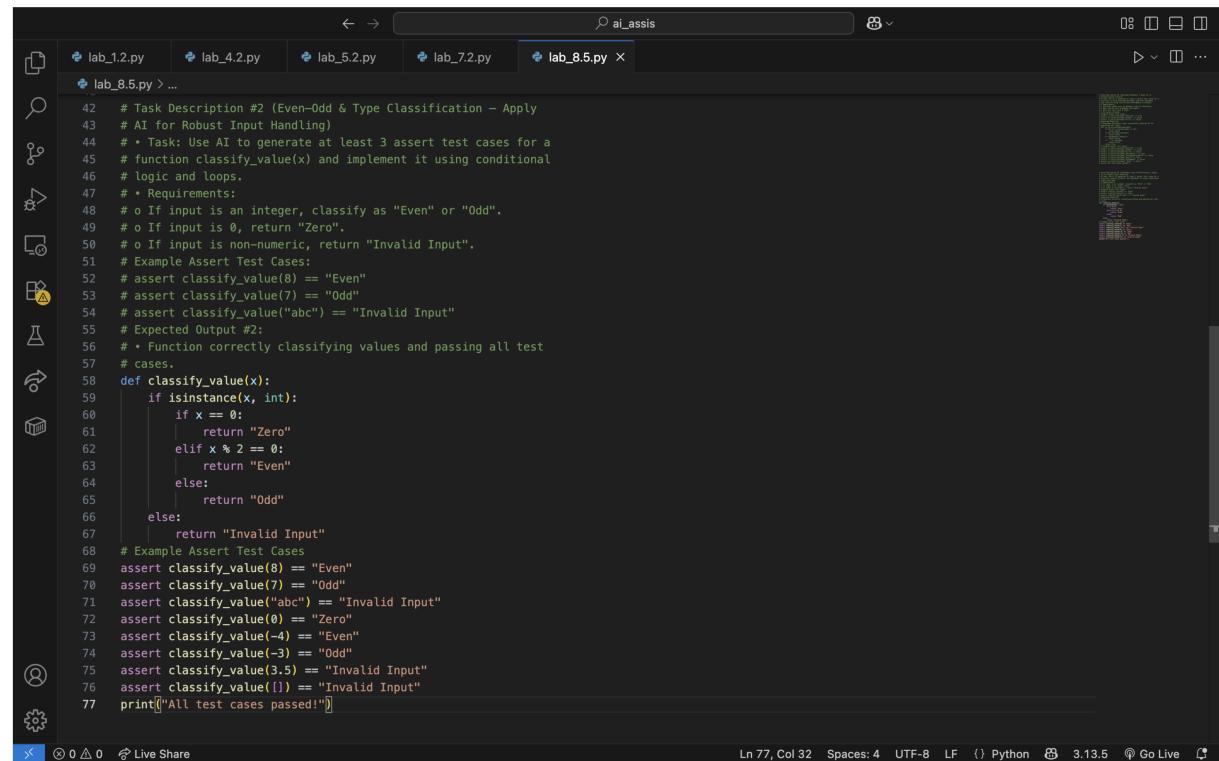
## Task Description #2 (Even–Odd & Type Classification – Apply AI for Robust Input Handling)

### Prompt:

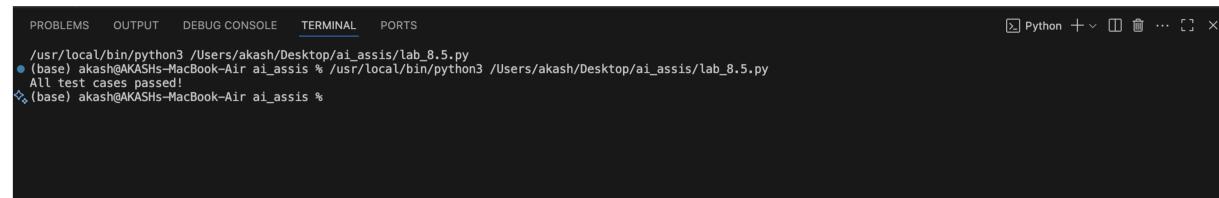
Generate at least 3 assert test cases for a Python function `classify_value(x)` and implement the function using conditional logic and loops with the following rules:

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

Ensure all tests pass.



```
42  # Task Description #2 (Even-Odd & Type Classification - Apply
43  # AI for Robust Input Handling)
44  # * Task: Use AI to generate at least 3 assert test cases for a
45  # function classify_value(x) and implement it using conditional
46  # logic and loops.
47  # * Requirements:
48  # o If input is an integer, classify as "Even" or "Odd".
49  # o If input is 0, return "Zero".
50  # o If input is non-numeric, return "Invalid Input".
51  # Example Assert Test Cases:
52  # assert classify_value(8) == "Even"
53  # assert classify_value(7) == "Odd"
54  # assert classify_value("abc") == "Invalid Input"
55  # Expected Output #2:
56  # * Function correctly classifying values and passing all test
57  # cases.
58 def classify_value(x):
59     if isinstance(x, int):
60         if x == 0:
61             return "Zero"
62         elif x % 2 == 0:
63             return "Even"
64         else:
65             return "Odd"
66     else:
67         return "Invalid Input"
68 # Example Assert Test Cases
69 assert classify_value(8) == "Even"
70 assert classify_value(7) == "Odd"
71 assert classify_value("abc") == "Invalid Input"
72 assert classify_value(0) == "Zero"
73 assert classify_value(-4) == "Even"
74 assert classify_value(-3) == "Odd"
75 assert classify_value(3.5) == "Invalid Input"
76 assert classify_value([]) == "Invalid Input"
77 print("All test cases passed!")
```



```
/usr/local/bin/python3 /Users/akash/Desktop/ai_assis/lab_8.5.py
● (base) akash@AKASH-MacBook-Air ai_assis % /usr/local/bin/python3 /Users/akash/Desktop/ai_assis/lab_8.5.py
All test cases passed!
● (base) akash@AKASH-MacBook-Air ai_assis %
```

Code:

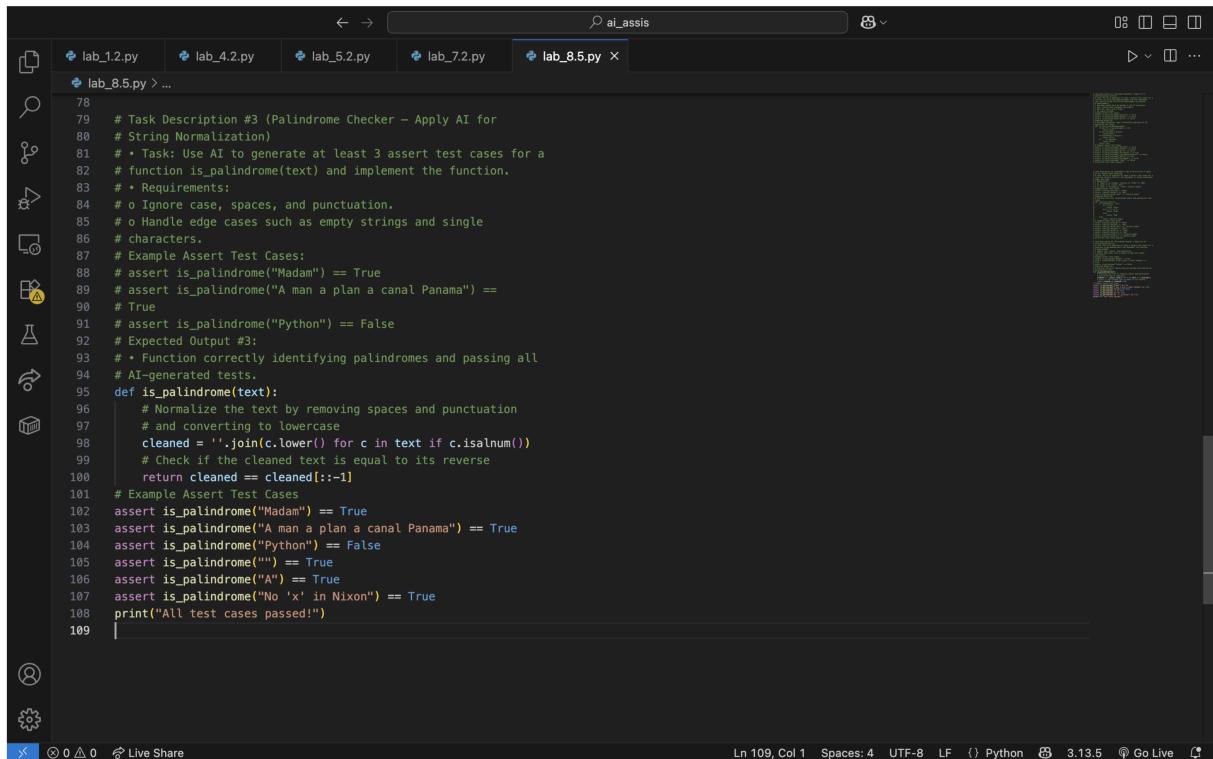
### Task Description #3 (Palindrome Checker – Apply AI for String Normalization)

#### Prompt:

Generate at least 3 assert test cases for a function `is_palindrome(text)` and implement the function.

Requirements:

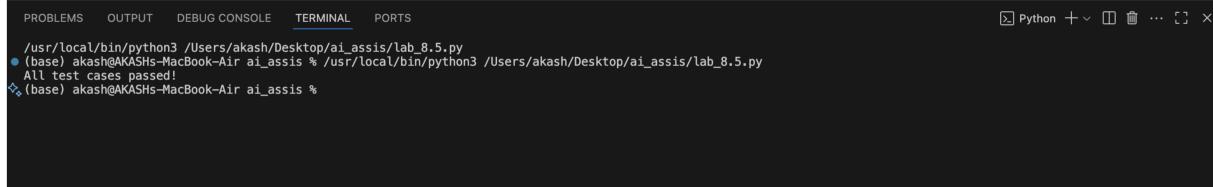
- Ignore case, spaces, and punctuation.
- Handle empty string and single characters.
- Ensure all tests pass.



```

78  # Task Description #3 (Palindrome Checker – Apply AI for
79  # String Normalization)
80  # * Task: Use AI to generate at least 3 assert test cases for a
81  # function is_palindrome(text) and implement the function.
82  # * Requirements:
83  # o Ignore case, spaces, and punctuation.
84  # o Handle edge cases such as empty strings and single
85  # characters.
86  # Example Assert Test Cases:
87  # assert is_palindrome("Madam") == True
88  # assert is_palindrome("A man a plan a canal Panama") == True
89  # assert is_palindrome("Python") == False
90  # Expected Output #3:
91  # * Function correctly identifying palindromes and passing all
92  # AI-generated tests.
93  def is_palindrome(text):
94      # Normalize the text by removing spaces and punctuation
95      # and converting to lowercase
96      cleaned = ''.join(c.lower() for c in text if c.isalnum())
97      # Check if the cleaned text is equal to its reverse
98      return cleaned == cleaned[::-1]
99
100 # Example Assert Test Cases
101 assert is_palindrome("Madam") == True
102 assert is_palindrome("A man a plan a canal Panama") == True
103 assert is_palindrome("Python") == False
104 assert is_palindrome("") == True
105 assert is_palindrome("A") == True
106 assert is_palindrome("No 'x' in Nixon") == True
107
108 print("All test cases passed!")
109

```



```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
/usr/local/bin/python3 /Users/akash/Desktop/ai_assis/lab_8.5.py
(base) akash@AKASH-MacBook-Air ~ % /usr/local/bin/python3 /Users/akash/Desktop/ai_assis/lab_8.5.py
All test cases passed!
(base) akash@AKASH-MacBook-Air ~ %

```

Code:

### Task Description #4 (BankAccount Class – Apply AI for Object-Oriented Test-Driven Development)

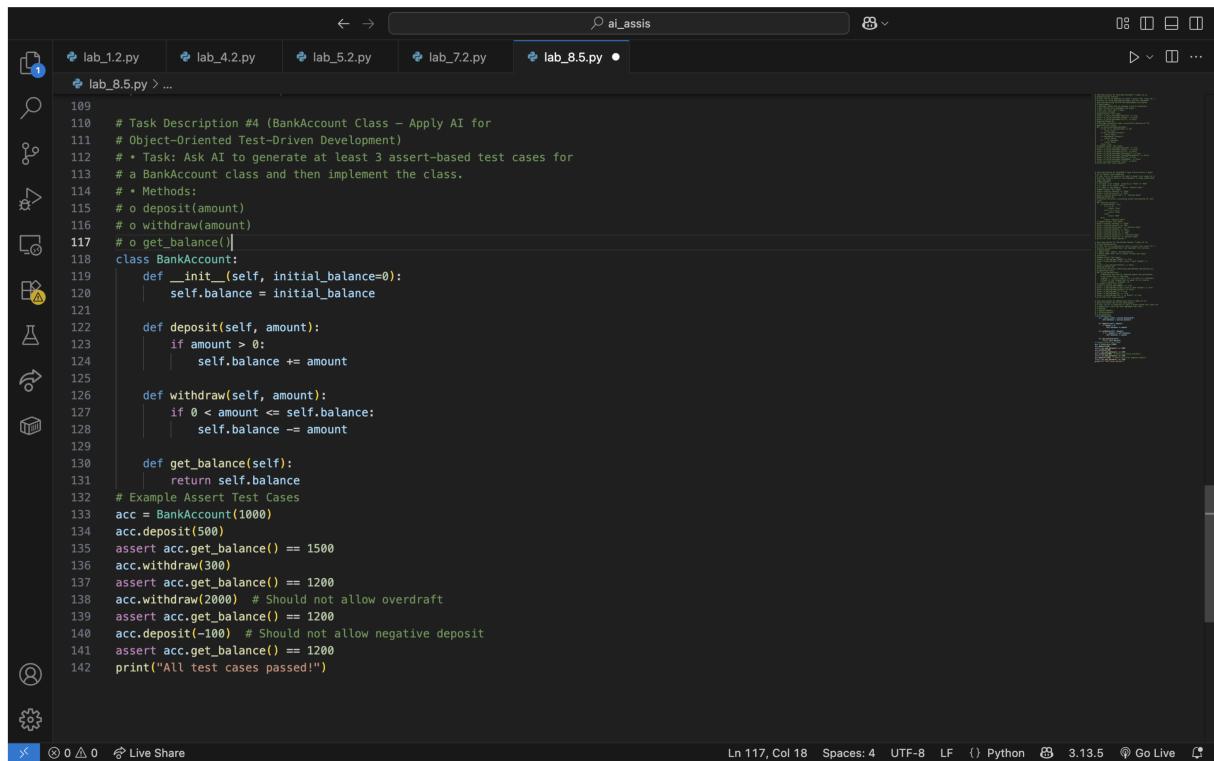
#### Prompt:

Generate at least 3 assert-based test cases for a Python BankAccount class and then implement the class.

Methods required:

- deposit(amount)
- withdraw(amount)
- get\_balance()

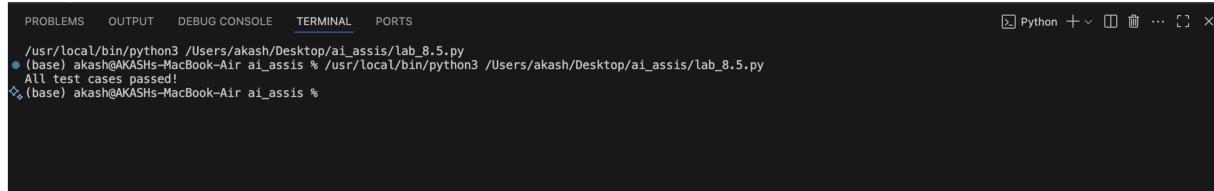
Ensure implementation passes all tests.



```

109  # Task Description #4 (BankAccount Class – Apply AI for
110  # Object-Oriented Test-Driven Development)
111  # * Task: Ask AI to generate at least 3 assert-based test cases for
112  # a BankAccount class and then implement the class.
113  # * Methods:
114  # o deposit(amount)
115  # o withdraw(amount)
116  # o get_balance()
117  # Example Assert Test Cases
118  class BankAccount:
119      def __init__(self, initial_balance=0):
120          self.balance = initial_balance
121
122      def deposit(self, amount):
123          if amount > 0:
124              self.balance += amount
125
126      def withdraw(self, amount):
127          if 0 < amount <= self.balance:
128              self.balance -= amount
129
130      def get_balance(self):
131          return self.balance
132
133  # Example Assert Test Cases
134  acc = BankAccount(1000)
135  acc.deposit(500)
136  acc.withdraw(300)
137  assert acc.get_balance() == 1200
138  acc.withdraw(2000) # Should not allow overdraft
139  assert acc.get_balance() == 1200
140  acc.deposit(-100) # Should not allow negative deposit
141  assert acc.get_balance() == 1200
142  print("All test cases passed!")

```



```

/usr/local/bin/python3 /Users/akash/Desktop/ai_assis/lab_8.5.py
(base) akash@AKASHS-MacBook-Air ai_assis % /usr/local/bin/python3 /Users/akash/Desktop/ai_assis/lab_8.5.py
All test cases passed!
(base) akash@AKASHS-MacBook-Air ai_assis %

```

Code:

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

#### Prompt:

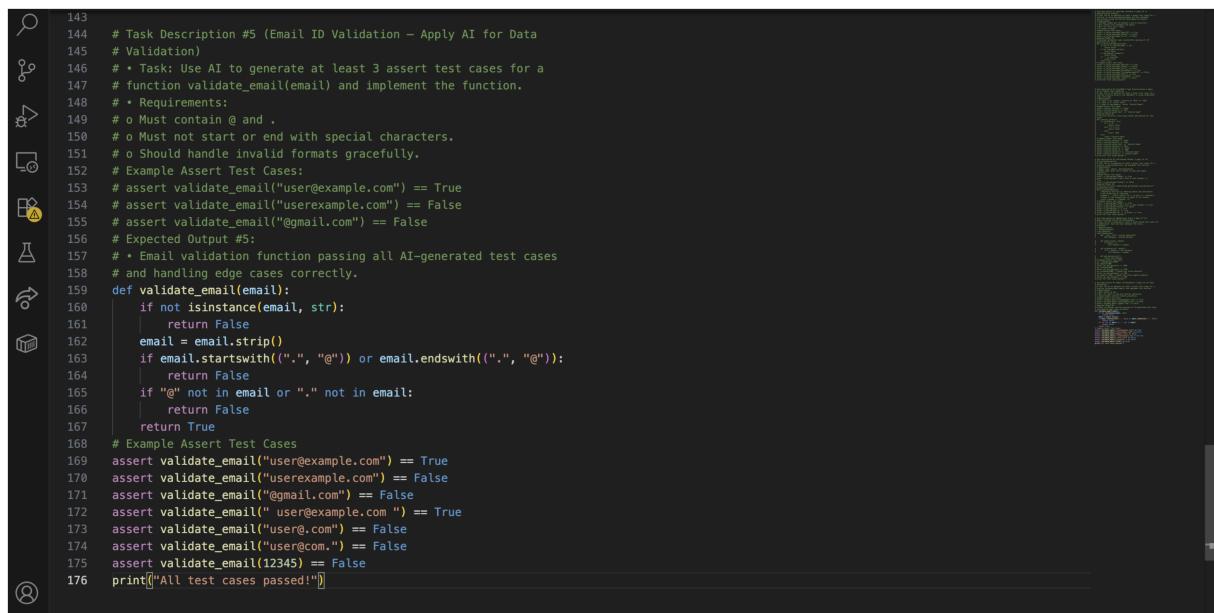
Generate at least 3 assert test cases for a function validate\_email(email) and implement the function.

Rules:

- Must contain @ and .
- Must not start or end with special characters.
- Handle invalid formats gracefully.

Ensure all tests pass.

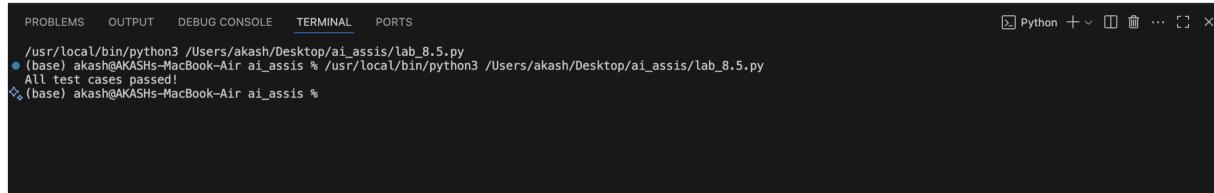
Code:



```

143     # Task Description #5 (Email ID Validation – Apply AI for Data
144     # Validation)
145     # * Task: Use AI to generate at least 3 assert test cases for a
146     # function validate_email(email) and implement the function.
147     # * Requirements:
148     # o Must contain @ and .
149     # o Must not start or end with special characters.
150     # o Should handle invalid formats gracefully.
151     # Example Assert Test Cases:
152     # assert validate_email("user@example.com") == True
153     # assert validate_email("user@example.com") == False
154     # assert validate_email("@gmail.com") == False
155     # Expected Output #5:
156     # * Email validation function passing all AI-generated test cases
157     # and handling edge cases correctly.
158
159     def validate_email(email):
160         if not isinstance(email, str):
161             return False
162         email = email.strip()
163         if email.startswith(".", "@") or email.endswith(".", "@"):
164             return False
165         if "@" not in email or "." not in email:
166             return False
167         return True
168
169     # Example Assert Test Cases
170     assert validate_email("user@example.com") == True
171     assert validate_email("user@example.com") == False
172     assert validate_email("@gmail.com") == False
173     assert validate_email(" user@example.com ") == True
174     assert validate_email("user@.com") == False
175     assert validate_email("user@com.") == False
176     assert validate_email("12345") == False
177
178     print("All test cases passed!")

```



PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

/usr/local/bin/python3 /Users/akash/Desktop/ai\_assis/lab\_8.5.py  
 ● (base) akash@AKASH-MacBook-Air ai\_assis % /usr/local/bin/python3 /Users/akash/Desktop/ai\_assis/lab\_8.5.py  
 All test cases passed!  
 ◁ (base) akash@AKASH-MacBook-Air ai\_assis %