

LAB EXAM-2

NAME: G. Bala Varshitha

HT.NO: 2403A52050

BATCH: AIB03

Subgroup-C

Task -1

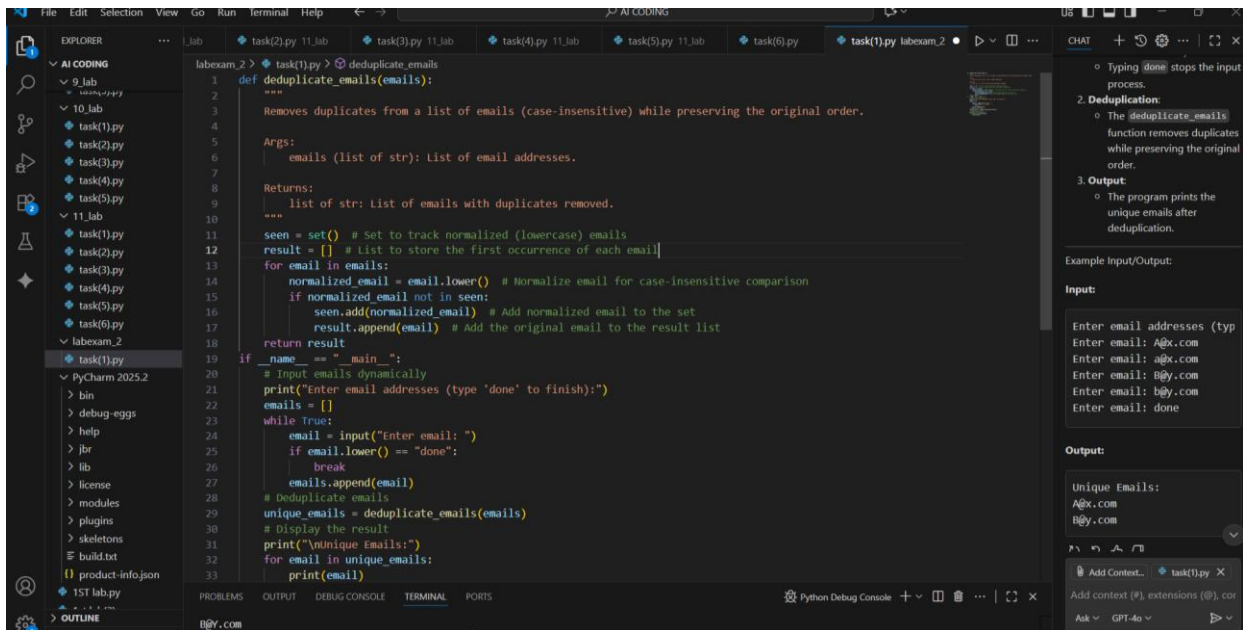
Prompt:

given a list of customer emails from a telecom system.

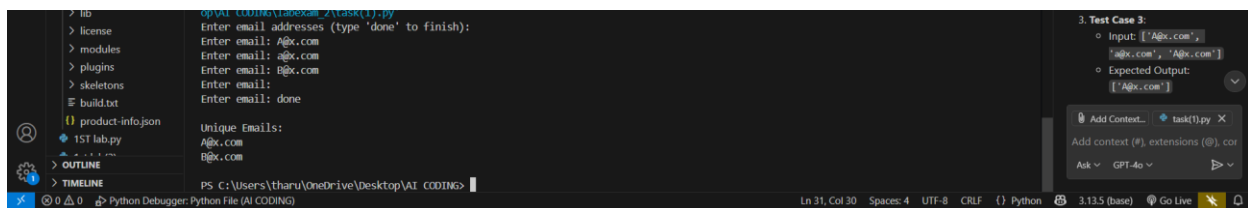
Sometimes the same email appears more than once, but with different letter case (for example, A@x.com and [a@x.com](#)).

Now Removes duplicates case-insensitively (treat A@x.com and a@x.com as the same). Keeps the first occurrence of each email in the list. Preserve the original order and the original case of the first appearance. I want to give input dynamically

Code:



OP:



Observation: Emails are compared in lowercase so different letter cases count as the same, while the original text is kept in the result. Only the first time an address shows up is kept, and everything stays in the original order. A quick check is used to see if an address was already recorded.

Task-2

Prompt: write a function `slugify(text)` that converts all letters to lowercase, replace spaces with `-`, removes all characters except

letters, numbers, and -, collapses multiple - into a single -, removes any leading or trailing -.

Code:

```
11.2 > test_slugify.py > test_slugify
1 import re
2 import pytest
3 def slugify(text: str) -> str:
4     """
5     Converts a given text into an SEO-friendly slug.
6
7     Args:
8         text (str): The input text to be converted.
9
10    Returns:
11        str: The SEO-friendly slug.
12    """
13    # Convert to lowercase
14    text = text.lower()
15    # Remove non-alphanumeric characters except hyphens and spaces
16    text = re.sub(r'^a-z0-9\s-]', '', text)
17    # Replace spaces with hyphens
18    text = re.sub(r'\s+', '-', text)
19    # Collapse multiple hyphens into a single hyphen
20    text = re.sub(r'-+', '-', text)
21    # Trim leading and trailing hyphens
22    text = text.strip('-')
23    return text
24 @pytest.mark.parametrize("input_text, expected_slug", [
25     # Basic cases
26     ("Hello World!", "hello-world"),
27     ("AI & You", "ai-you"),
28     ("Set14-C2", "set14-c2"),
29     # Edge cases
30     (" Leading and trailing spaces ", "leading-and-trailing-spaces"),
31     ("Multiple Spaces", "multiple-spaces"),
32     ("---Boundary---Hyphens---", "boundary-hyphens"),
33     ("Special!@#%&*()Characters", "specialcharacters"),
34     ("Already-SEO-friendly", "already-seo-friendly"),
35     ("", ""), # Empty string
36     ("-", "-"), # Single hyphen
37     ("A", "a"), # Single character
38 ])
39 def test_slugify(input_text, expected_slug):
40     assert slugify(input_text) == expected_slug
```

This will print each element on a new line, matching your expected output.

Create tests and implement bump_version(name) that adds or increments _vNN with zero-padding.

```
import re

def bump_version(name):
    """
    Adds or increments a '_vNN' version suffix.
    If '_vNN' exists, increment NN; otherwise,
    :param name: The input string.
    :return: The string with bumped version.
    """
    match = re.search(r'_v(\d{2})$', name)
    if match:
        num = int(match.group(1)) + 1
        return f"{name[:-3]}_v{num:02d}"
    else:
        return f"{name}_v01"

# Tests
assert bump_version("file") == "file_v01"
assert bump_version("report_v01") == "report_v02"
assert bump_version("data_v09") == "data_v10"
assert bump_version("image_v99") == "image_v01"
print("All bump_version tests passed!")
```

Add Context... test_slugify.py X

Add context (#), extensions (@), commands (/)

Ask GPT-4.1

OP:

```
PS C:\Users\Admin\OneDrive\Desktop\AI Assisted coding assignments\AI Assisted Coding> & "C:/Users/Admin/OneDrive/Desktop/AI Assisted coding assignments/AI Assisted Coding/.venv/Scripts/Activate.ps1"
(.venv) PS C:\Users\Admin\OneDrive\Desktop\AI Assisted coding assignments\AI Assisted Coding> pytest -s
===== test session starts =====
platform win32 -- Python 3.11.9, pytest-8.4.2, pluggy-1.6.0
rootdir: C:\Users\Admin\OneDrive\Desktop\AI Assisted coding assignments\AI Assisted Coding
collected 11 items

11.2\test_slugify.py .....

===== 11 passed in 0.03s =====
(.venv) PS C:\Users\Admin\OneDrive\Desktop\AI Assisted coding assignments\AI Assisted Coding>
```

Observation:

This code turns any piece of text into a simple web-friendly version by making everything small letters and swapping spaces with dashes. It takes out special symbols so only plain letters, numbers, and dashes remain. Even if the input has extra spaces or lots of dashes together, it removes them so just single dashes are used. The checks make sure the original order is kept and test whether different examples work as intended, like handling empty text, single letters, or titles with unusual characters