LAB TEST-2 (AI)

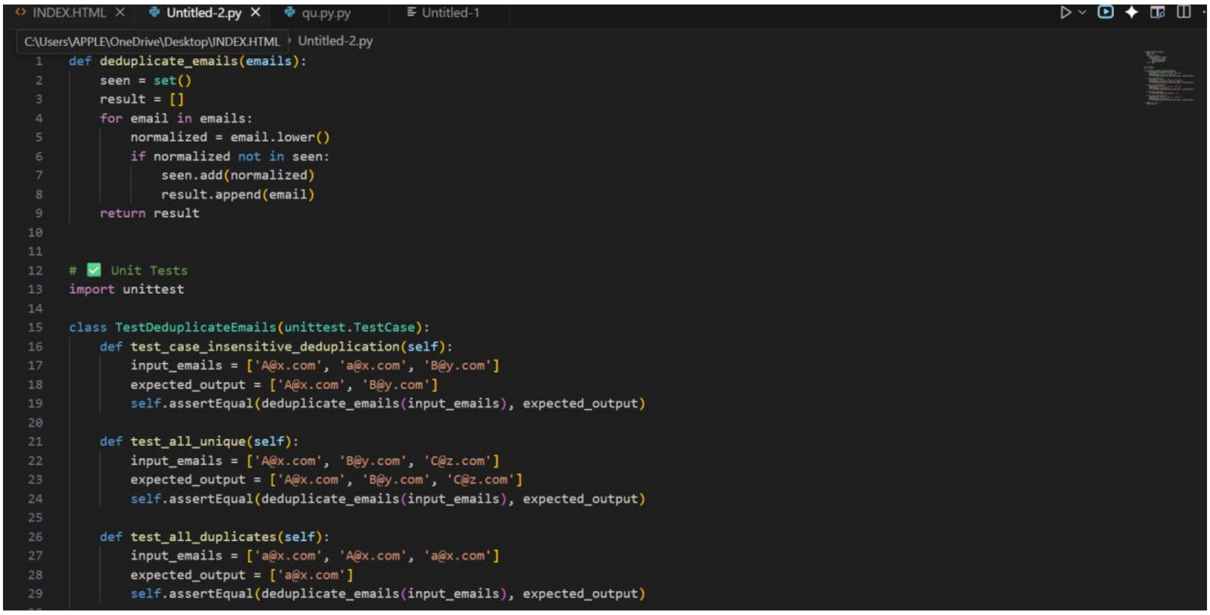
S.MOHANTH SIDDARTHA

2403A52047

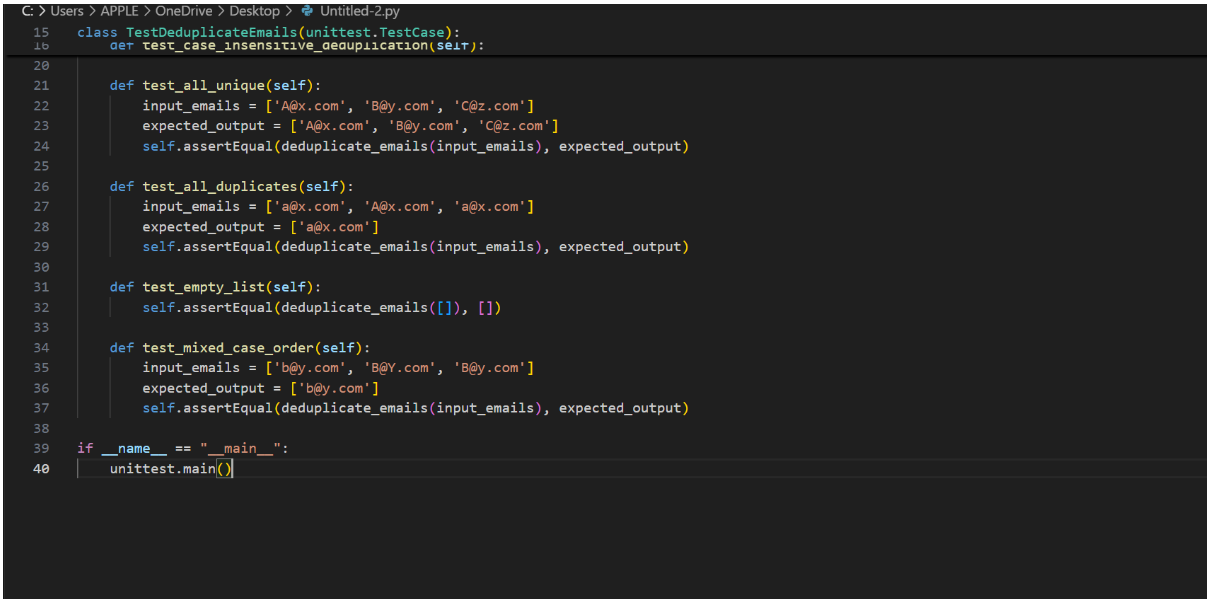
BATCH:03

C.1 — [S14C1] Debug de-duplication (case-insensitive)

CODE:



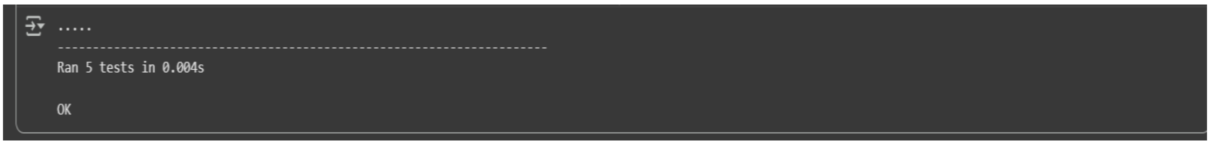
```python
def deduplicate_emails(emails):
    seen = set()
    result = []
    for email in emails:
        normalized = email.lower()
        if normalized not in seen:
            seen.add(normalized)
            result.append(email)
    return result


# ✅ Unit Tests
import unittest

class TestDeduplicateEmails(unittest.TestCase):
    def test_case_insensitive_deduplication(self):
        input_emails = ['A@x.com', 'a@x.com', 'B@y.com']
        expected_output = ['A@x.com', 'B@y.com']
        self.assertEqual(deduplicate_emails(input_emails), expected_output)

    def test_all_unique(self):
        input_emails = ['A@x.com', 'B@y.com', 'C@z.com']
        expected_output = ['A@x.com', 'B@y.com', 'C@z.com']
        self.assertEqual(deduplicate_emails(input_emails), expected_output)

    def test_all_duplicates(self):
        input_emails = ['a@x.com', 'A@x.com', 'a@x.com']
        expected_output = ['a@x.com']
        self.assertEqual(deduplicate_emails(input_emails), expected_output)
```



C: > Users > APPLE > OneDrive > Desktop > Untitled-2.py

```python
class TestDeduplicateEmails(unittest.TestCase):
    def test_case_insensitive_deduplication(self):

    def test_all_unique(self):
        input_emails = ['A@x.com', 'B@y.com', 'C@z.com']
        expected_output = ['A@x.com', 'B@y.com', 'C@z.com']
        self.assertEqual(deduplicate_emails(input_emails), expected_output)

    def test_all_duplicates(self):
        input_emails = ['a@x.com', 'A@x.com', 'a@x.com']
        expected_output = ['a@x.com']
        self.assertEqual(deduplicate_emails(input_emails), expected_output)

    def test_empty_list(self):
        self.assertEqual(deduplicate_emails([]), [])

    def test_mixed_case_order(self):
        input_emails = ['b@y.com', 'B@Y.com', 'B@y.com']
        expected_output = ['b@y.com']
        self.assertEqual(deduplicate_emails(input_emails), expected_output)

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

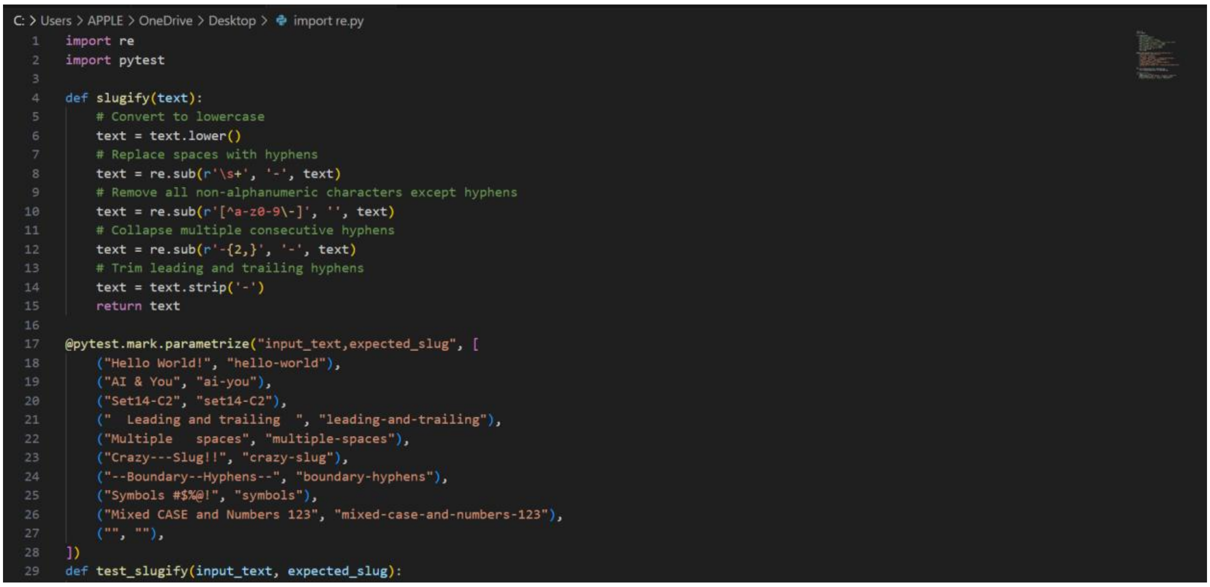
OUTPUT:



```
.....
----------------------------------------------------------------------
Ran 5 tests in 0.004s

OK
```

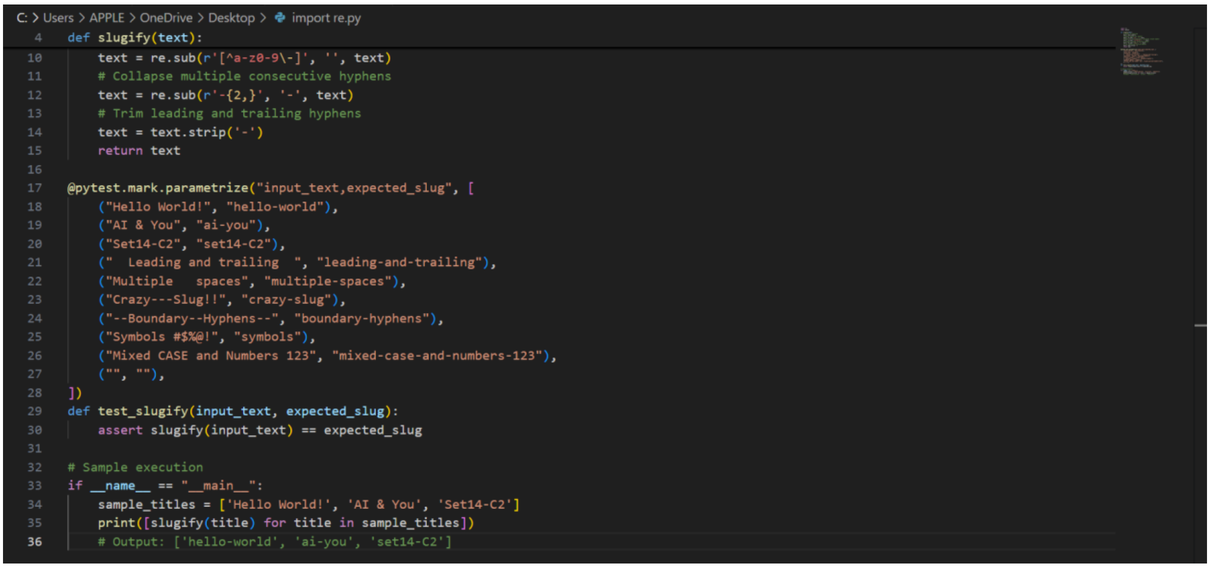
C.2 — [S14C2] TDD: slugify titles

CODE:



C: > Users > APPLE > OneDrive > Desktop > import re.py

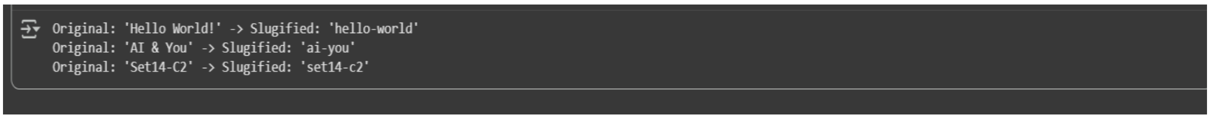
```python
import re
import pytest

def slugify(text):
    # Convert to lowercase
    text = text.lower()
    # Replace spaces with hyphens
    text = re.sub(r'\s+', '-', text)
    # Remove all non-alphanumeric characters except hyphens
    text = re.sub(r'[^a-z0-9\-]', '', text)
    # Collapse multiple consecutive hyphens
    text = re.sub(r'-{2,}', '-', text)
    # Trim leading and trailing hyphens
    text = text.strip('-')
    return text

@pytest.mark.parametrize("input_text,expected_slug", [
    ("Hello World!", "hello-world"),
    ("AI & You", "ai-you"),
    ("Set14-C2", "set14-C2"),
    ("  Leading and trailing  ", "leading-and-trailing"),
    ("Multiple   spaces", "multiple-spaces"),
    ("Crazy---Slug!!", "crazy-slug"),
    ("--Boundary--Hyphens--", "boundary-hyphens"),
    ("Symbols #$%@!", "symbols"),
    ("Mixed CASE and Numbers 123", "mixed-case-and-numbers-123"),
    ("", ""),
])
def test_slugify(input_text, expected_slug):
```



C: > Users > APPLE > OneDrive > Desktop > import re.py

```python
def slugify(text):
    text = re.sub(r'[^a-z0-9\-]', '', text)
    # Collapse multiple consecutive hyphens
    text = re.sub(r'-{2,}', '-', text)
    # Trim leading and trailing hyphens
    text = text.strip('-')
    return text

@pytest.mark.parametrize("input_text,expected_slug", [
    ("Hello World!", "hello-world"),
    ("AI & You", "ai-you"),
    ("Set14-C2", "set14-C2"),
    ("  Leading and trailing  ", "leading-and-trailing"),
    ("Multiple   spaces", "multiple-spaces"),
    ("Crazy---Slug!!", "crazy-slug"),
    ("--Boundary--Hyphens--", "boundary-hyphens"),
    ("Symbols #$%@!", "symbols"),
    ("Mixed CASE and Numbers 123", "mixed-case-and-numbers-123"),
    ("", ""),
])
def test_slugify(input_text, expected_slug):
    assert slugify(input_text) == expected_slug

# Sample execution
if __name__ == "__main__":
    sample_titles = ['Hello World!', 'AI & You', 'Set14-C2']
    print([slugify(title) for title in sample_titles])
    # Output: ['hello-world', 'ai-you', 'set14-C2']
```

OUTPUT:



```
Original: 'Hello World!' -> Slugified: 'hello-world'
Original: 'AI & You' -> Slugified: 'ai-you'
Original: 'Set14-C2' -> Slugified: 'set14-c2'
```