AI ASSISTED CODING

ASSISGNMENT-4.3

M.KEERTHANA

2403A51259

BATCH-11

CSE-GEN

Task 1:

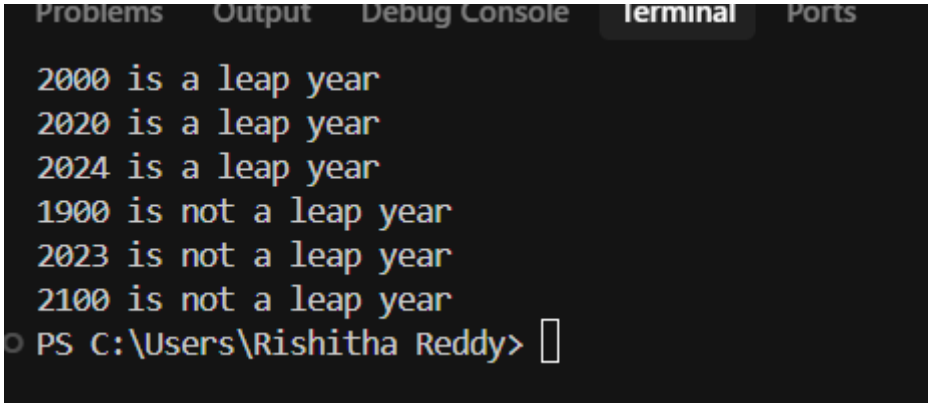
- Zero-shot: Prompt AI to write a function that checks whether a given year is a leap year.

Expected Output#1

- AI-generated function with no examples provided

```python
def is_leap_year(year):
    """
    Checks whether the given year is a leap year.

    Args:
        year (int): The year to check.

    Returns:
        bool: True if the year is a leap year, False otherwise.
    """
    return (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0)

# Test the function with some example years
if __name__ == "__main__":
    test_years = [2000, 2020, 2024, 1900, 2023, 2100]

    print("Leap Year Checker")
    print("=" * 20)

    for year in test_years:
        result = is_leap_year(year)
        status = "is" if result else "is not"
        print(f"{year} {status} a leap year")

```

OUTPUT:

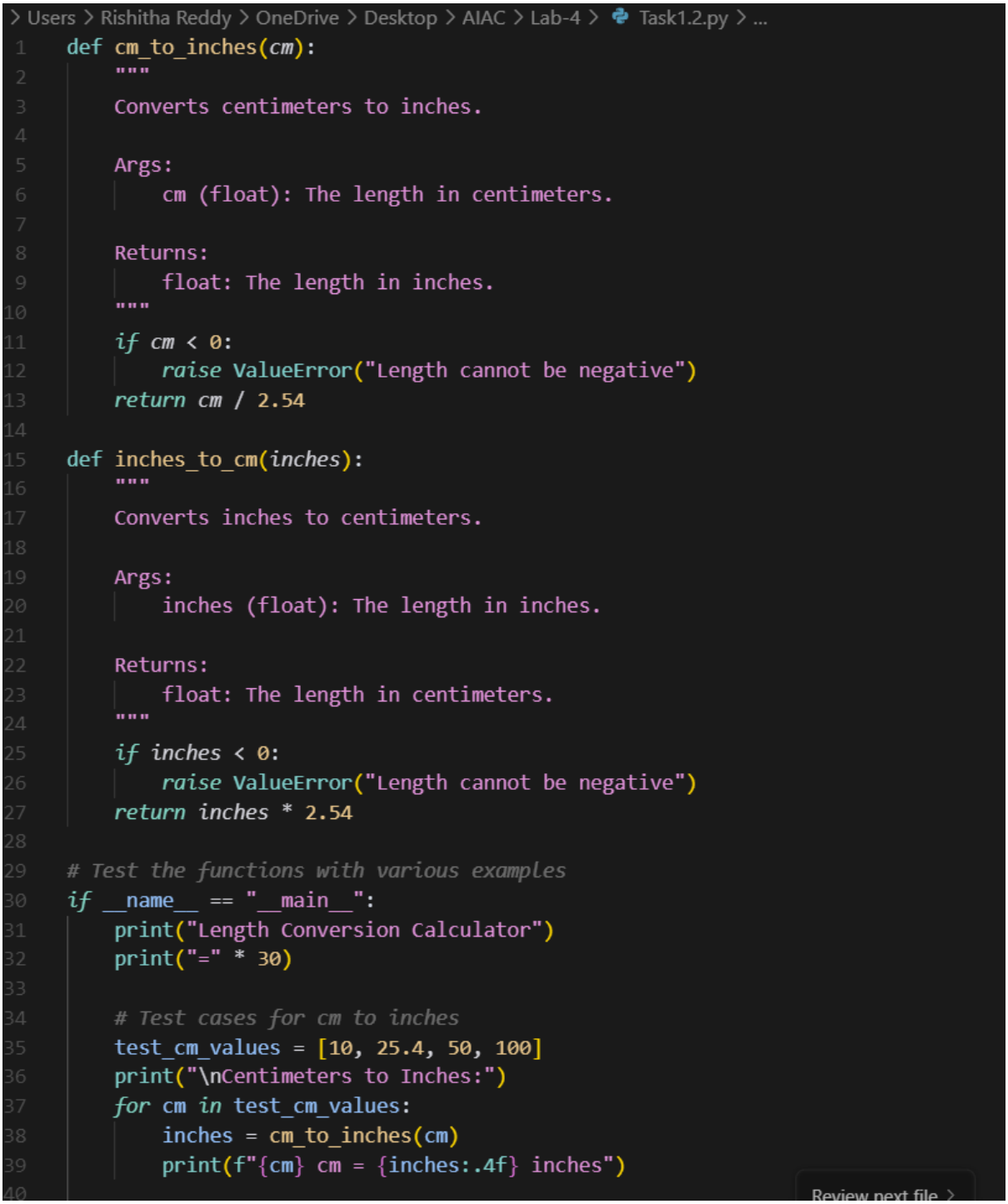


Task 2:

- One-shot: Give one input-output example to guide AI in writing a function that converts centimeters to inches.

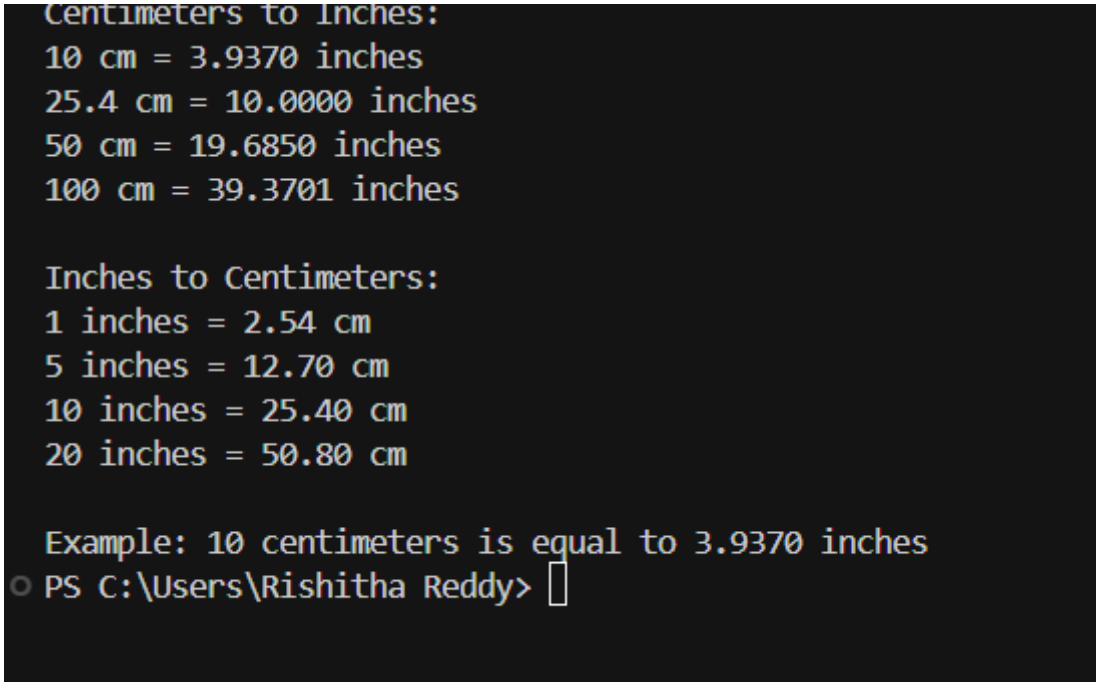
Expected Output#2

- Function with correct conversion logic

- 
Users > Rishitha Reddy > OneDrive > Desktop > AIAC > Lab-4 > Task1.2.py > ...
def cm_to_inches(cm):
"""
Converts centimeters to inches.
Args:
cm (float): The length in centimeters.
Returns:
float: The length in inches.
"""
if cm < 0:
raise ValueError("Length cannot be negative")
return cm / 2.54
def inches_to_cm(inches):
"""
Converts inches to centimeters.
Args:
inches (float): The length in inches.
Returns:
float: The length in centimeters.
"""
if inches < 0:
raise ValueError("Length cannot be negative")
return inches * 2.54
# Test the functions with various examples
if __name__ == "__main__":
print("Length Conversion Calculator")
print("=" * 30)
# Test cases for cm to inches
test_cm_values = [10, 25.4, 50, 100]
print("\nCentimeters to Inches:")
for cm in test_cm_values:
inches = cm_to_inches(cm)
print(f"{cm} cm = {inches:.4f} inches")
Review next file

```python
def inches_to_cm(inches):
    """
    Converts inches to centimeters.

    Args:
        inches (float): The length in inches.

    Returns:
        float: The length in centimeters.
    """
    if inches < 0:
        raise ValueError("Length cannot be negative")
    return inches * 2.54

# Test the functions with various examples
if __name__ == "__main__":
    print("Length Conversion Calculator")
    print("=" * 30)

    # Test cases for cm to inches
    test_cm_values = [10, 25.4, 50, 100]
    print("\nCentimeters to Inches:")
    for cm in test_cm_values:
        inches = cm_to_inches(cm)
        print(f"{cm} cm = {inches:.4f} inches")

    # Test cases for inches to cm
    test_inch_values = [1, 5, 10, 20]
    print("\nInches to Centimeters:")
    for inches in test_inch_values:
        cm = inches_to_cm(inches)
        print(f"{inches} inches = {cm:.2f} cm")

    # Interactive example
    print(f"\nExample: 10 centimeters is equal to {cm_to_inches(10):.4f} inches")
```


OUTPUT:



Task 3:

- Few-shot: Provide 2–3 examples to generate a function that formats full names as "Last, First".

Expected Output#3

Well-structured function respecting the examples

```python
def format_name(full_name):
    """
    Formats a full name as "Last, First".

    Args:
        full_name (str): The full name in "First Last" format.

    Returns:
        str: The formatted name as "Last, First".

    Examples:
        >>> format_name("John Smith")
        'Smith, John'
        >>> format_name("Alice Johnson")
        'Johnson, Alice'
        >>> format_name("Mary Ann Lee")
        'Lee, Mary Ann'
    """
    parts = full_name.strip().split()
    if len(parts) < 2:
        raise ValueError("Full name must contain at least first and last name")
    first = " ".join(parts[:-1])
    last = parts[-1]
    return f"{last}, {first}"

# Example usage
if __name__ == "__main__":
    print(format_name("John Smith"))        # Output: Smith, John
    print(format_name("Alice Johnson"))     # Output: Johnson, Alice
    print(format_name("Mary Ann Lee"))      # Output: Lee, Mary Ann
```

Ctrl+L to chat, Ctrl+K to generate

OUTPUT:

```
Smith, John
Johnson, Alice
Lee, Mary Ann
PS C:\Users\Rishitha Reddy>
```

Task 4

- Compare zero-shot and few-shot prompts for writing a function that counts the number of vowels in a string.

Expected Output#4

- Functional output and comparative reflection

```python
# Zero-shot implementation (based only on the task description)
def count_vowels_zero_shot(s):
    vowels = "aeiouAEIOU"
    count = 0
    for char in s:
        if char in vowels:
            count += 1
    return count

# Few-shot implementation (informed by examples)
def count_vowels_few_shot(s):
    # Based on the examples, the function should be case-insensitive and count all vowels
    return sum(1 for char in s if char.lower() in 'aeiou')

# Test cases for comparison
test_strings = [
    "hello",        # 2 vowels
    "AIAC",         # 3 vowels
    "sky",          # 0 vowels
    "Python",       # 1 vowel
    "Beautiful",    # 5 vowels
    "bcd",          # 0 vowels
    "AEIOUaeiou",   # 10 vowels
]

print("Comparing Zero-shot and Few-shot Prompt Implementations:\n")
print(f"{'Input String':<15} {'Zero-shot':<10} {'Few-shot':<10}")
print("-" * 40)
for s in test_strings:
    zero_shot_result = count_vowels_zero_shot(s)
    few_shot_result = count_vowels_few_shot(s)
    print(f"{s:<15} {zero_shot_result:<10} {few_shot_result:<10}")

print("\nReflection:")
print("""
- Both zero-shot and few-shot implementations produce correct results for the provided test cases.
- The zero-shot approach relies solely on the task description, so it may be more verbose or less optimized.
- The few-shot approach, informed by examples, can lead to more concise and idiomatic code (e.g., using generator expressions).
- Few-shot prompts help clarify edge cases (like case sensiti  Review next file >  ted behavior, reducing ambiguity.
""")
```

OUTPUT:

```
Rtop/AIAC/Lab-4/Task1.4.py
Comparing Zero-shot and Few-shot Prompt Implementations:

Input String    Zero-shot  Few-shot
----------------------------------------
hello           2          2
AIAC            3          3
sky             0          0
Python          1          1
Beautiful       5          5
bcd             0          0
AEIOUaeiou      10         10

Reflection:

- Both zero-shot and few-shot implementations produce correct results for the provided test cases.
- The zero-shot approach relies solely on the task description, so it may be more verbose or less optimized.
- The few-shot approach, informed by examples, can lead to more concise and idiomatic code (e.g., using generator expressions).
- Few-shot prompts help clarify edge cases (like case sensitivity) and expected behavior, reducing ambiguity.

PS C:\Users\Rishitha Reddy>
```

Task 5:

- Use few-shot prompting to generate a function that reads a .txt file and returns the number of lines.

Expected Output#5

- Working file-processing function with AI-guided logic

```python
def count_lines_in_file(example):
    """
    Reads a .txt file and returns the number of lines.

    Few-shot examples (inputs -> outputs):
    - "notes.txt" -> 12
    - "empty.txt" -> 0
    - "C:/data/logs/today.txt" -> 347

    Args:
        filename (str): The path to the .txt file.

    Returns:
        int: The number of lines in the file.
    """
    with open(example, 'r', encoding='utf-8') as file:
        return sum(1 for _ in file)
Ctrl+L to chat, Ctrl+K to generate
# Interactive usage
if __name__ == "__main__":
    print("File Line Counter")
    print("=" * 18)
    user_path = input("Enter the path to a .txt file (e.g., sample.txt): ").strip().strip('"')
    if not user_path:
        print("No input provided.")
    elif not user_path.lower().endswith('.txt'):
        print("Please provide a .txt file.")
    else:
        try:
            lines = count_lines_in_file(user_path)
            print(f"Number of lines in '{user_path}': {lines}")
        except FileNotFoundError:
            print(f"File not found: {user_path}")
        except PermissionError:
            print(f"Permission denied: {user_path}")
        except UnicodeDecodeError:
            print(f"Encoding error while reading: {user_path}")
```

OUTPUT:

```
==================
Enter the path to a .txt file (e.g., sample.txt): example.txt
Number of lines in 'example.txt': 3
PS C:\Users\Rishitha Reddy\OneDrive\Desktop\AIAC\Lab-4>
```