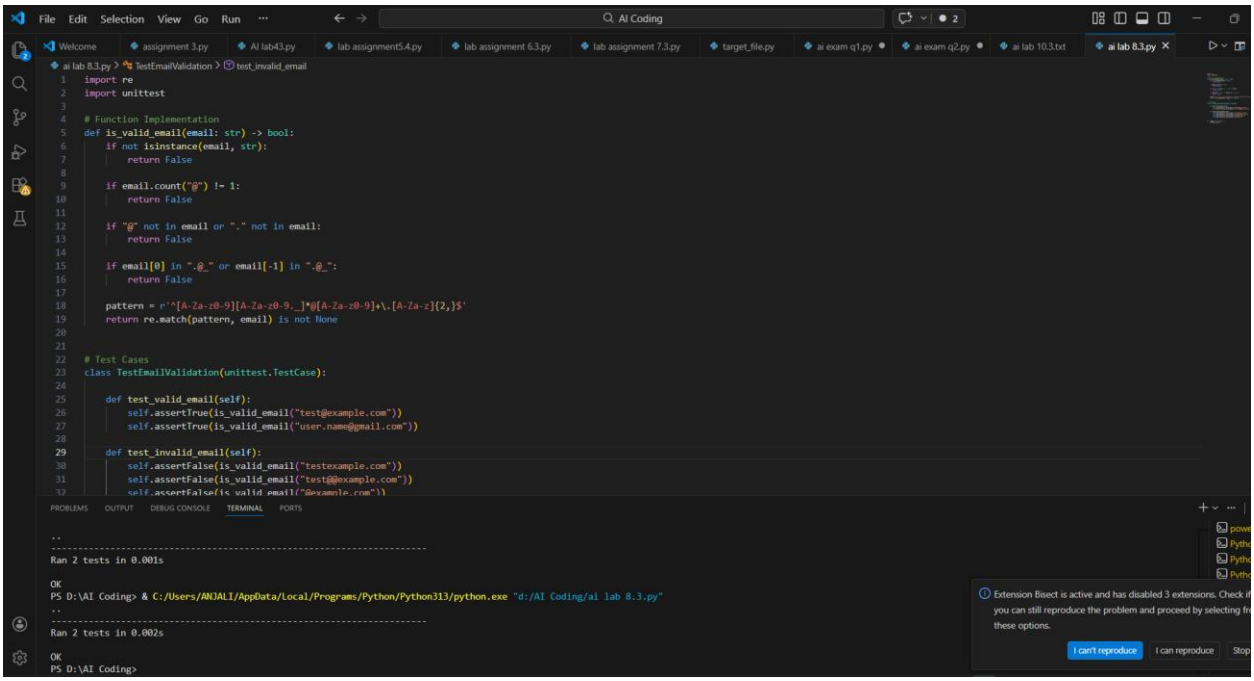
2303a51739

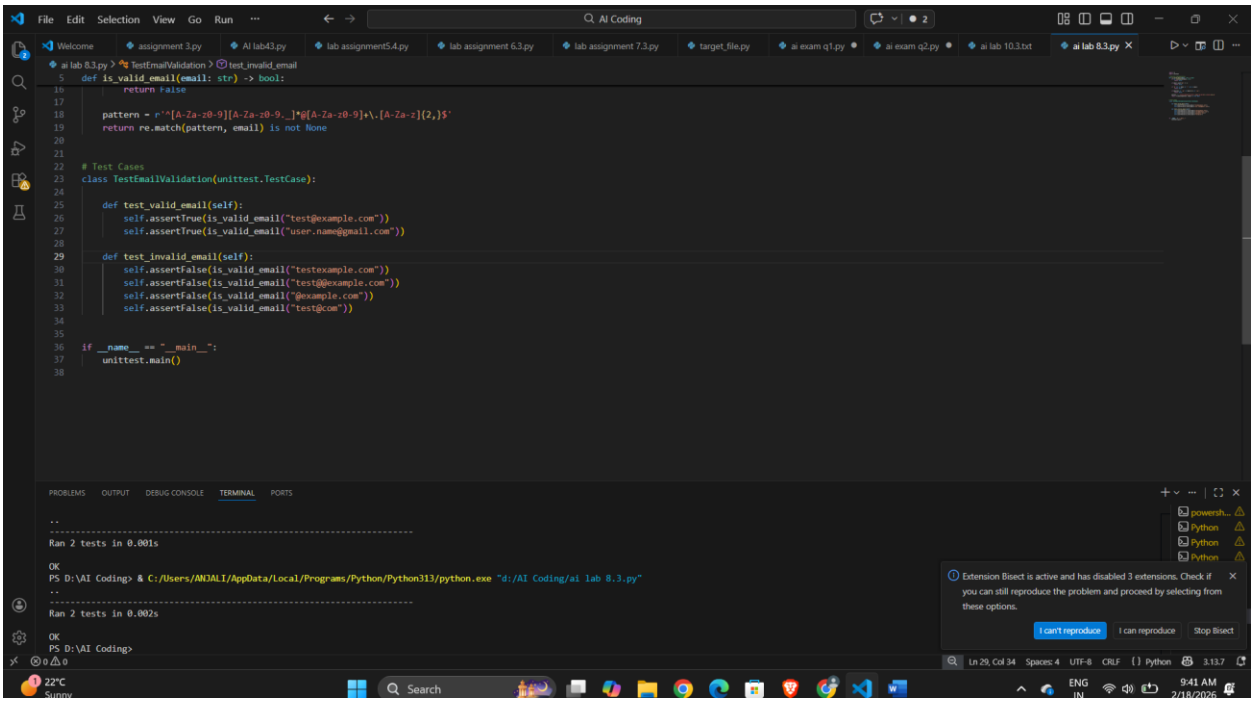
Lab assignment 8.3

Task 1: Email Validation using TDD

Scenario

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

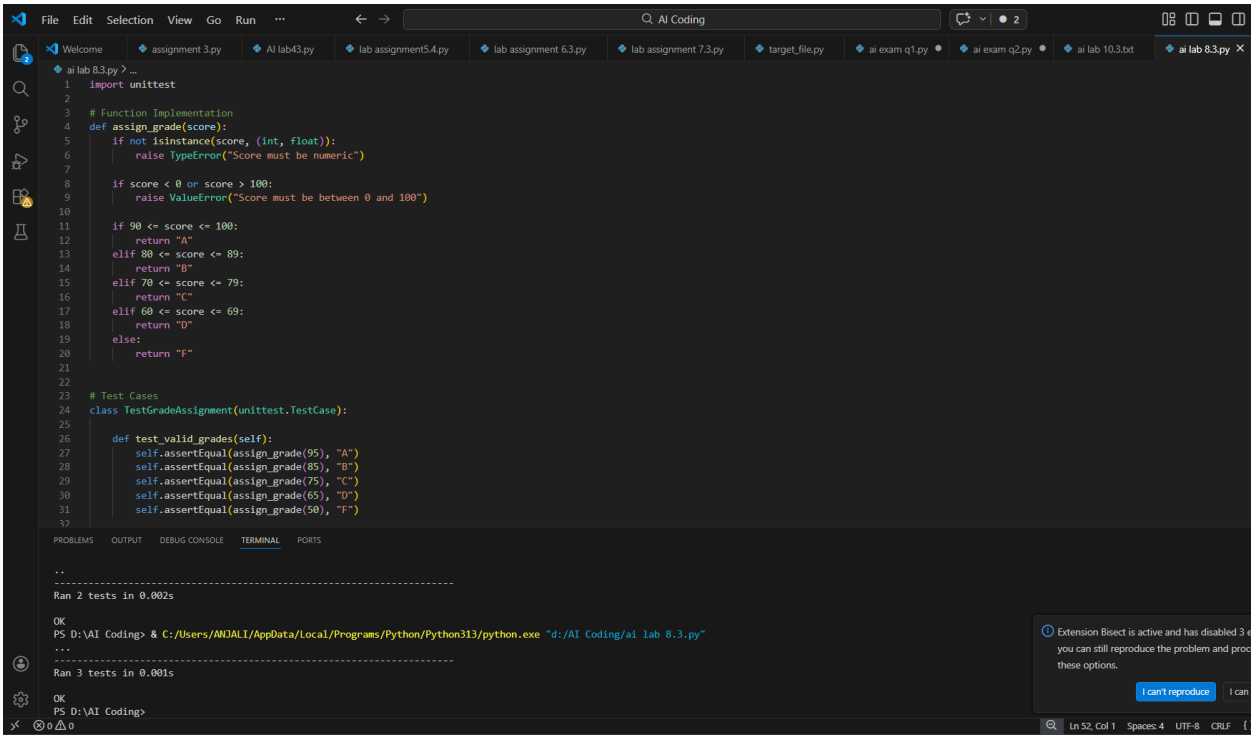




Task 2: Grade Assignment using Loops

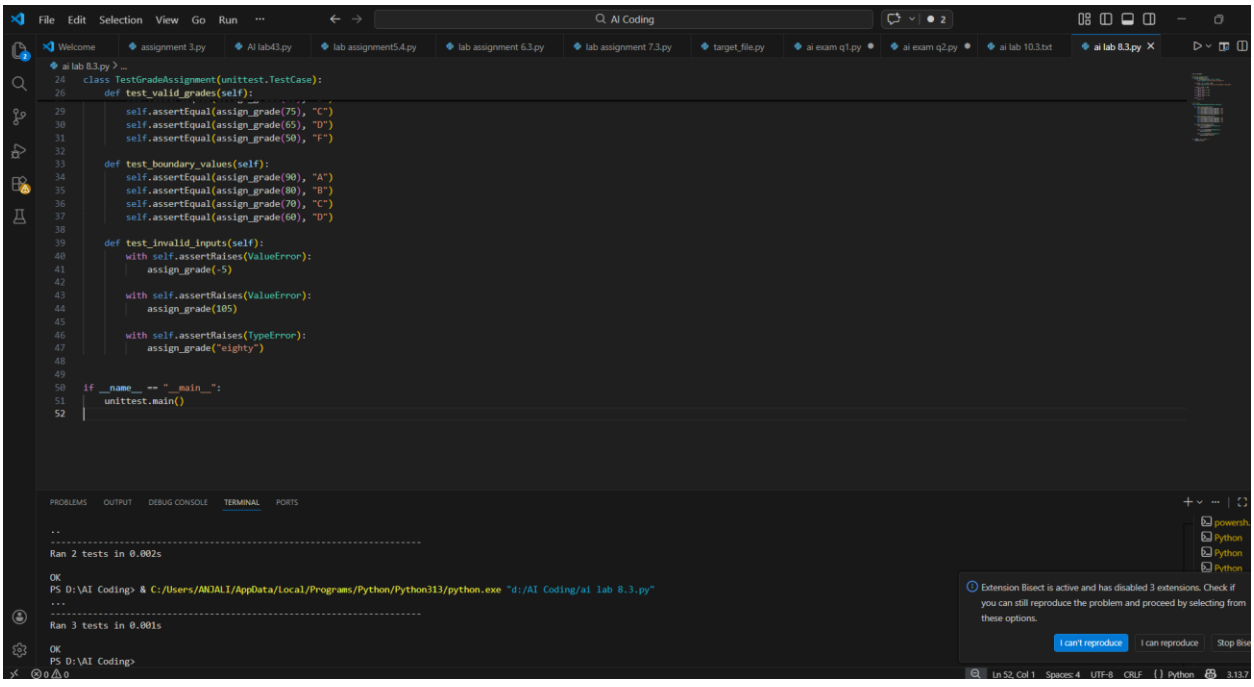
Scenario

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



```
import unittest

# Function Implementation
def assign_grade(score):
    if not isinstance(score, (int, float)):
        raise TypeError("Score must be numeric")

    if score < 0 or score > 100:
        raise ValueError("Score must be between 0 and 100")

    if 90 <= score <= 100:
        return "A"
    elif 80 <= score <= 89:
        return "B"
    elif 70 <= score <= 79:
        return "C"
    elif 60 <= score <= 69:
        return "D"
    else:
        return "F"


# Test Cases
class TestGradeAssignment(unittest.TestCase):

    def test_valid_grades(self):
        self.assertEqual(assign_grade(95), "A")
        self.assertEqual(assign_grade(85), "B")
        self.assertEqual(assign_grade(75), "C")
        self.assertEqual(assign_grade(65), "D")
        self.assertEqual(assign_grade(50), "F")
```

```
..
----------------------------------------------------------------------
Ran 2 tests in 0.002s

OK
PS D:\AI Coding> & C:/Users/ANJALI/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>
```



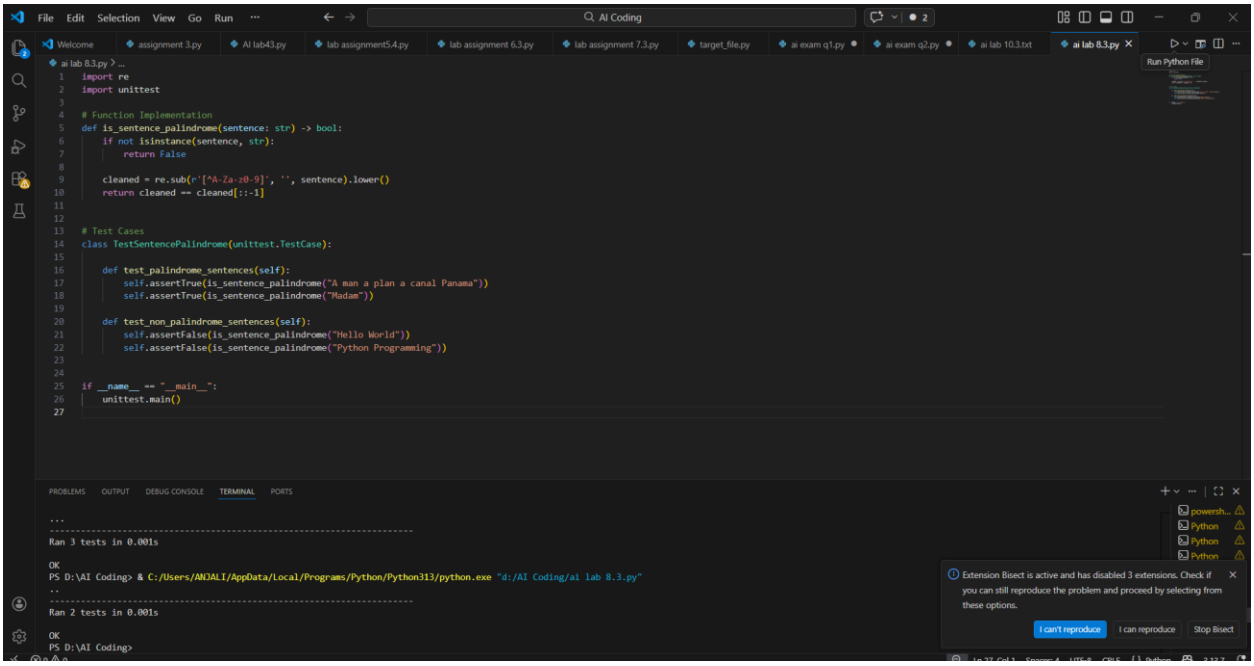
```
class TestGradeAssignment(unittest.TestCase):
    def test_valid_grades(self):
        self.assertEqual(assign_grade(75), "C")
        self.assertEqual(assign_grade(65), "D")
        self.assertEqual(assign_grade(50), "F")

    def test_boundary_values(self):
        self.assertEqual(assign_grade(90), "A")
        self.assertEqual(assign_grade(80), "B")
        self.assertEqual(assign_grade(70), "C")
        self.assertEqual(assign_grade(60), "D")

    def test_invalid_inputs(self):
        with self.assertRaises(ValueError):
            assign_grade(-5)

        with self.assertRaises(ValueError):
            assign_grade(105)

        with self.assertRaises(TypeError):
            assign_grade("eighty")


if __name__ == "__main__":
    unittest.main()
```

```
..
----------------------------------------------------------------------
Ran 2 tests in 0.002s

OK
PS D:\AI Coding> & C:/Users/ANJALI/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>
```

Task 3: Sentence Palindrome Checker

Scenario

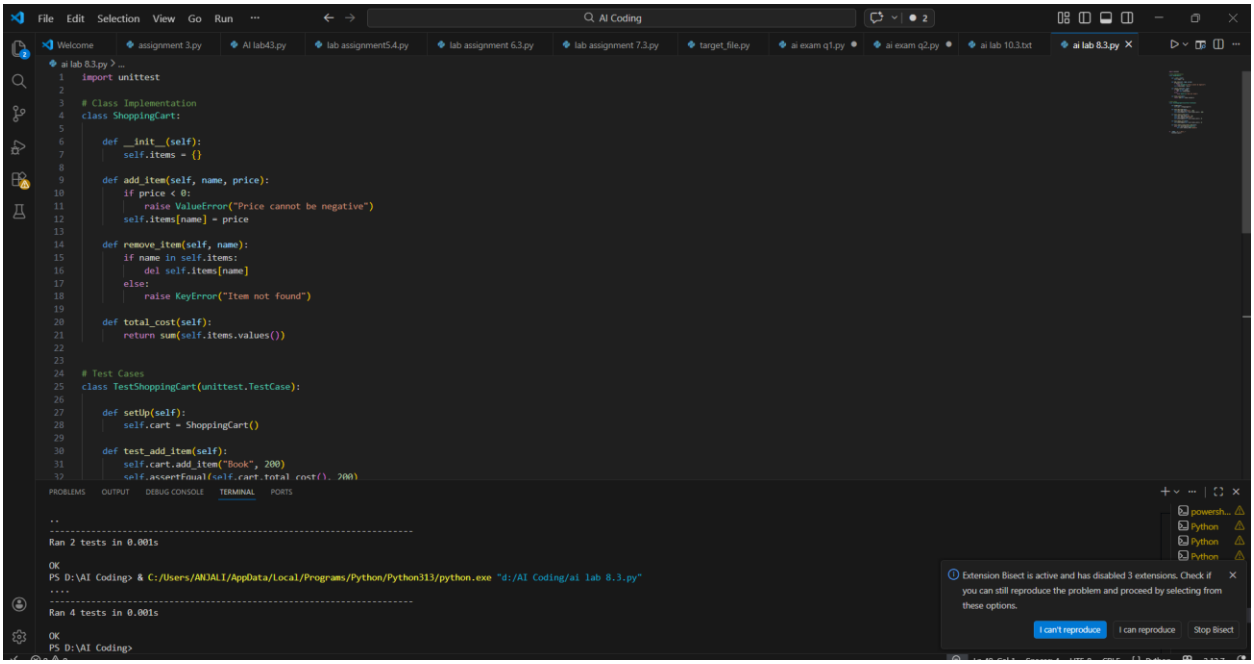
You are developing a text-processing utility to analyze sentences.



Task 4: ShoppingCart Class

Scenario

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

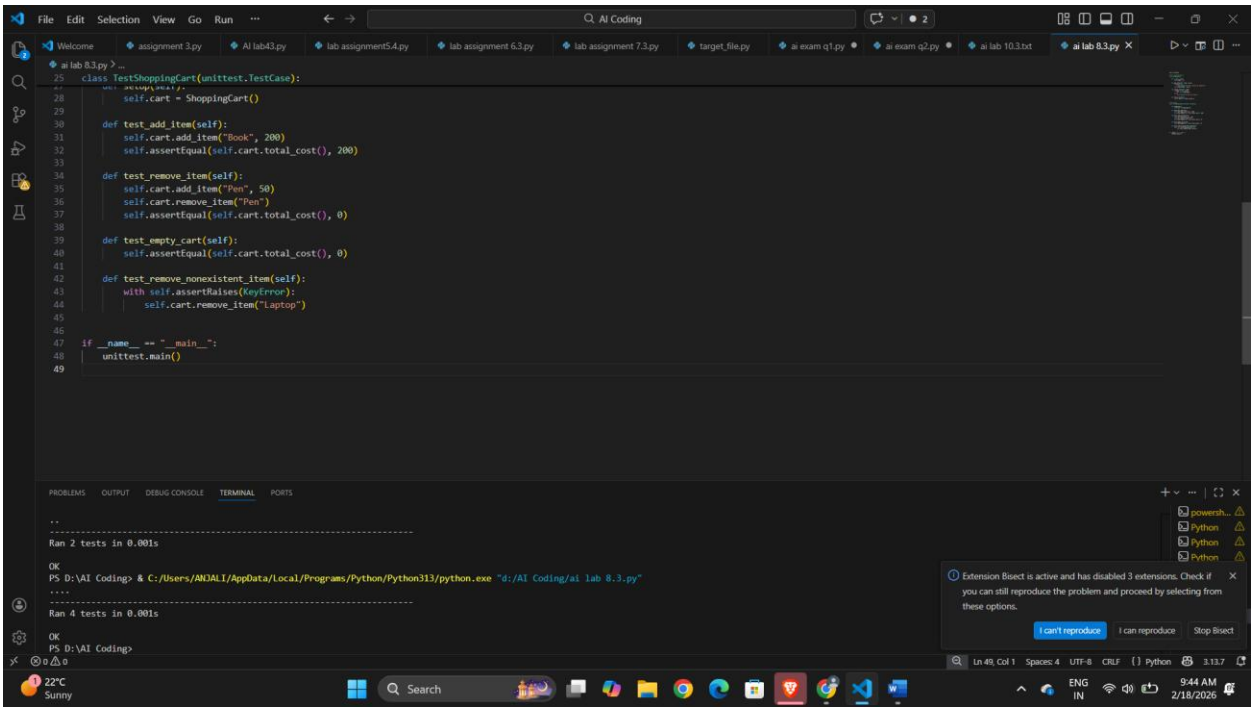


```
import unittest

# Class Implementation
class ShoppingCart:

    def __init__(self):
        self.items = {}

    def add_item(self, name, price):
        if price < 0:
            raise ValueError("Price cannot be negative")
        self.items[name] = price

    def remove_item(self, name):
        if name in self.items:
            del self.items[name]
        else:
            raise KeyError("Item not found")

    def total_cost(self):
        return sum(self.items.values())


# Test Cases
class TestShoppingCart(unittest.TestCase):

    def setUp(self):
        self.cart = ShoppingCart()

    def test_add_item(self):
        self.cart.add_item("Book", 200)
        self.assertEqual(self.cart.total_cost(), 200)
```

```
..
----------------------------------------------------------------------
Ran 2 tests in 0.001s

OK
PS D:\AI Coding> & C:/Users/ANJALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/ai lab 8.3.py"
....
----------------------------------------------------------------------
Ran 4 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



```
class TestShoppingCart(unittest.TestCase):
    def setUp(self):
        self.cart = ShoppingCart()

    def test_add_item(self):
        self.cart.add_item("Book", 200)
        self.assertEqual(self.cart.total_cost(), 200)

    def test_remove_item(self):
        self.cart.add_item("Pen", 50)
        self.cart.remove_item("Pen")
        self.assertEqual(self.cart.total_cost(), 0)

    def test_empty_cart(self):
        self.assertEqual(self.cart.total_cost(), 0)

    def test_remove_nonexistent_item(self):
        with self.assertRaises(KeyError):
            self.cart.remove_item("Laptop")


if __name__ == "__main__":
    unittest.main()
```

```
..
----------------------------------------------------------------------
Ran 2 tests in 0.001s

OK
PS D:\AI Coding> & C:/Users/ANJALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/ai lab 8.3.py"
....
----------------------------------------------------------------------
Ran 4 tests in 0.001s

OK
PS D:\AI Coding>
```

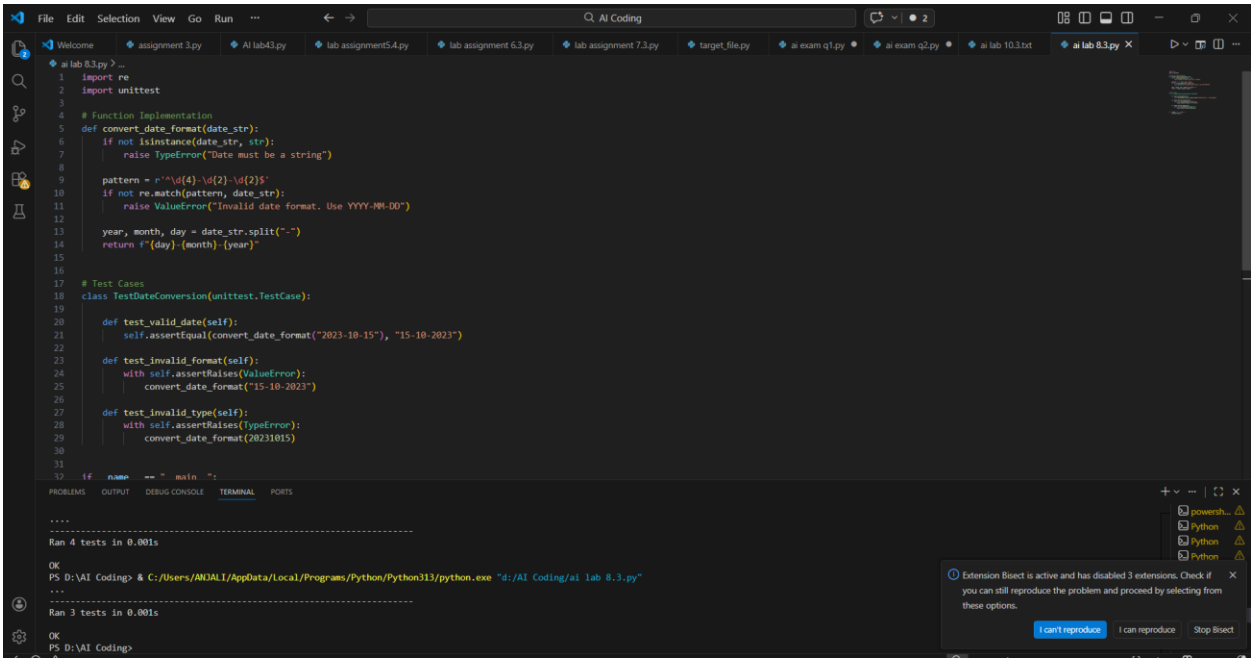
Extension Bisect is active and has disabled 3 extensions. Check if you can still reproduce the problem and proceed by selecting from these options.

I can't reproduce | I can reproduce | Stop Bisect

Task 5: Date Format Conversion

Scenario

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

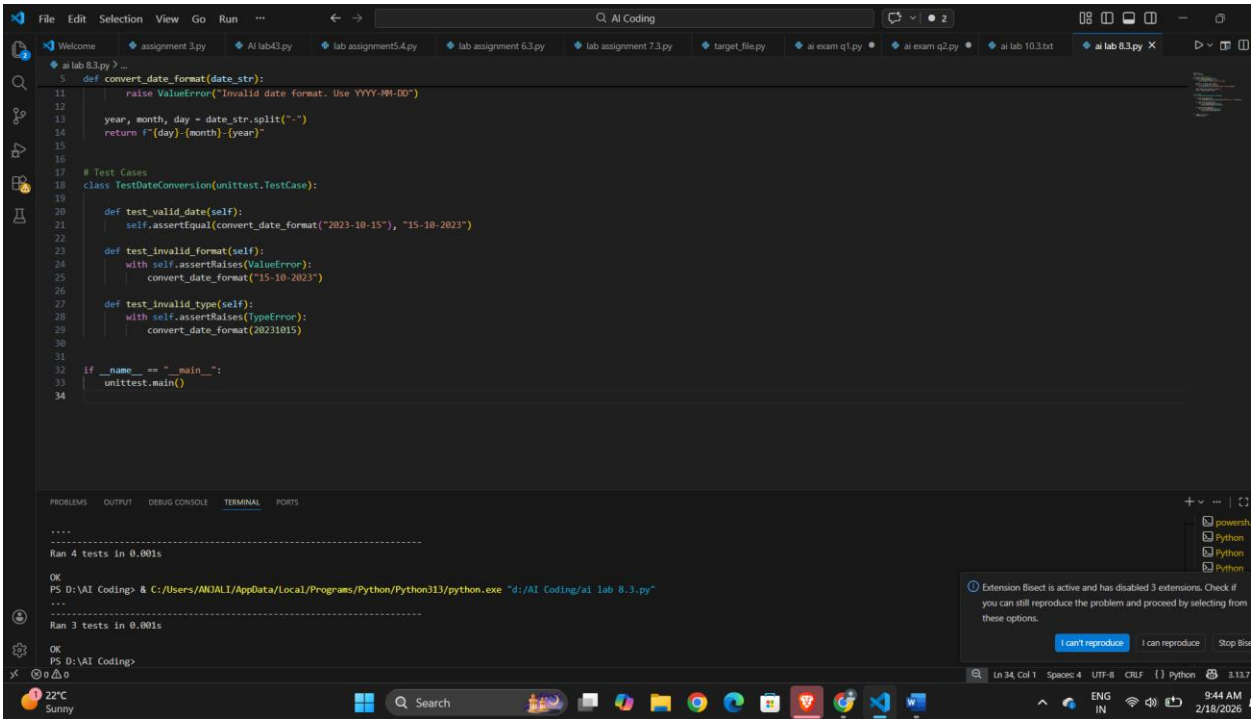


```
import re
import unittest

# Function Implementation
def convert_date_format(date_str):
    if not isinstance(date_str, str):
        raise TypeError("Date must be a string")

    pattern = r'^\d{4}-\d{2}-\d{2}$'
    if not re.match(pattern, date_str):
        raise ValueError("Invalid date format. Use YYYY-MM-DD")

    year, month, day = date_str.split("-")
    return f"{day}-{month}-{year}"


# Test Cases
class TestDateConversion(unittest.TestCase):

    def test_valid_date(self):
        self.assertEqual(convert_date_format("2023-10-15"), "15-10-2023")

    def test_invalid_format(self):
        with self.assertRaises(ValueError):
            convert_date_format("15-10-2023")

    def test_invalid_type(self):
        with self.assertRaises(TypeError):
            convert_date_format(20231015)


if __name__ == "__main__":
```

```
....
----------------------------------------------------------------------
Ran 4 tests in 0.001s

OK
PS D:\AI Coding> & C:/Users/ANJALI/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



```
def convert_date_format(date_str):
        raise ValueError("Invalid date format. Use YYYY-MM-DD")

    year, month, day = date_str.split("-")
    return f"{day}-{month}-{year}"


# Test Cases
class TestDateConversion(unittest.TestCase):

    def test_valid_date(self):
        self.assertEqual(convert_date_format("2023-10-15"), "15-10-2023")

    def test_invalid_format(self):
        with self.assertRaises(ValueError):
            convert_date_format("15-10-2023")

    def test_invalid_type(self):
        with self.assertRaises(TypeError):
            convert_date_format(20231015)


if __name__ == "__main__":
    unittest.main()
```

```
....
----------------------------------------------------------------------
Ran 4 tests in 0.001s

OK
PS D:\AI Coding> & C:/Users/ANJALI/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 Bis