

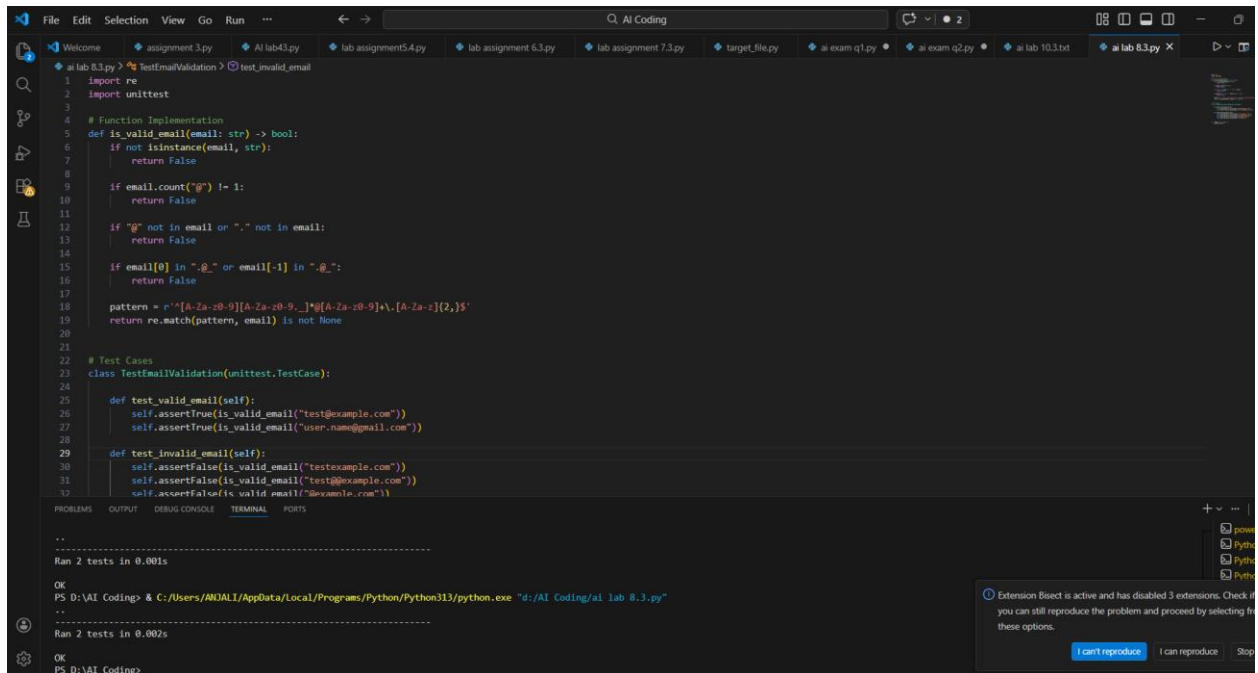
2303a51739

Lab assignment 8.3

Task 1: Email Validation using TDD

Scenario

You are developing a user registration system that requires reliable email input validation



```
1 import re
2 import unittest
3
4 # Function Implementation
5 def is_valid_email(email: str) -> bool:
6     if not isinstance(email, str):
7         return False
8
9     if email.count("@") != 1:
10        return False
11
12    if "@" not in email or "." not in email:
13        return False
14
15    if email[0] in ".@_" or email[-1] in ".@_":
16        return False
17
18    pattern = r'^[A-Za-z0-9][A-Za-z0-9_]*@[A-Za-z0-9]+\.[A-Za-z]{2}$'
19    return re.match(pattern, email) is not None
20
21
22 # Test Cases
23 class TestEmailValidation(unittest.TestCase):
24
25     def test_valid_email(self):
26         self.assertTrue(is_valid_email("test@example.com"))
27         self.assertTrue(is_valid_email("user.name@gmail.com"))
28
29     def test_invalid_email(self):
30         self.assertFalse(is_valid_email("testexample.com"))
31         self.assertFalse(is_valid_email("test@example.com"))
32         self.assertFalse(is_valid_email("@example.com"))
33         self.assertFalse(is_valid_email("test@com"))
34
35 if __name__ == "__main__":
36     unittest.main()
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

..

Ran 2 tests in 0.001s

OK

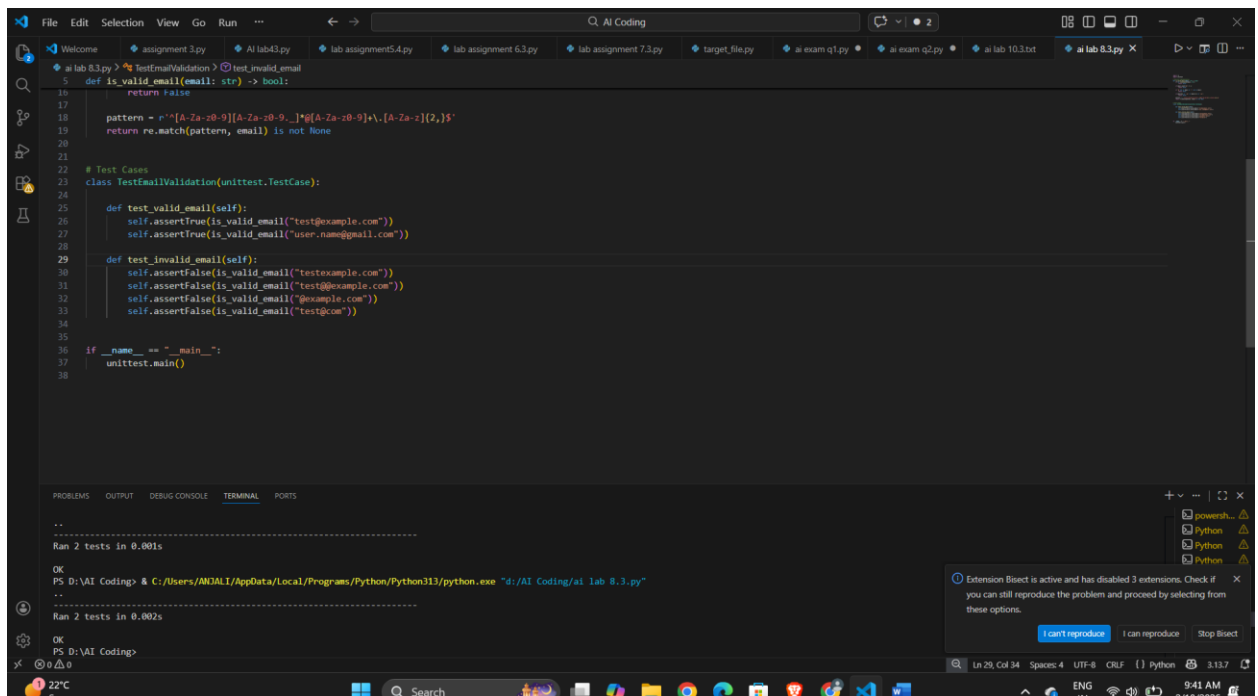
PS D:\AI Coding> & C:/Users/ANDALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/ai lab 8.3.py"

..

Ran 2 tests in 0.002s

OK

PS D:\AI Coding>



```
5 def is_valid_email(email: str) -> bool:
6     return False
7
8
9
10
11
12
13
14
15
16
17
18 pattern = r'^[A-Za-z0-9][A-Za-z0-9_]*@[A-Za-z0-9]+\.[A-Za-z]{2}$'
19 return re.match(pattern, email) is not None
20
21
22 # Test Cases
23 class TestEmailValidation(unittest.TestCase):
24
25     def test_valid_email(self):
26         self.assertTrue(is_valid_email("test@example.com"))
27         self.assertTrue(is_valid_email("user.name@gmail.com"))
28
29     def test_invalid_email(self):
30         self.assertFalse(is_valid_email("testexample.com"))
31         self.assertFalse(is_valid_email("test@example.com"))
32         self.assertFalse(is_valid_email("@example.com"))
33         self.assertFalse(is_valid_email("test@com"))
34
35 if __name__ == "__main__":
36     unittest.main()
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

..

Ran 2 tests in 0.001s

OK

PS D:\AI Coding> & C:/Users/ANDALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/ai lab 8.3.py"

..

Ran 2 tests in 0.002s

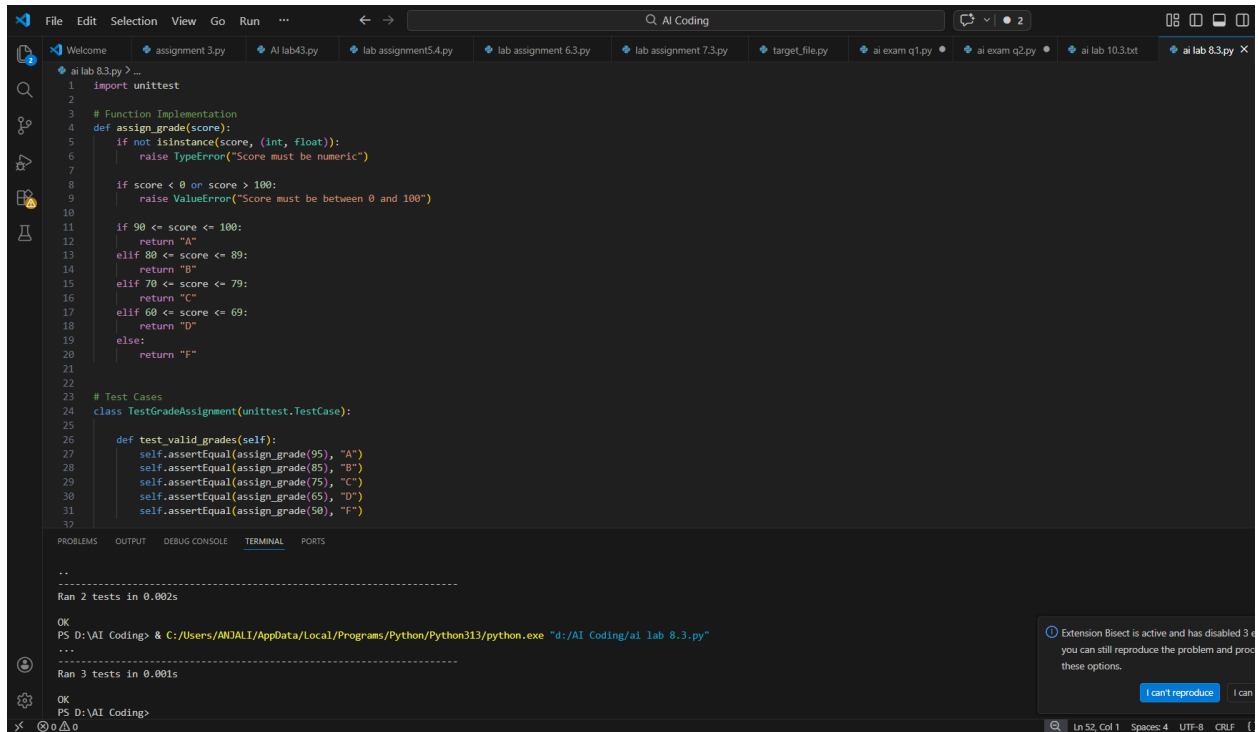
OK

PS D:\AI Coding>

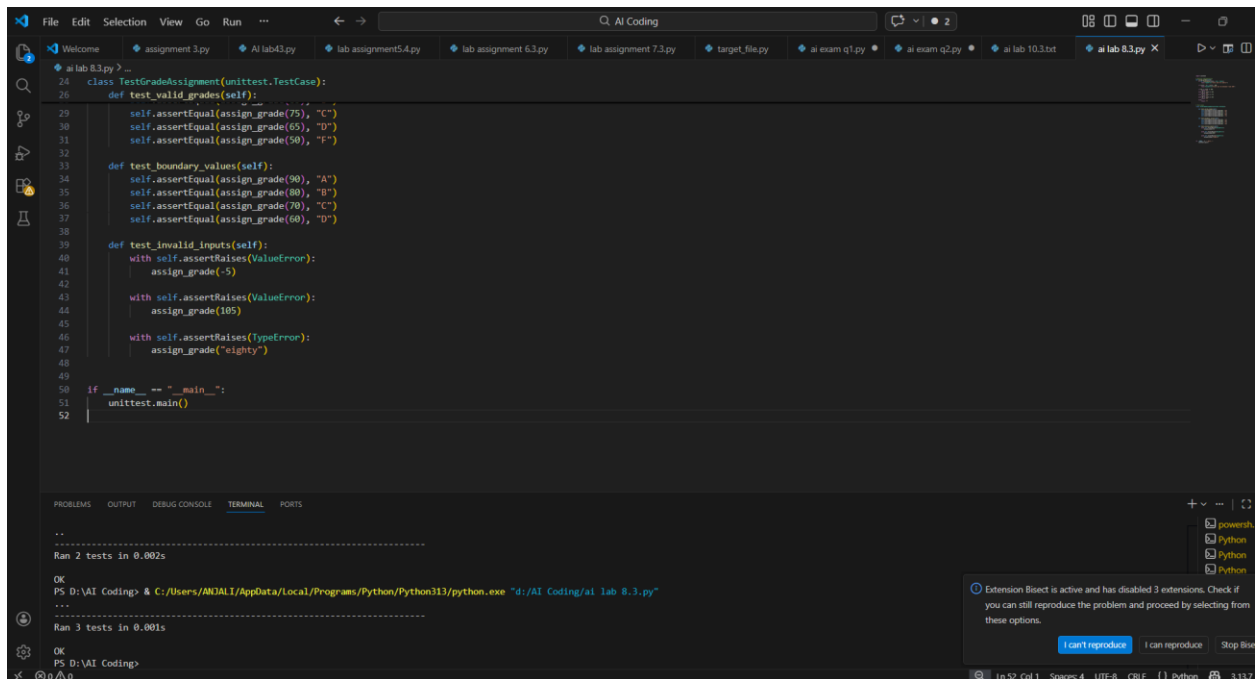
Task 2: Grade Assignment using Loops

Scenario

You are building an automated grading system for an online examination platform.



```
File Edit Selection View Go Run ... AI Coding
ai lab 8.3.py > ...
1 import unittest
2
3 # Function Implementation
4 def assign_grade(score):
5     if not isinstance(score, (int, float)):
6         raise TypeError("Score must be numeric")
7
8     if score < 0 or score > 100:
9         raise ValueError("Score must be between 0 and 100")
10
11     if 90 <= score <= 100:
12         return "A"
13     elif 80 <= score <= 89:
14         return "B"
15     elif 70 <= score <= 69:
16         return "C"
17     elif 60 <= score <= 59:
18         return "D"
19     else:
20         return "F"
21
22 # Test Cases
23 class TestGradeAssignment(unittest.TestCase):
24
25     def test_valid_grades(self):
26         self.assertEqual(assign_grade(95), "A")
27         self.assertEqual(assign_grade(85), "B")
28         self.assertEqual(assign_grade(75), "C")
29         self.assertEqual(assign_grade(65), "D")
30         self.assertEqual(assign_grade(50), "F")
31
32
33 PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
34
35 ..
36 Ran 2 tests in 0.002s
37
38 OK
39 PS D:\AI Coding> & C:/Users/NDALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/ai lab 8.3.py"
40 ...
41 Ran 3 tests in 0.001s
42
43 OK
44 PS D:\AI Coding>
```

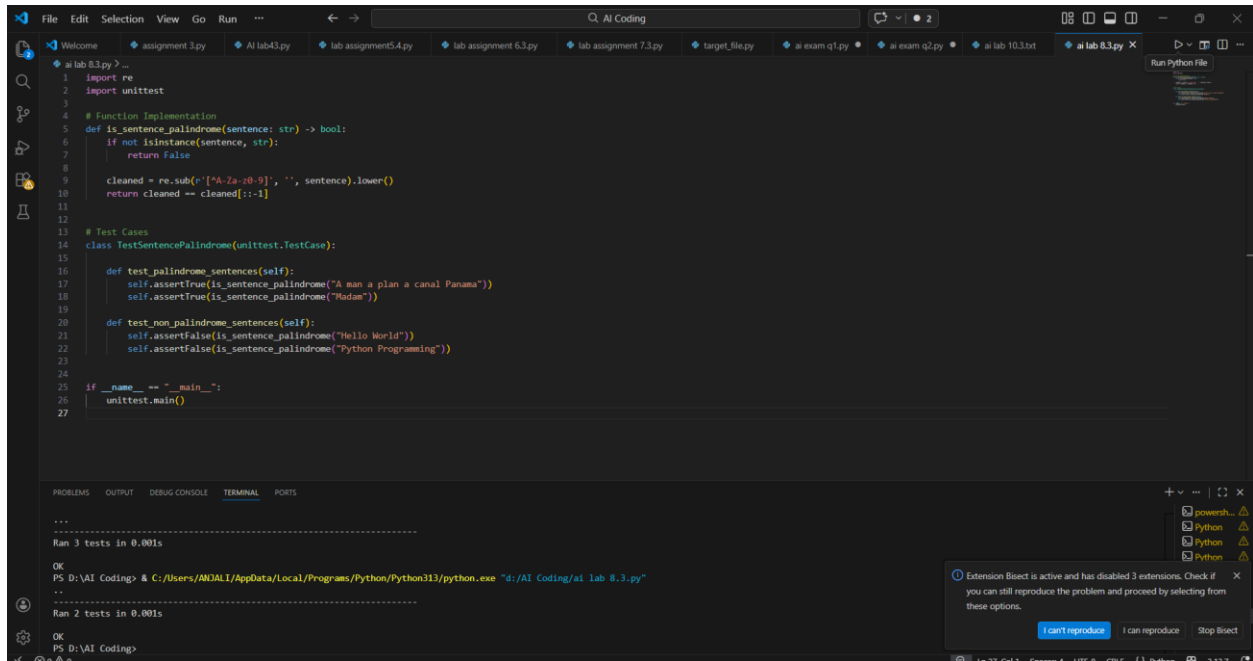


```
File Edit Selection View Go Run ... AI Coding
ai lab 8.3.py > ...
24 class TestGradeAssignment(unittest.TestCase):
25
26     def test_valid_grades(self):
27         self.assertEqual(assign_grade(75), "C")
28         self.assertEqual(assign_grade(65), "D")
29         self.assertEqual(assign_grade(50), "F")
30
31     def test_boundary_values(self):
32         self.assertEqual(assign_grade(90), "A")
33         self.assertEqual(assign_grade(80), "B")
34         self.assertEqual(assign_grade(70), "C")
35         self.assertEqual(assign_grade(60), "D")
36
37     def test_invalid_inputs(self):
38         with self.assertRaises(ValueError):
39             assign_grade(-5)
40
41         with self.assertRaises(ValueError):
42             assign_grade(105)
43
44         with self.assertRaises(TypeError):
45             assign_grade("eighty")
46
47
48 if __name__ == "__main__":
49     unittest.main()
50
51
52 PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
53
54 ..
55 Ran 2 tests in 0.002s
56
57 OK
58 PS D:\AI Coding> & C:/Users/NDALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/ai lab 8.3.py"
59 ...
60 Ran 3 tests in 0.001s
61
62 OK
63 PS D:\AI Coding>
```

Task 3: Sentence Palindrome Checker

Scenario

You are developing a text-processing utility to analyze sentences.



The screenshot shows a Visual Studio Code editor with a Python file named `ai lab 8.3.py`. The code implements a function `is_sentence_palindrome` and a test class `testSentencePalindrome`. The terminal output shows the execution of the tests, which passed successfully.

```
1 import re
2 import unittest
3
4 # Function Implementation
5 def is_sentence_palindrome(sentence: str) -> bool:
6     if not isinstance(sentence, str):
7         return False
8
9     cleaned = re.sub(r'[^A-Za-z0-9]', '', sentence).lower()
10    return cleaned == cleaned[::-1]
11
12
13 # Test Cases
14 class testSentencePalindrome(unittest.TestCase):
15
16     def test_palindrome_sentences(self):
17         self.assertTrue(is_sentence_palindrome("A man a plan a canal Panama"))
18         self.assertTrue(is_sentence_palindrome("Madam"))
19
20     def test_non_palindrome_sentences(self):
21         self.assertFalse(is_sentence_palindrome("Hello World"))
22         self.assertFalse(is_sentence_palindrome("Python Programming"))
23
24
25 if __name__ == "__main__":
26     unittest.main()
27
```

Terminal Output:

```
...
-----
Ran 3 tests in 0.001s
OK
PS D:\AI Coding> & C:\Users\MDALI\AppData\Local\Programs\Python\Python313\python.exe "d:/AI Coding/ai lab 8.3.py"
...
-----
Ran 2 tests in 0.001s
OK
PS D:\AI Coding>
```

Extension Bisect notification:

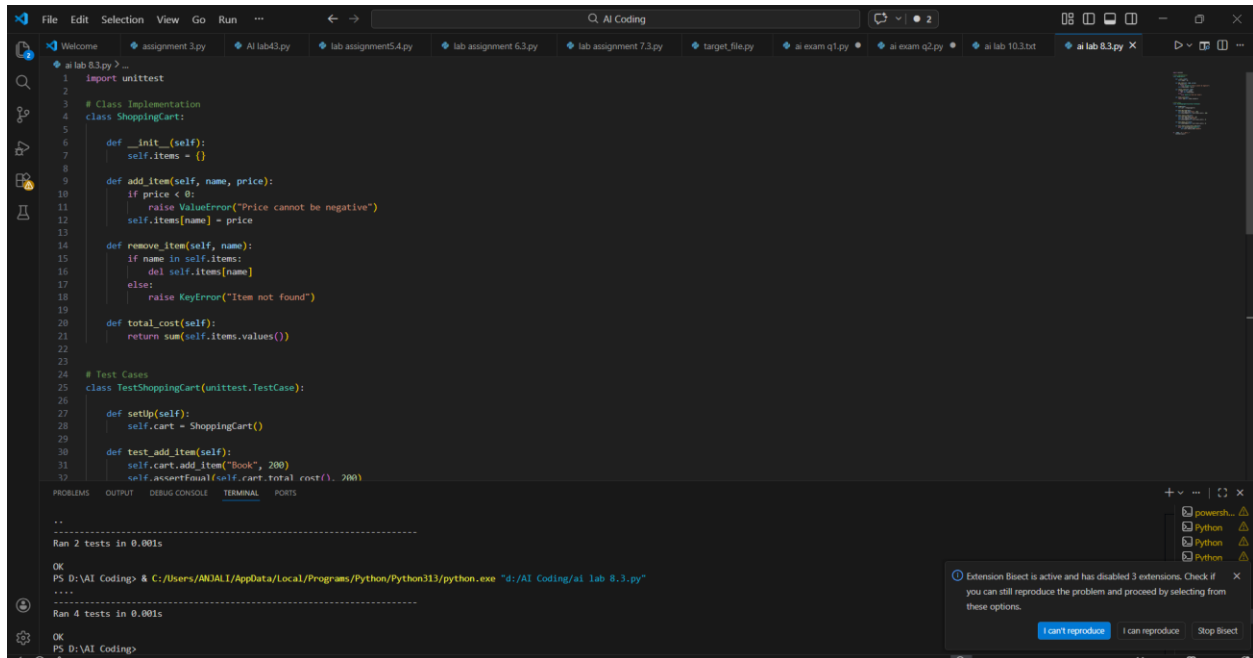
Extension Bisect is active and has disabled 3 extensions. Check if you can still reproduce the problem and proceed by selecting from these options.

[I can't reproduce] [I can reproduce] [Stop Bisect]

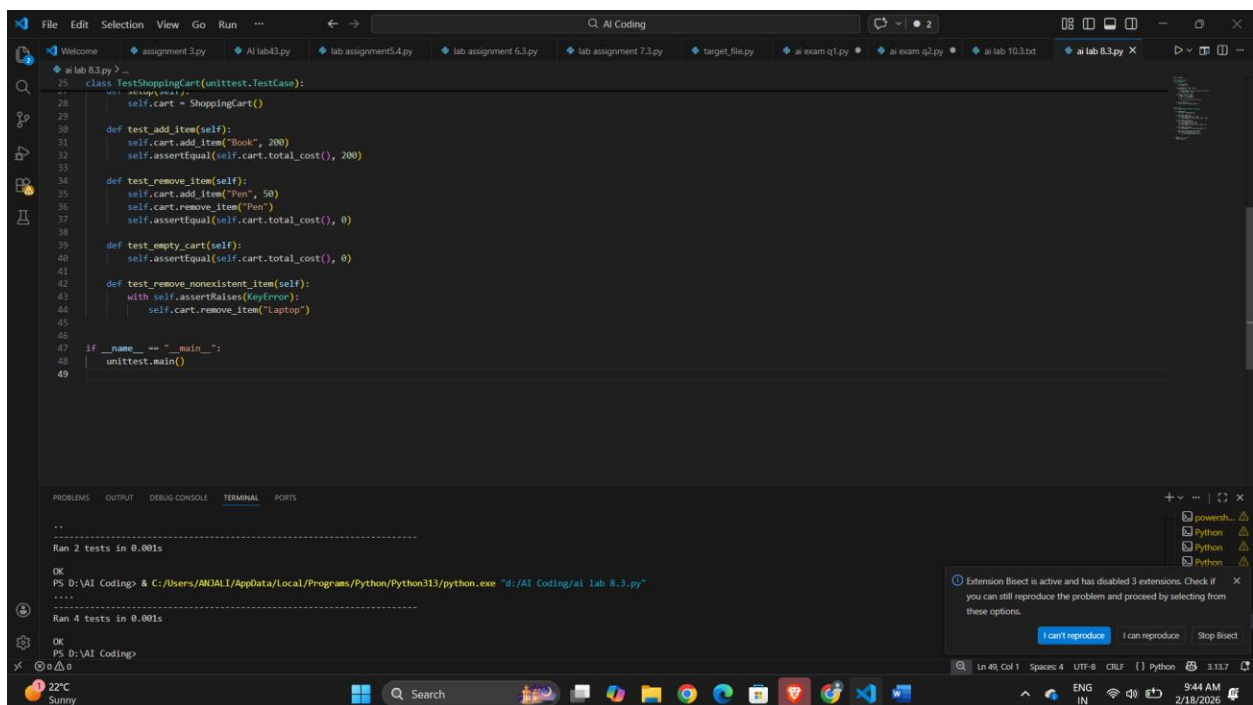
Task 4: ShoppingCart Class

Scenario

You are designing a basic shopping cart module for an e-commerce application.



```
1 import unittest
2
3 # Class Implementation
4 class ShoppingCart:
5
6     def __init__(self):
7         self.items = {}
8
9     def add_item(self, name, price):
10         if price < 0:
11             raise ValueError("Price cannot be negative")
12         self.items[name] = price
13
14     def remove_item(self, name):
15         if name in self.items:
16             del self.items[name]
17         else:
18             raise KeyError("Item not found")
19
20     def total_cost(self):
21         return sum(self.items.values())
22
23
24 # Test Cases
25 class TestShoppingCart(unittest.TestCase):
26
27     def setUp(self):
28         self.cart = ShoppingCart()
29
30     def test_add_item(self):
31         self.cart.add_item("Book", 200)
32         self.assertEqual(self.cart.total_cost(), 200)
33
34
35 ..
36 -----
37 Ran 2 tests in 0.001s
38
39 OK
40 PS D:\AI Coding> & C:\Users\ANDALI\AppData\Local\Programs\Python\Python313\python.exe "d:/AI Coding/ai lab 8.3.py"
41 -----
42 Ran 4 tests in 0.001s
43
44 OK
45 PS D:\AI Coding>
```

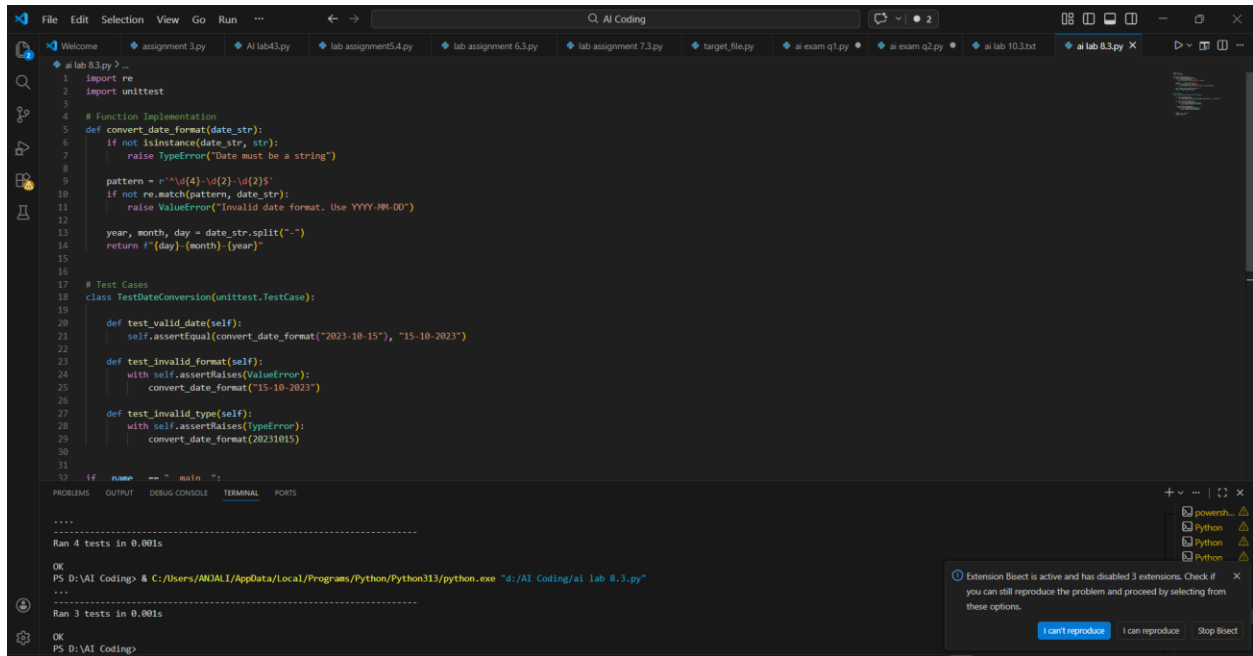


```
25 class TestShoppingCart(unittest.TestCase):
26     def setUp(self):
27         self.cart = ShoppingCart()
28
29     def test_add_item(self):
30         self.cart.add_item("Book", 200)
31         self.assertEqual(self.cart.total_cost(), 200)
32
33     def test_remove_item(self):
34         self.cart.add_item("Pen", 50)
35         self.cart.remove_item("Pen")
36         self.assertEqual(self.cart.total_cost(), 0)
37
38     def test_empty_cart(self):
39         self.assertEqual(self.cart.total_cost(), 0)
40
41     def test_remove_nonexistent_item(self):
42         with self.assertRaises(KeyError):
43             self.cart.remove_item("Laptop")
44
45
46 if __name__ == "__main__":
47     unittest.main()
48
49
```

Task 5: Date Format Conversion

Scenario

You are creating a utility function to convert date formats for reports.



```
1 import re
2 import unittest
3
4 # Function Implementation
5 def convert_date_format(date_str):
6     if not isinstance(date_str, str):
7         raise TypeError("Date must be a string")
8     pattern = r"^\d{4}-\d{2}-\d{2}$"
9     if not re.match(pattern, date_str):
10         raise ValueError("Invalid date format. Use YYYY-MM-DD")
11
12     year, month, day = date_str.split("-")
13     return f"{day}-{month}-{year}"
14
15 # Test Cases
16 class TestDateConversion(unittest.TestCase):
17
18     def test_valid_date(self):
19         self.assertEqual(convert_date_format("2023-10-15"), "15-10-2023")
20
21     def test_invalid_format(self):
22         with self.assertRaises(ValueError):
23             convert_date_format("15-10-2023")
24
25     def test_invalid_type(self):
26         with self.assertRaises(TypeError):
27             convert_date_format(20231015)
28
29 if __name__ == "__main__":
30     unittest.main()
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

.....

Ran 4 tests in 0.001s

OK

PS D:\AI Coding> & C:\Users\ANDALI\AppData\Local\Programs\Python\Python313\python.exe "d:/AI Coding/ai_lab 8.3.py"

.....

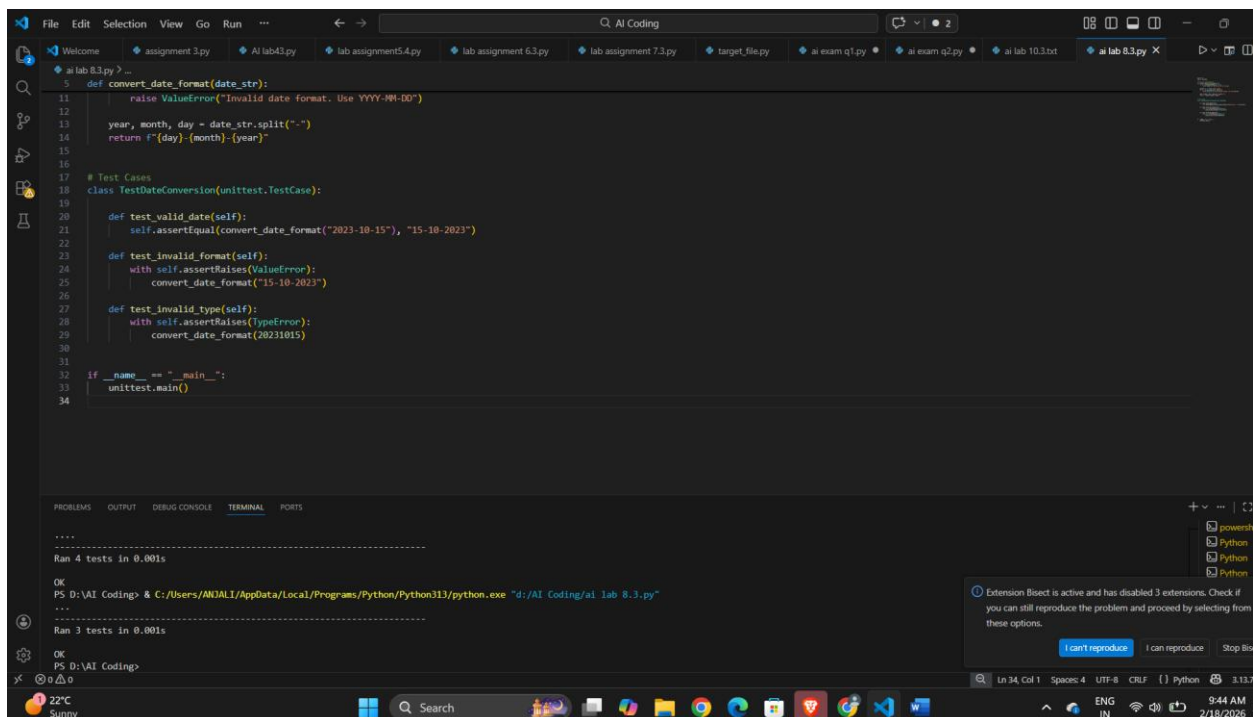
Ran 3 tests in 0.001s

OK

PS D:\AI Coding>

Extension Bisect is active and has disabled 3 extensions. Check if you can still reproduce the problem and proceed by selecting from these options.

I can't reproduce I can reproduce Stop Bisect



```
5 def convert_date_format(date_str):
6     raise ValueError("Invalid date format. Use YYYY-MM-DD")
7
8     year, month, day = date_str.split("-")
9     return f"{day}-{month}-{year}"
10
11 # Test Cases
12 class TestDateConversion(unittest.TestCase):
13
14     def test_valid_date(self):
15         self.assertEqual(convert_date_format("2023-10-15"), "15-10-2023")
16
17     def test_invalid_format(self):
18         with self.assertRaises(ValueError):
19             convert_date_format("15-10-2023")
20
21     def test_invalid_type(self):
22         with self.assertRaises(TypeError):
23             convert_date_format(20231015)
24
25 if __name__ == "__main__":
26     unittest.main()
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

.....

Ran 4 tests in 0.001s

OK

PS D:\AI Coding> & C:\Users\ANDALI\AppData\Local\Programs\Python\Python313\python.exe "d:/AI Coding/ai_lab 8.3.py"

.....

Ran 3 tests in 0.001s

OK

PS D:\AI Coding>

Extension Bisect is active and has disabled 3 extensions. Check if you can still reproduce the problem and proceed by selecting from these options.

I can't reproduce I can reproduce Stop Bisect

Ln 34, Col 1 Spaces: 4 UTF-8 CRLF Python 3.13.7

22°C Sunny

9:44 AM 2/18/2026

