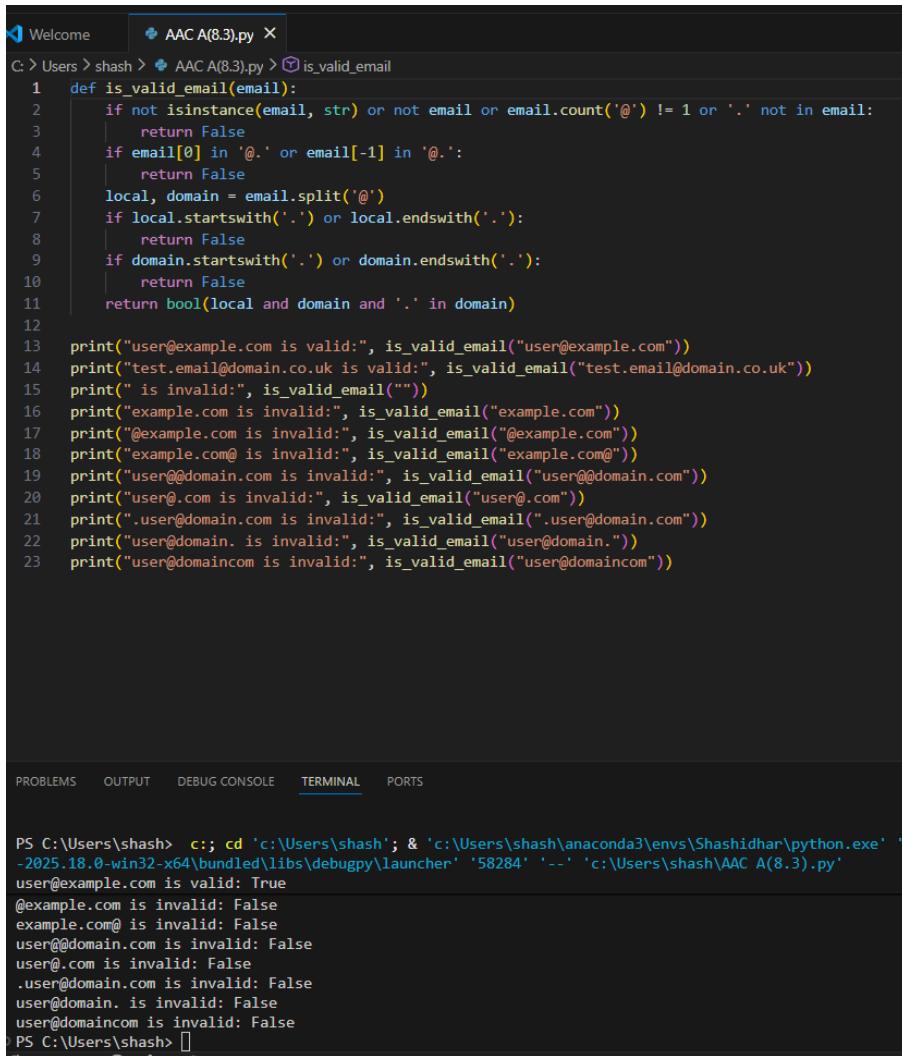


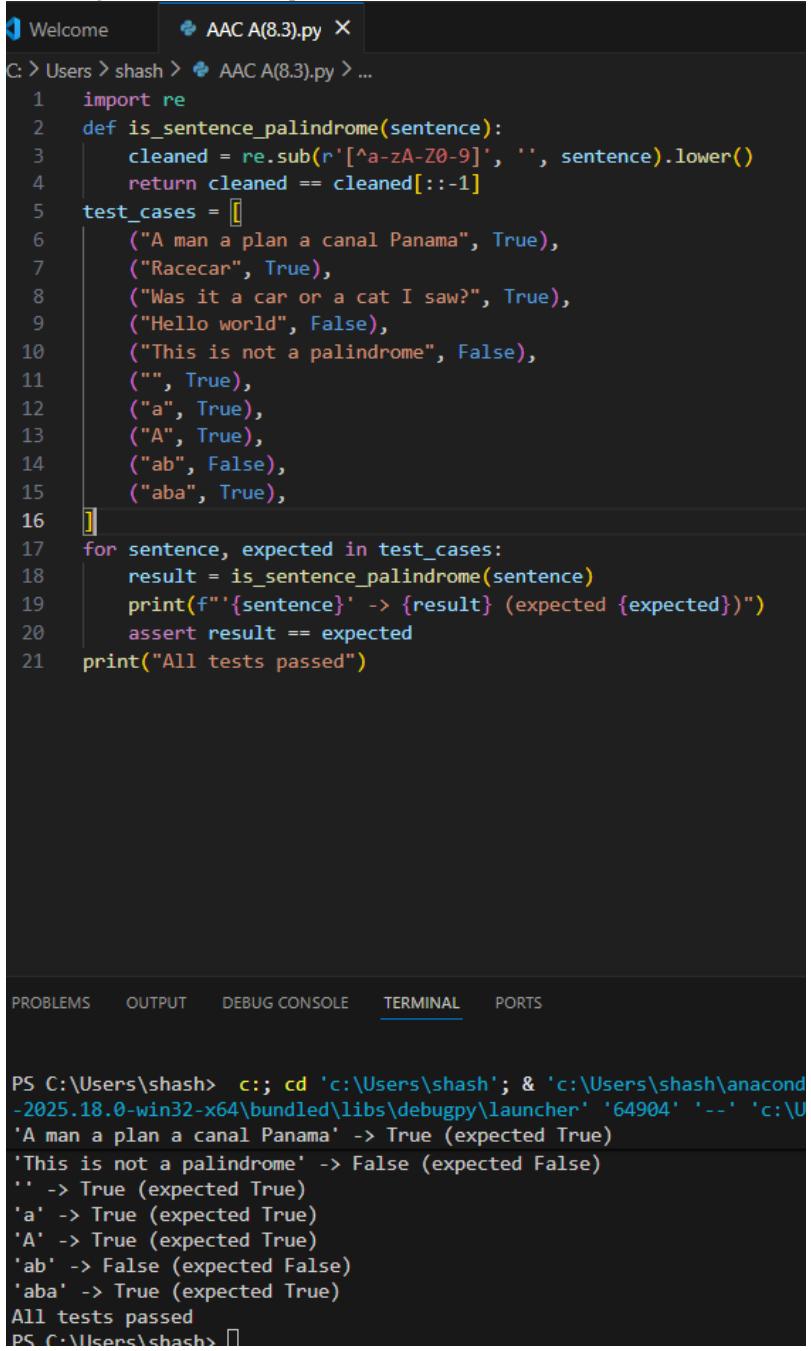
Name:Ch.Shivamani H.No:2303A51806 Batch:26

SCHOOL OF COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE		DEPARTMENT OF COMPUTER SCIENCE ENGINEERING	
<b>Program Name:</b> B. Tech		<b>Assignment Type:</b> Lab	
<b>Course Coordinator Name</b>		Dr. Rishabh Mittal	
<b>Instructor(s) Name</b>		Mr. S Naresh Kumar Ms. B. Swathi Dr. Sasanko Shekhar Gantayat Mr. Md Sallauddin Dr. Mathivanan Mr. Y Srikanth Ms. N Shilpa Dr. Rishabh Mittal (Coordinator) Dr. R. Prashant Kumar Mr. Ankushavali MD Mr. B Viswanath Ms. Sujitha Reddy Ms. A. Anitha Ms. M.Madhuri Ms. Katherashala Swetha Ms. Velpula sumalatha Mr. Bingi Raju	
<b>CourseCode</b>	23CS002PC304	<b>Course Title</b>	AI Assisted Coding
<b>Year/Sem</b>	III/II	<b>Regulation</b>	R23
<b>Date and Day of Assignment</b>	Week3 – Wednesday	<b>Time(s)</b>	23CSBTB01 To 23CSBTB52
<b>Duration</b>	2 Hours	<b>Applicable to Batches</b>	All batches
<b>Assignment Number:</b> 8.3(Present assignment number)/ <b>24</b> (Total number of assignments)			

<b>Q.No.</b>	<b>Question</b>	<i>Expected Time to complete</i>
1	<b>Lab 8: Test-Driven Development with AI – Generating and Working with Test Cases</b> <b>Lab Objectives</b> <ul style="list-style-type: none"> <li>• Introduce TDD using AI</li> <li>• Generate test cases before implementation</li> <li>• Emphasize testing and validation</li> <li>• Encourage clean, reliable code</li> </ul> <b>Lab Outcomes</b> <p>Students will be able to:</p> <ul style="list-style-type: none"> <li>• Write AI-generated test cases</li> </ul>	Week4 - Wednesday

	<ul style="list-style-type: none"> <li>• Implement code using test-first approach</li> <li>• Validate using unittest</li> <li>• Analyze test coverage</li> <li>• Compare AI vs manual tests</li> </ul> <p><b>Task 1: Email Validation using TDD</b></p> <p><b>Scenario</b></p> <p>You are developing a user registration system that requires reliable email input validation.</p> <p><b>Requirements</b></p> <ul style="list-style-type: none"> <li>• Must contain @ and . characters</li> <li>• Must not start or end with special characters</li> <li>• Should not allow multiple @ symbols</li> <li>• AI should generate test cases covering valid and invalid email formats</li> <li>• Implement is_valid_email(email) to pass all AI-generated test cases</li> </ul> <p><b>Expected Output</b></p> <ul style="list-style-type: none"> <li>• Python function for email validation</li> <li>• All AI-generated test cases pass successfully</li> <li>• Invalid email formats are correctly rejected</li> <li>• Valid email formats return True</li> </ul>  <pre> 1  def is_valid_email(email): 2      if not isinstance(email, str) or not email or email.count('@') != 1 or '.' not in email: 3          return False 4      if email[0] in '@.' or email[-1] in '@.': 5          return False 6      local, domain = email.split('@') 7      if local.startswith('.') or local.endswith('.'): 8          return False 9      if domain.startswith('.') or domain.endswith('.'): 10         return False 11     return bool(local and domain and '.' in domain) 12 13 print("user@example.com is valid:", is_valid_email("user@example.com")) 14 print("test.email@domain.co.uk is valid:", is_valid_email("test.email@domain.co.uk")) 15 print(" is invalid:", is_valid_email("")) 16 print("example.com is invalid:", is_valid_email("example.com")) 17 print("@example.com is invalid:", is_valid_email("@example.com")) 18 print("example.com@ is invalid:", is_valid_email("example.com@")) 19 print("user@domain.com is invalid:", is_valid_email("user@domain.com")) 20 print("user@.com is invalid:", is_valid_email("user@.com")) 21 print(".user@domain.com is invalid:", is_valid_email(".user@domain.com")) 22 print("user@domain. is invalid:", is_valid_email("user@domain.")) 23 print("user@domaincom is invalid:", is_valid_email("user@domaincom")) </pre> <p>PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS</p> <pre> PS C:\Users\shash&gt; c;; cd 'c:\Users\shash' &amp; 'c:\Users\shash\anaconda3\envs\Shashidhar\python.exe' 'c:\Users\shash\VSCode\Python\TDD\AAC A(8.3).py' user@example.com is valid: True @example.com is invalid: False example.com@ is invalid: False user@domain.com is invalid: False user@.com is invalid: False .user@domain.com is invalid: False user@domain. is invalid: False user@domaincom is invalid: False PS C:\Users\shash&gt; [] </pre>	
	<p><b>Task 2: Grade Assignment using Loops</b></p> <p><b>Scenario</b></p>	

	<p>You are building an automated grading system for an online examination platform.</p> <p><b>Requirements</b></p> <ul style="list-style-type: none"> <li>AI should generate test cases for <code>assign_grade(score)</code> where:</li> <li>- 90–100 → A</li> <li>- 80–89 → B</li> <li>- 70–79 → C</li> <li>- 60–69 → D</li> <li>- Below 60 → F</li> <li>Include boundary values (60, 70, 80, 90)</li> <li>Include invalid inputs such as -5, 105, "eighty"</li> <li>Implement the function using a test-driven approach</li> </ul> <p><b>Expected Output</b></p> <ul style="list-style-type: none"> <li>Grade assignment function implemented in Python</li> <li>Boundary values handled correctly</li> <li>Invalid inputs handled gracefully</li> <li>All AI-generated test cases pass</li> </ul> <pre> 1  Welcome AAC A(8.3).py ... 2  C: &gt; Users &gt; shash &gt; AAC A(8.3).py &gt; ... 3  1 4  2  def assign_grade(score): 5  3      if not isinstance(score, (int, float)) or not (0 &lt;= score &lt;= 100): 6  4          return "Invalid" 7  5      grades = [('A', 90), ('B', 80), ('C', 70), ('D', 60), ('F', 0)] 8  6      for grade, min_score in grades: 9  7          if score &gt;= min_score: 10 8              return grade 11 9      return 'F' 12 10 13 11  print("90:", assign_grade(90)) 14 12  print("89:", assign_grade(89)) 15 13  print("80:", assign_grade(80)) 16 14  print("79:", assign_grade(79)) 17 15  print("70:", assign_grade(70)) 18 16  print("69:", assign_grade(69)) 19 17  print("60:", assign_grade(60)) 20 18  print("59:", assign_grade(59)) 21 19  print("0:", assign_grade(0)) 22 20  print("100:", assign_grade(100)) 23 21  print("-5:", assign_grade(-5)) 24 22  print("105:", assign_grade(105)) 25 23  print("eighty:", assign_grade("eighty")) </pre> <p>PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS</p> <pre> PS C:\Users\shash&gt; c;; cd 'c:\Users\shash'; &amp; 'c:\Users\shash\anaconda3\envs\Shash-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '60316' '--' 'c:\Users\shash\AA 90: A 60: D 59: F 0: F 100: A -5: Invalid 105: Invalid eighty: Invalid </pre>	
	<p><b>Task 3: Sentence Palindrome Checker</b></p> <p><b>Scenario</b></p> <p>You are developing a text-processing utility to analyze sentences.</p> <p><b>Requirements</b></p> <ul style="list-style-type: none"> <li>AI should generate test cases for <code>is_sentence_palindrome(sentence)</code></li> </ul>	

	<ul style="list-style-type: none"> <li>• Ignore case, spaces, and punctuation</li> <li>• Test both palindromic and non-palindromic sentences</li> <li>• Example:           <ul style="list-style-type: none"> <li>- "A man a plan a canal Panama" → True</li> </ul> </li> </ul> <p>Expected Output</p> <ul style="list-style-type: none"> <li>• Function correctly identifies sentence palindromes</li> <li>• Case and punctuation are ignored</li> <li>• Returns True or False accurately</li> <li>• All AI-generated test cases pass</li> </ul>  <pre> 1  import re 2  def is_sentence_palindrome(sentence): 3      cleaned = re.sub(r'[^\w\s]', '', sentence).lower() 4      return cleaned == cleaned[::-1] 5  test_cases = [ 6      ("A man a plan a canal Panama", True), 7      ("Racecar", True), 8      ("Was it a car or a cat I saw?", True), 9      ("Hello world", False), 10     ("This is not a palindrome", False), 11     ("", True), 12     ("a", True), 13     ("A", True), 14     ("ab", False), 15     ("aba", True), 16 ] 17 for sentence, expected in test_cases: 18     result = is_sentence_palindrome(sentence) 19     print(f'{sentence} -&gt; {result} (expected {expected})') 20     assert result == expected 21 print("All tests passed") </pre> <p>PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS</p> <pre> PS C:\Users\shash&gt; c:; cd 'c:\Users\shash'; &amp; 'c:\Users\shash\anaconda-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '64904' '--' 'c:\U 'A man a plan a canal Panama' -&gt; True (expected True) 'This is not a palindrome' -&gt; False (expected False) '' -&gt; True (expected True) 'a' -&gt; True (expected True) 'A' -&gt; True (expected True) 'ab' -&gt; False (expected False) 'aba' -&gt; True (expected True) All tests passed PS C:\Users\shash&gt; </pre>	
	<p><b>Task 4: ShoppingCart Class</b></p> <p><b>Scenario</b></p> <p>You are designing a basic shopping cart module for an e-commerce application.</p>	

Name:Ch.Shivamani H.No:2303A51806 Batch:26

	<p><b>Requirements</b></p> <ul style="list-style-type: none"><li>• AI should generate test cases for the ShoppingCart class</li><li>• Class must include the following methods:<ul style="list-style-type: none"><li>- add_item(name, price)</li><li>- remove_item(name)</li><li>- total_cost()</li></ul></li><li>• Validate correct addition, removal, and cost calculation</li><li>• Handle empty cart scenarios</li></ul> <p><b>Expected Output</b></p> <ul style="list-style-type: none"><li>• Fully implemented ShoppingCart class</li><li>• All methods pass AI-generated test cases</li><li>• Total cost is calculated accurately</li><li>• Items are added and removed correctly</li></ul>	
--	--	--

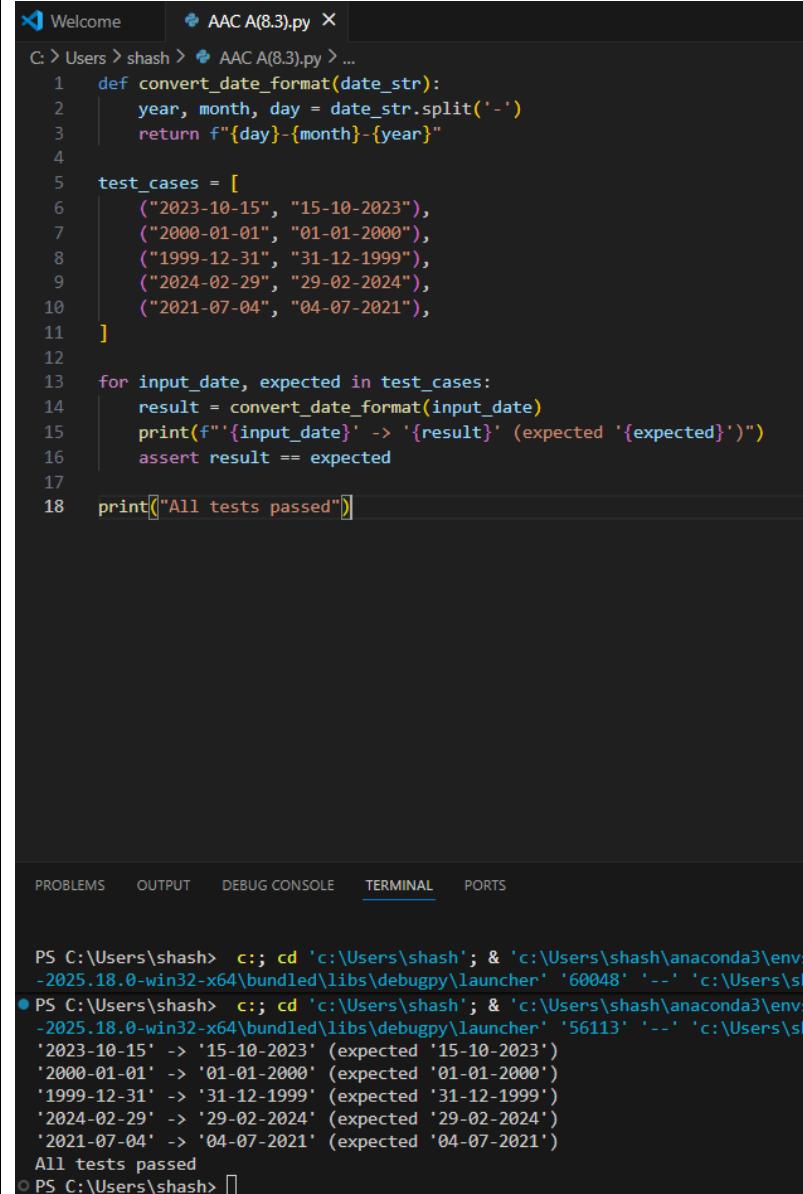
Name:Ch.Shivamani H.No:2303A51806 Batch:26

The screenshot shows a code editor interface with a dark theme. The top bar includes tabs for 'Welcome' and 'AAC A(8.3).py'. The main area displays Python code for a ShoppingCart class, with line numbers on the left. The code defines methods for initializing the cart, adding items, removing items, and calculating the total cost. It also includes several assert statements to verify the correctness of these methods. Below the code editor is a terminal window showing the execution of the test script. The terminal output indicates that all tests passed.

```
C: > Users > shash > AAC A(8.3).py > ShoppingCart > _init_
1  class ShoppingCart:
2      def __init__(self):
3          self.items = []
4      def add_item(self, name, price):
5          self.items.append((name, price))
6      def remove_item(self, name):
7          for i, (n, p) in enumerate(self.items):
8              if n == name:
9                  del self.items[i]
10                 break
11      def total_cost(self):
12          return sum(price for name, price in self.items)
13  cart = ShoppingCart()
14  assert cart.total_cost() == 0
15  cart.add_item("apple", 1.0)
16  cart.add_item("banana", 2.0)
17  assert cart.total_cost() == 3.0
18  cart.add_item("apple", 1.0)
19  assert cart.total_cost() == 4.0
20  cart.remove_item("apple")
21  assert cart.total_cost() == 3.0
22  cart.remove_item("banana")
23  assert cart.total_cost() == 1.0
24  cart.remove_item("orange")
25  assert cart.total_cost() == 1.0
26  cart.remove_item("apple")
27  assert cart.total_cost() == 0
28  cart.add_item("milk", 3.5)
29  cart.add_item("bread", 2.5)
30  cart.add_item("milk", 3.5)
31  assert cart.total_cost() == 9.5
32  cart.remove_item("milk")
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

PS C:\Users\shash> c:; cd 'c:\Users\shash'; & 'c:\Users\shash\anaconda3\envs\py38\python.exe' 'c:\Users\shash\ai-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '64904' '--'
'A man a plan a canal Panama' -> True (expected True)
'A' -> True (expected True)
'ab' -> False (expected False)
'aba' -> True (expected True)
All tests passed
PS C:\Users\shash> c:; cd 'c:\Users\shash'; & 'c:\Users\shash\anaconda3\envs\py38\python.exe' 'c:\Users\shash\ai-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '60048' '--'
All tests passed
PS C:\Users\shash>
```

	<pre> Welcome AAC A(8.3).py X C: &gt; Users &gt; shash &gt; AAC A(8.3).py &gt; ShoppingCart &gt; _init_ 1  class ShoppingCart: 6    def remove_item(self, name): 10   break 11   def total_cost(self): 12     return sum(price for name, price in self.items) 13 cart = ShoppingCart() 14 assert cart.total_cost() == 0 15 cart.add_item("apple", 1.0) 16 cart.add_item("banana", 2.0) 17 assert cart.total_cost() == 3.0 18 cart.add_item("apple", 1.0) 19 assert cart.total_cost() == 4.0 20 cart.remove_item("apple") 21 assert cart.total_cost() == 3.0 22 cart.remove_item("banana") 23 assert cart.total_cost() == 1.0 24 cart.remove_item("orange") 25 assert cart.total_cost() == 1.0 26 cart.remove_item("apple") 27 assert cart.total_cost() == 0 28 cart.add_item("milk", 3.5) 29 cart.add_item("bread", 2.5) 30 cart.add_item("milk", 3.5) 31 assert cart.total_cost() == 9.5 32 cart.remove_item("milk") 33 assert cart.total_cost() == 6.0 34 print("All tests passed") </pre> <p>PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS</p> <pre> PS C:\Users\shash&gt; c;; cd 'c:\Users\shash'; &amp; 'c:\Users\shash -2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '64904' '-A man a plan a canal Panama' -&gt; True (expected True) 'A' -&gt; True (expected True) 'ab' -&gt; False (expected False) 'aba' -&gt; True (expected True) All tests passed ● PS C:\Users\shash&gt; c;; cd 'c:\Users\shash'; &amp; 'c:\Users\shash -2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '60048' '- All tests passed PS C:\Users\shash&gt; </pre>	
	<p><b>Task 5: Date Format Conversion</b></p> <p><b>Scenario</b></p> <p>You are creating a utility function to convert date formats for reports.</p> <p><b>Requirements</b></p> <ul style="list-style-type: none"> <li>AI should generate test cases for convert_date_format(date_str)</li> <li>Input format must be "YYYY-MM-DD"</li> <li>Output format must be "DD-MM-YYYY"</li> <li>Example:</li> </ul>	

	<p>- "2023-10-15" → "15-10-2023"</p> <p><b>Expected Output</b></p> <ul style="list-style-type: none"><li>• Date conversion function implemented in Python</li><li>• Correct format conversion for all valid inputs</li><li>• All AI-generated test cases pass successfully</li></ul>  <p>The screenshot shows a code editor with a Python file named AAC A(8.3).py. The code defines a function convert_date_format that takes a date string and returns it in a different format. It also contains a list of test cases and a loop that prints each input date followed by its expected result and the actual result from the function. The terminal below shows the execution of the script and the output of the test cases, which all pass.</p> <pre>C:\Users\shash&gt; shash &gt; AAC A(8.3).py &gt; ... 1 def convert_date_format(date_str): 2     year, month, day = date_str.split('-') 3     return f"{day}-{month}-{year}" 4 5 test_cases = [ 6     ("2023-10-15", "15-10-2023"), 7     ("2000-01-01", "01-01-2000"), 8     ("1999-12-31", "31-12-1999"), 9     ("2024-02-29", "29-02-2024"), 10    ("2021-07-04", "04-07-2021"), 11 ] 12 13 for input_date, expected in test_cases: 14     result = convert_date_format(input_date) 15     print(f"'{input_date}' -&gt; '{result}' (expected '{expected}')") 16     assert result == expected 17 18 print(["All tests passed"])  PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS  PS C:\Users\shash&gt; c:&amp; cd 'c:\Users\shash'; &amp; 'c:\Users\shash\anaconda3\envs\2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '60048' '--' 'c:\Users\shash\AAC A(8.3).py' ● PS C:\Users\shash&gt; c:&amp; cd 'c:\Users\shash'; &amp; 'c:\Users\shash\anaconda3\envs\2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '56113' '--' 'c:\Users\shash\AAC A(8.3).py' '2023-10-15' -&gt; '15-10-2023' (expected '15-10-2023') '2000-01-01' -&gt; '01-01-2000' (expected '01-01-2000') '1999-12-31' -&gt; '31-12-1999' (expected '31-12-1999') '2024-02-29' -&gt; '29-02-2024' (expected '29-02-2024') '2021-07-04' -&gt; '04-07-2021' (expected '04-07-2021') All tests passed ○ PS C:\Users\shash&gt;</pre> <p><b>Note:</b> Report should be submitted as a word document for all tasks in a single document with prompts, comments &amp; code explanation, and output and if required, screenshots.</p>	
--	--	--