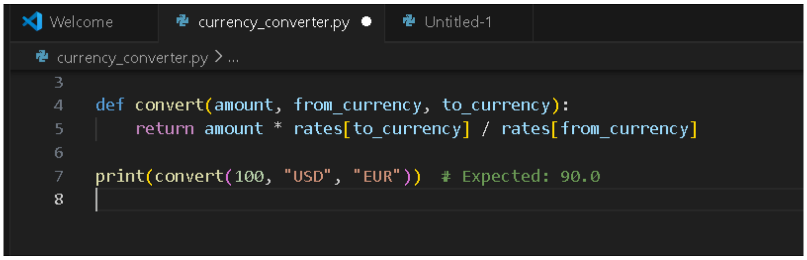
AI ASSISTED CODING TEST-1

HT.NO# 2403A51276

TASK-1:

1. Create a Python function that converts an amount from one currency to another using exchange rates stored in a dictionary. Use GitHub Copilot along with VS Code. Use Few shot prompting.

Initial code given:



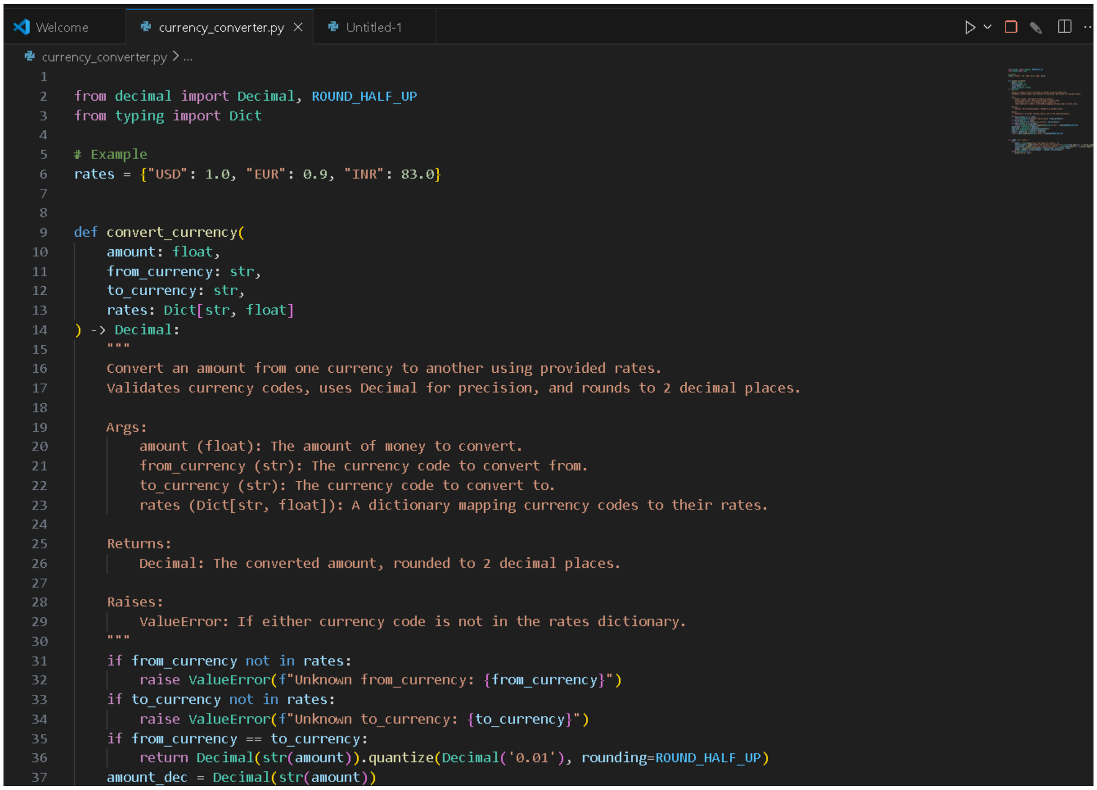
```python

def convert(amount, from_currency, to_currency):
    return amount * rates[to_currency] / rates[from_currency]

print(convert(100, "USD", "EUR"))  # Expected: 90.0

```

Short Prompt given to github copilot next:

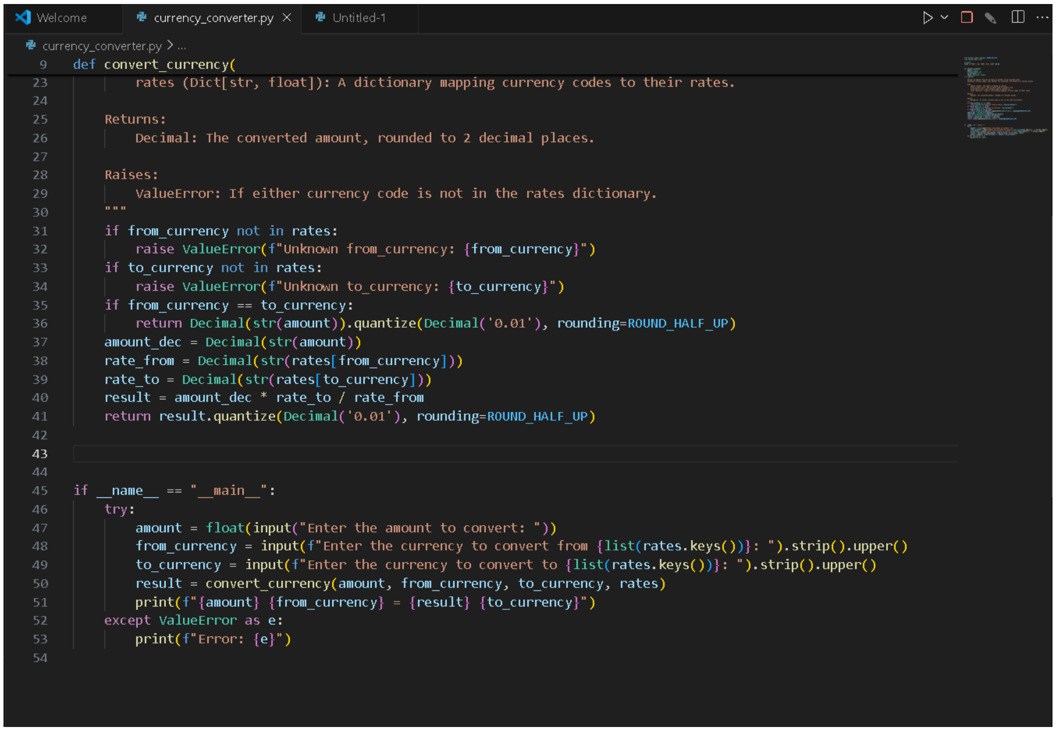
Based on the example function in my file, generate a more general function called convert_currency(amount, from_currency, to_currency, rates).
Requirements:

- Validate that both currencies exist in the rates dictionary.
- If the currencies are the same, just return the amount.
- Use Python’s Decimal for precision.
- Round results to 2 decimal places.
- Add type hints and a clear docstring.

Refined code from Copilot after the prompt:

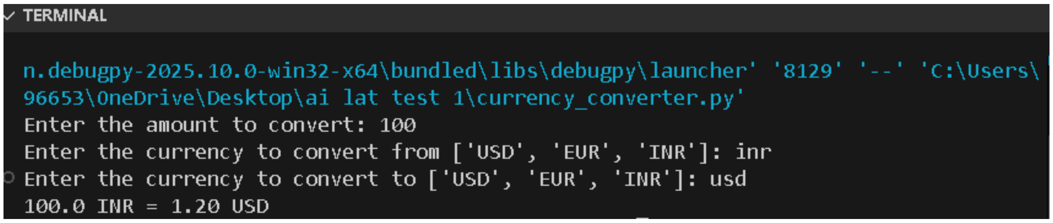


```python

from decimal import Decimal, ROUND_HALF_UP
from typing import Dict

# Example
rates = {"USD": 1.0, "EUR": 0.9, "INR": 83.0}


def convert_currency(
    amount: float,
    from_currency: str,
    to_currency: str,
    rates: Dict[str, float]
) -> Decimal:
    """
    Convert an amount from one currency to another using provided rates.
    Validates currency codes, uses Decimal for precision, and rounds to 2 decimal places.

    Args:
        amount (float): The amount of money to convert.
        from_currency (str): The currency code to convert from.
        to_currency (str): The currency code to convert to.
        rates (Dict[str, float]): A dictionary mapping currency codes to their rates.

    Returns:
        Decimal: The converted amount, rounded to 2 decimal places.

    Raises:
        ValueError: If either currency code is not in the rates dictionary.
    """
    if from_currency not in rates:
        raise ValueError(f"Unknown from_currency: {from_currency}")
    if to_currency not in rates:
        raise ValueError(f"Unknown to_currency: {to_currency}")
    if from_currency == to_currency:
        return Decimal(str(amount)).quantize(Decimal('0.01'), rounding=ROUND_HALF_UP)
    amount_dec = Decimal(str(amount))
```



```python
def convert_currency(
        rates (Dict[str, float]): A dictionary mapping currency codes to their rates.

    Returns:
        Decimal: The converted amount, rounded to 2 decimal places.

    Raises:
        ValueError: If either currency code is not in the rates dictionary.
    """
    if from_currency not in rates:
        raise ValueError(f"Unknown from_currency: {from_currency}")
    if to_currency not in rates:
        raise ValueError(f"Unknown to_currency: {to_currency}")
    if from_currency == to_currency:
        return Decimal(str(amount)).quantize(Decimal('0.01'), rounding=ROUND_HALF_UP)
    amount_dec = Decimal(str(amount))
    rate_from = Decimal(str(rates[from_currency]))
    rate_to = Decimal(str(rates[to_currency]))
    result = amount_dec * rate_to / rate_from
    return result.quantize(Decimal('0.01'), rounding=ROUND_HALF_UP)



if __name__ == "__main__":
    try:
        amount = float(input("Enter the amount to convert: "))
        from_currency = input(f"Enter the currency to convert from {list(rates.keys())}: ").strip().upper()
        to_currency = input(f"Enter the currency to convert to {list(rates.keys())}: ").strip().upper()
        result = convert_currency(amount, from_currency, to_currency, rates)
        print(f"{amount} {from_currency} = {result} {to_currency}")
    except ValueError as e:
        print(f"Error: {e}")

```

Example Output:



```
n.debugpy-2025.10.0-win32-x64\bundled\libs\debugpy\launcher' '8129' '--' 'C:\Users\
96653\OneDrive\Desktop\ai lat test 1\currency_converter.py'
Enter the amount to convert: 100
Enter the currency to convert from ['USD', 'EUR', 'INR']: inr
Enter the currency to convert to ['USD', 'EUR', 'INR']: usd
100.0 INR = 1.20 USD
```

TASK-2:

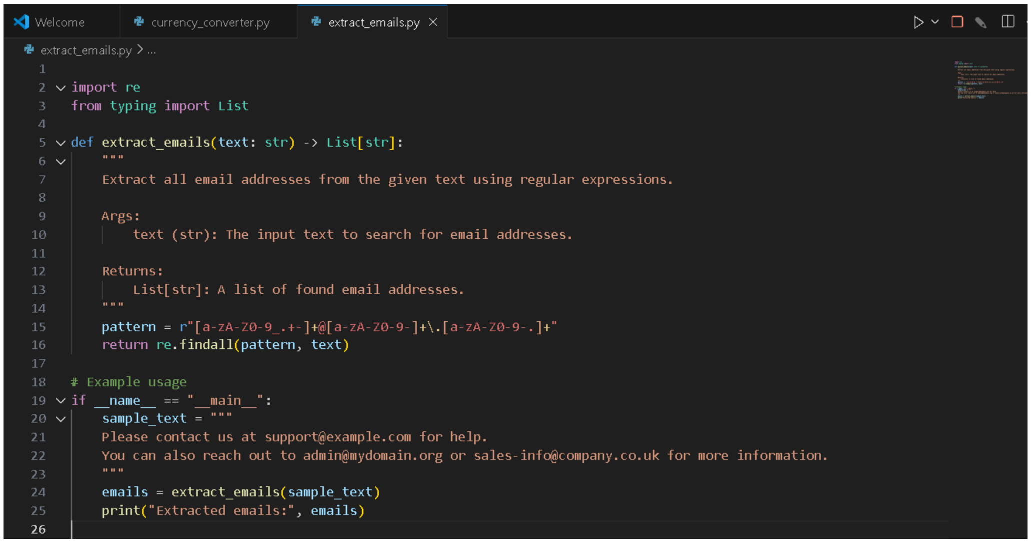
Write a Python program to extract all email addresses from a block of text using regular expressions. GitHub Copilot along with VS Code. Use zero shot prompting.

Prompt given to GitHub copilot:

Write a Python program that extracts all email addresses from a block of text using regular expressions.
Requirements:

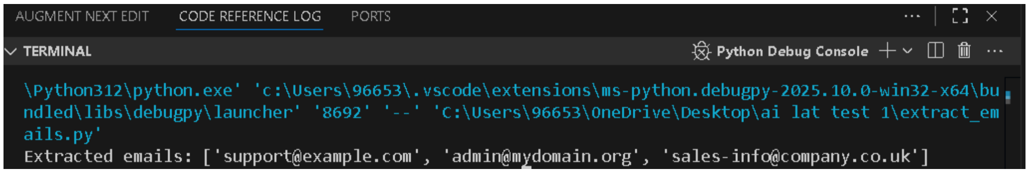
- Define a function extract_emails(text: str) -> list[str].
- Use the re module with a regex pattern for emails.
- Return a list of all found email addresses.
- Add an example usage with a sample text containing at least 3 emails and print the result.

Code from Copilot after the prompt:



```python

import re
from typing import List

def extract_emails(text: str) -> List[str]:
    """
    Extract all email addresses from the given text using regular expressions.

    Args:
        text (str): The input text to search for email addresses.

    Returns:
        List[str]: A list of found email addresses.
    """
    pattern = r"[a-zA-Z0-9_.+-]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-.]+"
    return re.findall(pattern, text)

# Example usage
if __name__ == "__main__":
    sample_text = """
    Please contact us at support@example.com for help.
    You can also reach out to admin@mydomain.org or sales-info@company.co.uk for more information.
    """
    emails = extract_emails(sample_text)
    print("Extracted emails:", emails)

```

Example Output:

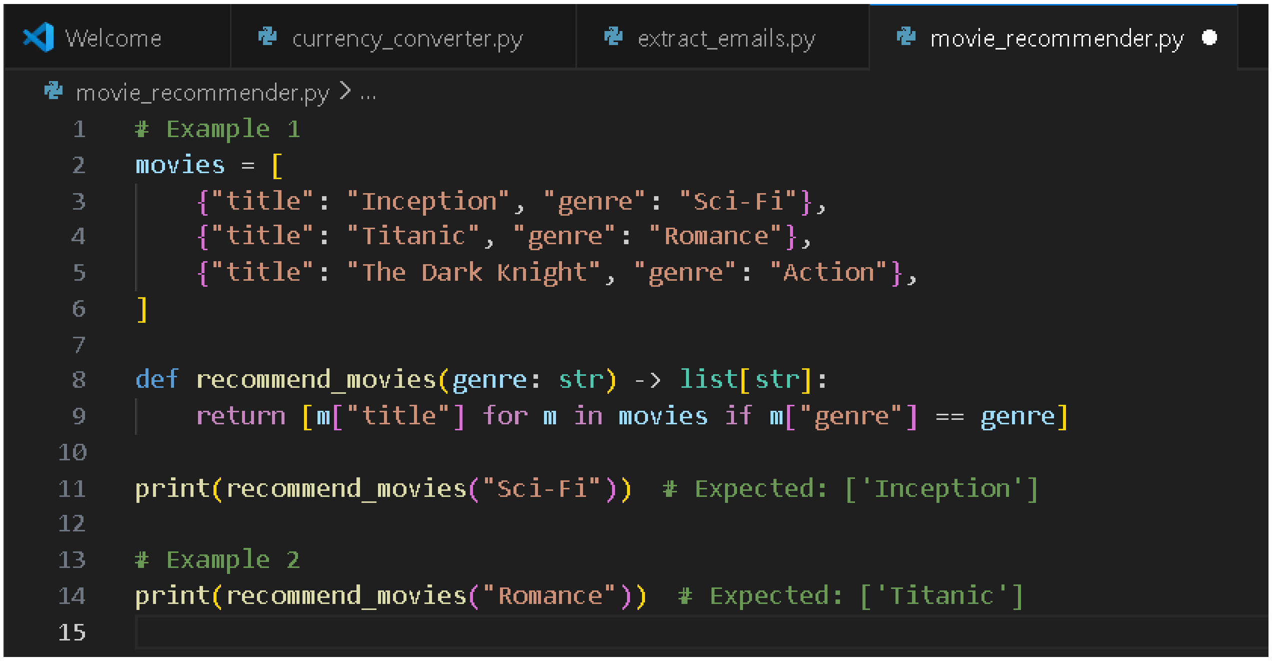


```
\Python312\python.exe' 'c:\Users\96653\.vscode\extensions\ms-python.debugpy-2025.10.0-win32-x64\bu
ndled\libs\debugpy\launcher' '8692' '--' 'C:\Users\96653\OneDrive\Desktop\ai lat test 1\extract_em
ails.py'
Extracted emails: ['support@example.com', 'admin@mydomain.org', 'sales-info@company.co.uk']
```

TASK-3:

Given a list of movies with their genres, write a Python function that recommends movies based on a user's preferred genre. Use the Cursor AI tool. Use few shot prompting.

Initial list of movies and code:



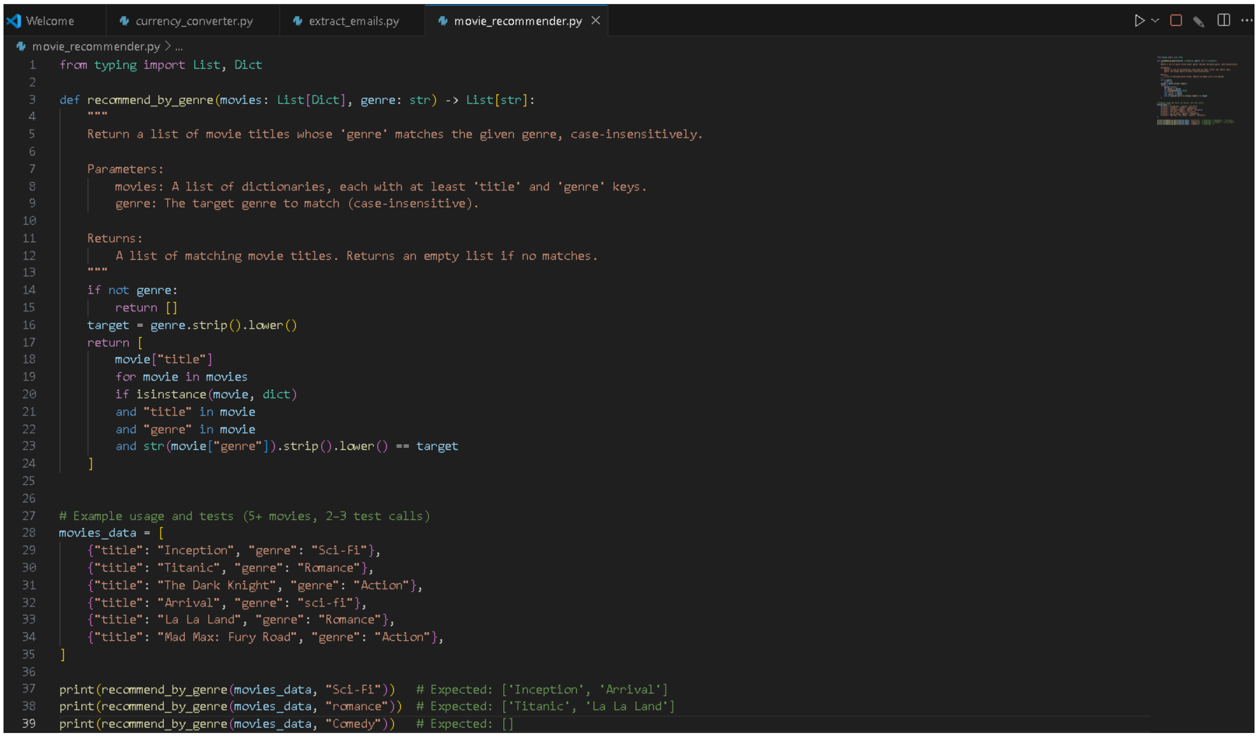
```python
# Example 1
movies = [
    {"title": "Inception", "genre": "Sci-Fi"},
    {"title": "Titanic", "genre": "Romance"},
    {"title": "The Dark Knight", "genre": "Action"},
]

def recommend_movies(genre: str) -> list[str]:
    return [m["title"] for m in movies if m["genre"] == genre]

print(recommend_movies("Sci-Fi"))  # Expected: ['Inception']

# Example 2
print(recommend_movies("Romance"))  # Expected: ['Titanic']
```

Prompt given to Cursor AI:

Based on the examples above, generate a more general movie recommendation function.
Requirements:

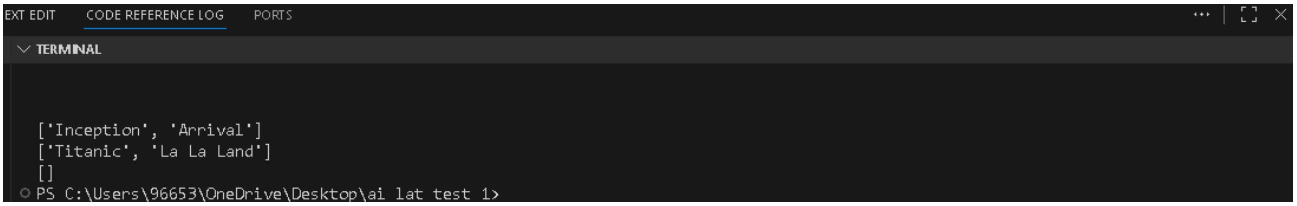
- Name the function recommend_by_genre(movies: list[dict], genre: str) -> list[str].
- The function should filter any list of movies, not just the hardcoded one.
- Case-insensitive matching of genre.
- If no movies match, return an empty list.
- Add docstring and type hints.
- Include an example usage with at least 5 movies and 2–3 test calls.

Refined code from Cursor AI after the prompt:



```python
from typing import List, Dict

def recommend_by_genre(movies: List[Dict], genre: str) -> List[str]:
    """
    Return a list of movie titles whose 'genre' matches the given genre, case-insensitively.

    Parameters:
        movies: A list of dictionaries, each with at least 'title' and 'genre' keys.
        genre: The target genre to match (case-insensitive).

    Returns:
        A list of matching movie titles. Returns an empty list if no matches.
    """
    if not genre:
        return []
    target = genre.strip().lower()
    return [
        movie["title"]
        for movie in movies
        if isinstance(movie, dict)
        and "title" in movie
        and "genre" in movie
        and str(movie["genre"]).strip().lower() == target
    ]


# Example usage and tests (5+ movies, 2-3 test calls)
movies_data = [
    {"title": "Inception", "genre": "Sci-Fi"},
    {"title": "Titanic", "genre": "Romance"},
    {"title": "The Dark Knight", "genre": "Action"},
    {"title": "Arrival", "genre": "sci-fi"},
    {"title": "La La Land", "genre": "Romance"},
    {"title": "Mad Max: Fury Road", "genre": "Action"},
]

print(recommend_by_genre(movies_data, "Sci-Fi"))   # Expected: ['Inception', 'Arrival']
print(recommend_by_genre(movies_data, "romance"))  # Expected: ['Titanic', 'La La Land']
print(recommend_by_genre(movies_data, "Comedy"))   # Expected: []
```

Example Output:



```
['Inception', 'Arrival']
['Titanic', 'La La Land']
[]
PS C:\Users\96653\OneDrive\Desktop\ai lat test 1>
```

TASK-4:

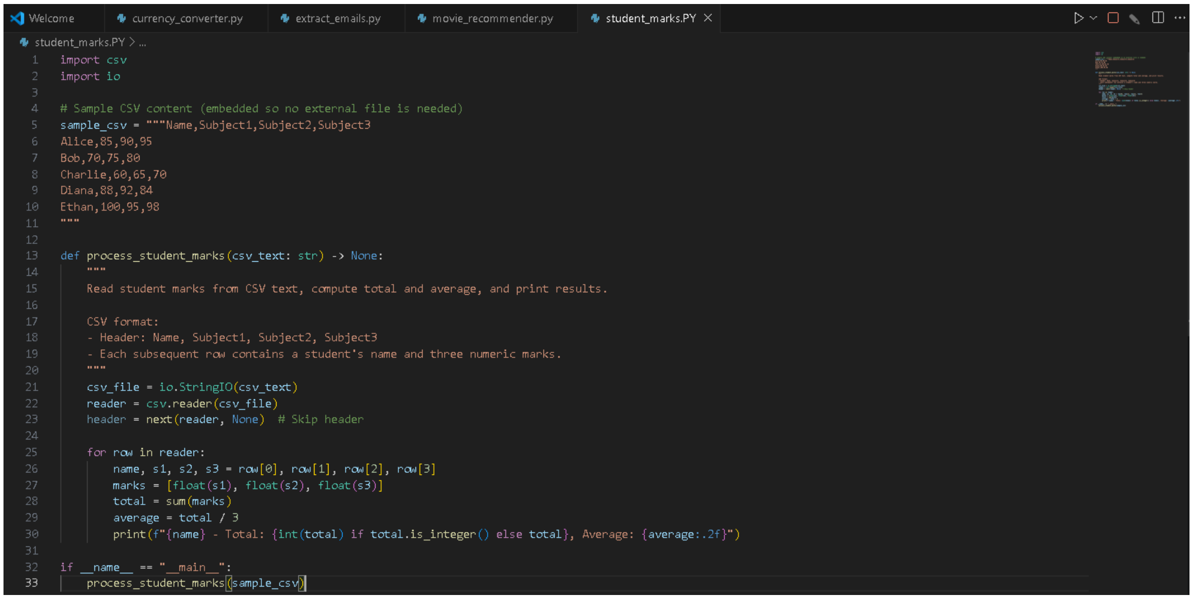
Write Python code that reads a CSV file containing student names and marks in 3 subjects. Calculate the total and average marks for each student. Use the Cursor AI tool

Prompt given to Cursor AI:

Write a Python program that reads a CSV file containing student names and marks in 3 subjects.
Requirements:

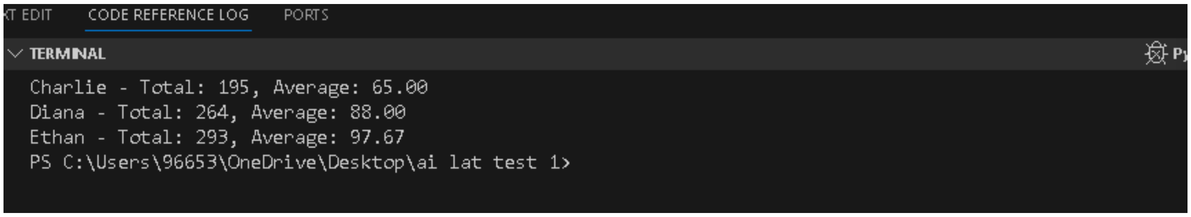
- Use the csv module.
- Each row has: Name, Subject1, Subject2, Subject3.
- For each student, calculate **total marks** and **average marks**.
- Print results in the format: Name - Total: X, Average: Y.
- Add a sample CSV file content inside the code (use io.StringIO) so the program runs without an external file.

Code from Cursor AI after the prompt:



```python
import csv
import io

# Sample CSV content (embedded so no external file is needed)
sample_csv = """Name,Subject1,Subject2,Subject3
Alice,85,90,95
Bob,70,75,80
Charlie,60,65,70
Diana,88,92,84
Ethan,100,95,98
"""

def process_student_marks(csv_text: str) -> None:
    """
    Read student marks from CSV text, compute total and average, and print results.

    CSV format:
    - Header: Name, Subject1, Subject2, Subject3
    - Each subsequent row contains a student's name and three numeric marks.
    """
    csv_file = io.StringIO(csv_text)
    reader = csv.reader(csv_file)
    header = next(reader, None)  # Skip header

    for row in reader:
        name, s1, s2, s3 = row[0], row[1], row[2], row[3]
        marks = [float(s1), float(s2), float(s3)]
        total = sum(marks)
        average = total / 3
        print(f"{name} - Total: {int(total) if total.is_integer() else total}, Average: {average:.2f}")

if __name__ == "__main__":
    process_student_marks(sample_csv)
```

Example Output:



```
Charlie - Total: 195, Average: 65.00
Diana - Total: 264, Average: 88.00
Ethan - Total: 293, Average: 97.67
PS C:\Users\96653\OneDrive\Desktop\ai lat test 1>
```