Assignment-2.1

S.Rajesh 2303A52301 B-35

Task 1: Statistical Summary for Survey Data Scenario:

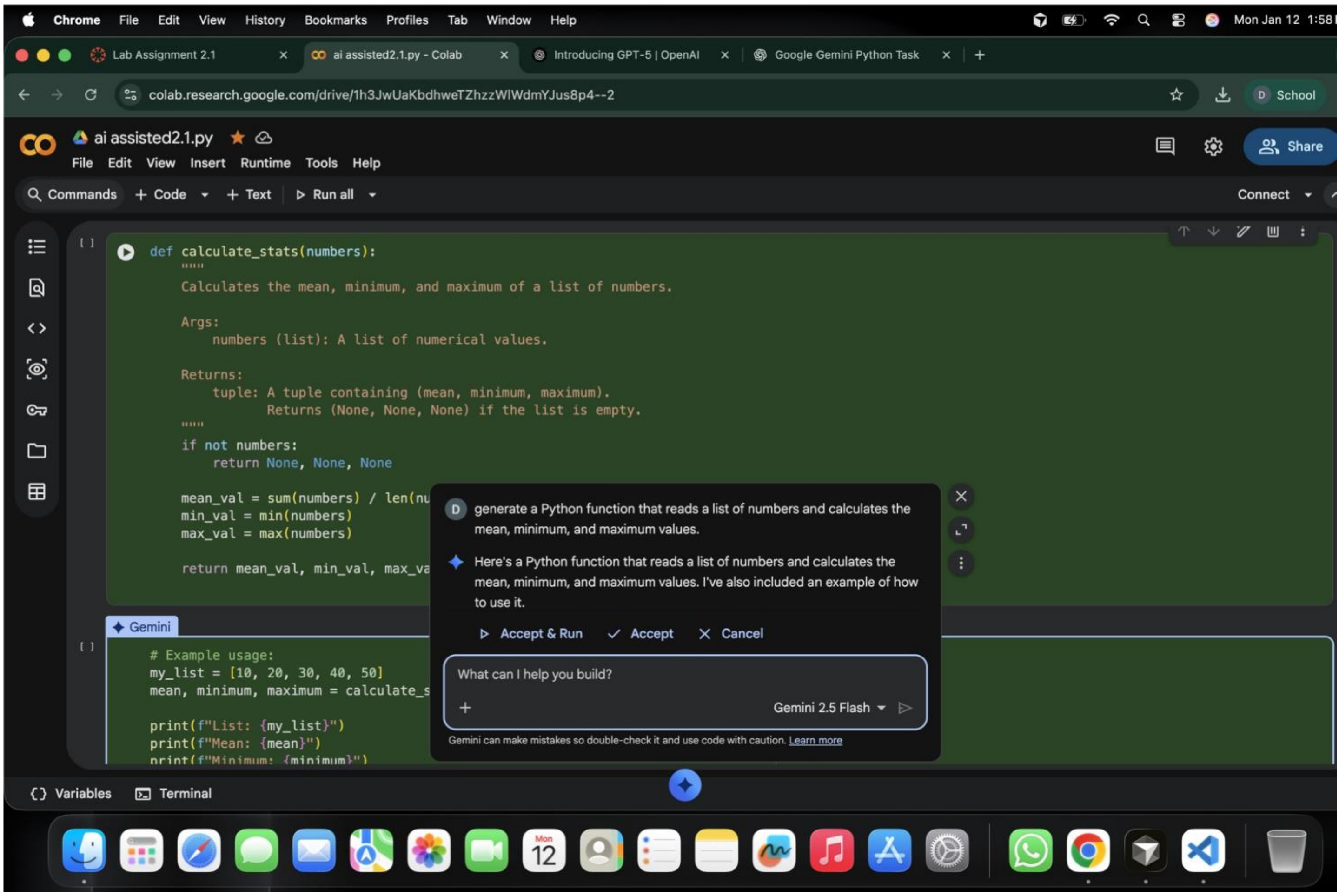
You are a data analyst intern working with survey responses stored as numerical lists.

▯ Task:

Use Google Gemini in Colab to generate a Python function that reads a list of numbers and calculates the mean, minimum, and maximum values.

Output:

Screenshot of Gemini prompt and result



```
def calculate_stats(numbers):
    """
    Calculates the mean, minimum, and maximum of a list of numbers.

    Args:
        numbers (list): A list of numerical values.

    Returns:
        tuple: A tuple containing (mean, minimum, maximum).
               Returns (None, None, None) if the list is empty.
    """
    if not numbers:
        return None, None, None

    mean_val = sum(numbers) / len(nu
    min_val = min(numbers)
    max_val = max(numbers)

    return mean_val, min_val, max_va
```

```
# Example usage:
my_list = [10, 20, 30, 40, 50]
mean, minimum, maximum = calculate_s

print(f"List: {my_list}")
print(f"Mean: {mean}")
print(f"Minimum: {minimum}")
```

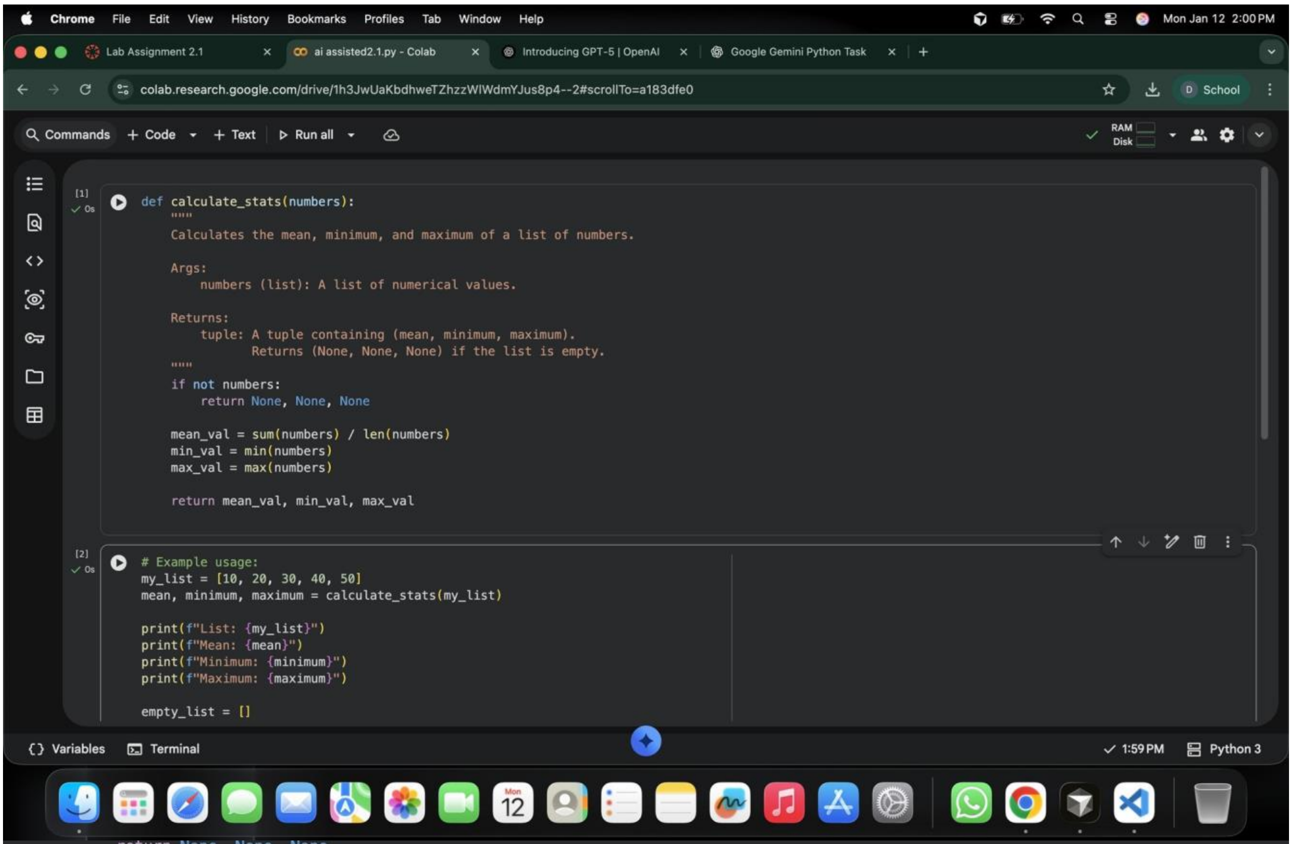
D generate a Python function that reads a list of numbers and calculates the mean, minimum, and maximum values.

Here's a Python function that reads a list of numbers and calculates the mean, minimum, and maximum values. I've also included an example of how to use it.

Gemini can make mistakes so double-check it and use code with caution. Learn more

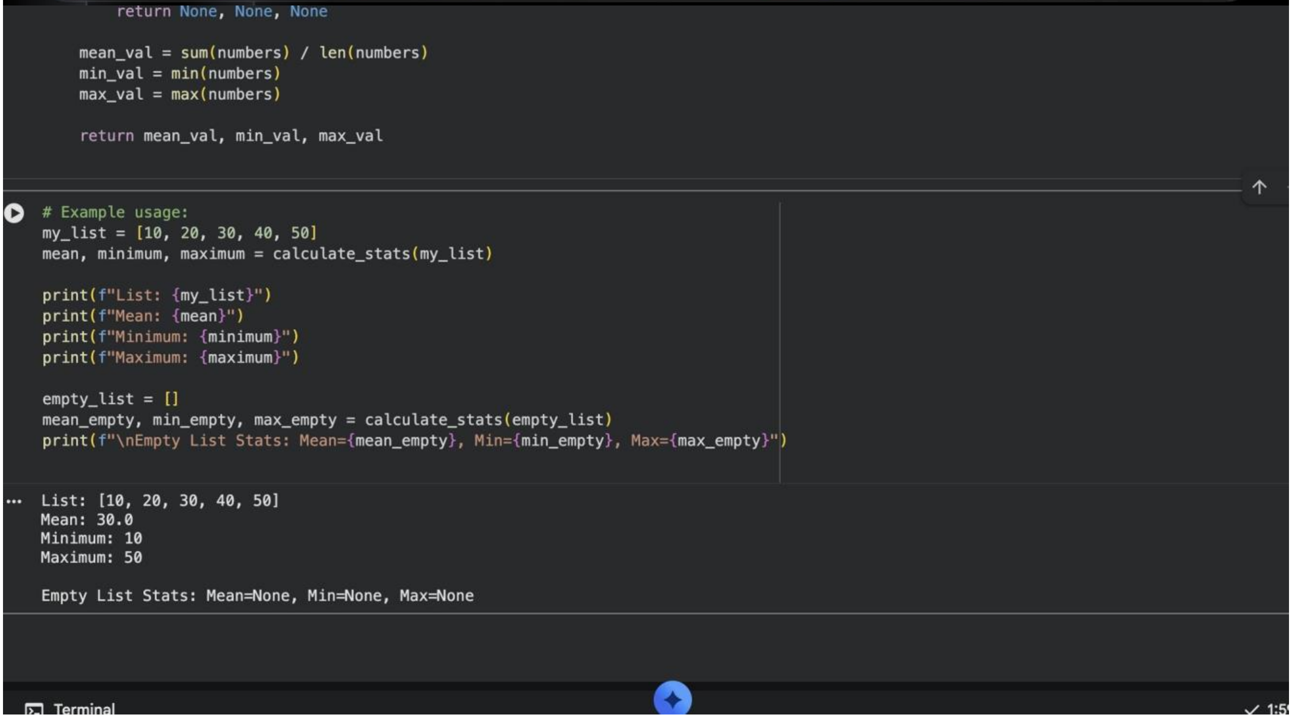
Chrome File Edit View History Bookmarks Profiles Tab Window Help
Mon Jan 12 1:58
Lab Assignment 2.1
ai assisted2.1.py - Colab
Introducing GPT-5 | OpenAI
Google Gemini Python Task
colab.research.google.com/drive/1h3JwUaKbdhweTZhzzWlWdmYJus8p4--2
School
ai assisted2.1.py
File Edit View Insert Runtime Tools Help
Share
Commands
+ Code
+ Text
Run all
Connect
Gemini
Accept & Run
Accept
Cancel
What can I help you build?
Gemini 2.5 Flash
Variables
Terminal

Correct Python function Output
shown in Colab


Chrome File Edit View History Bookmarks Profiles Tab Window Help
Mon Jan 12 2:00 PM
Lab Assignment 2.1
ai assisted2.1.py - Colab
Introducing GPT-5 | OpenAI
Google Gemini Python Task
colab.research.google.com/drive/1h3JwUaKbdhweTZhzzWIWdmYJus8p4--2#scrollTo=a183dfe0
School
Commands
+ Code
+ Text
Run all
RAM
Disk
[1]
0s
[2]
0s
Variables
Terminal
1:59 PM
Python 3

```python
def calculate_stats(numbers):
    """
    Calculates the mean, minimum, and maximum of a list of numbers.

    Args:
        numbers (list): A list of numerical values.

    Returns:
        tuple: A tuple containing (mean, minimum, maximum).
               Returns (None, None, None) if the list is empty.
    """
    if not numbers:
        return None, None, None

    mean_val = sum(numbers) / len(numbers)
    min_val = min(numbers)
    max_val = max(numbers)

    return mean_val, min_val, max_val
```

```python
# Example usage:
my_list = [10, 20, 30, 40, 50]
mean, minimum, maximum = calculate_stats(my_list)

print(f"List: {my_list}")
print(f"Mean: {mean}")
print(f"Minimum: {minimum}")
print(f"Maximum: {maximum}")

empty_list = []
```



```python
        return None, None, None

    mean_val = sum(numbers) / len(numbers)
    min_val = min(numbers)
    max_val = max(numbers)

    return mean_val, min_val, max_val
```

```python
# Example usage:
my_list = [10, 20, 30, 40, 50]
mean, minimum, maximum = calculate_stats(my_list)

print(f"List: {my_list}")
print(f"Mean: {mean}")
print(f"Minimum: {minimum}")
print(f"Maximum: {maximum}")

empty_list = []
mean_empty, min_empty, max_empty = calculate_stats(empty_list)
print(f"\nEmpty List Stats: Mean={mean_empty}, Min={min_empty}, Max={max_empty}")
```

```
List: [10, 20, 30, 40, 50]
Mean: 30.0
Minimum: 10
Maximum: 50

Empty List Stats: Mean=None, Min=None, Max=None
```

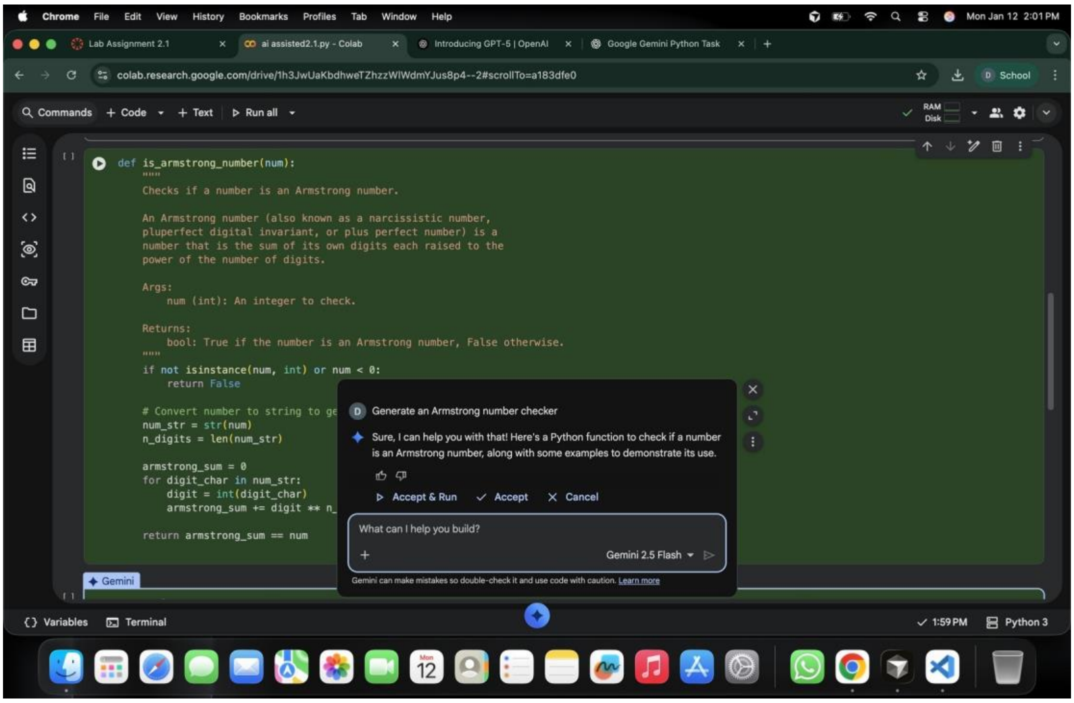
Task 2: Armstrong Number – AI Comparison ▯ Scenario:

You are evaluating AI tools for numeric validation logic.

▯ Task:

Generate an Armstrong number checker using Gemini and GitHub Copilot.

Compare their outputs, logic style, and clarity.

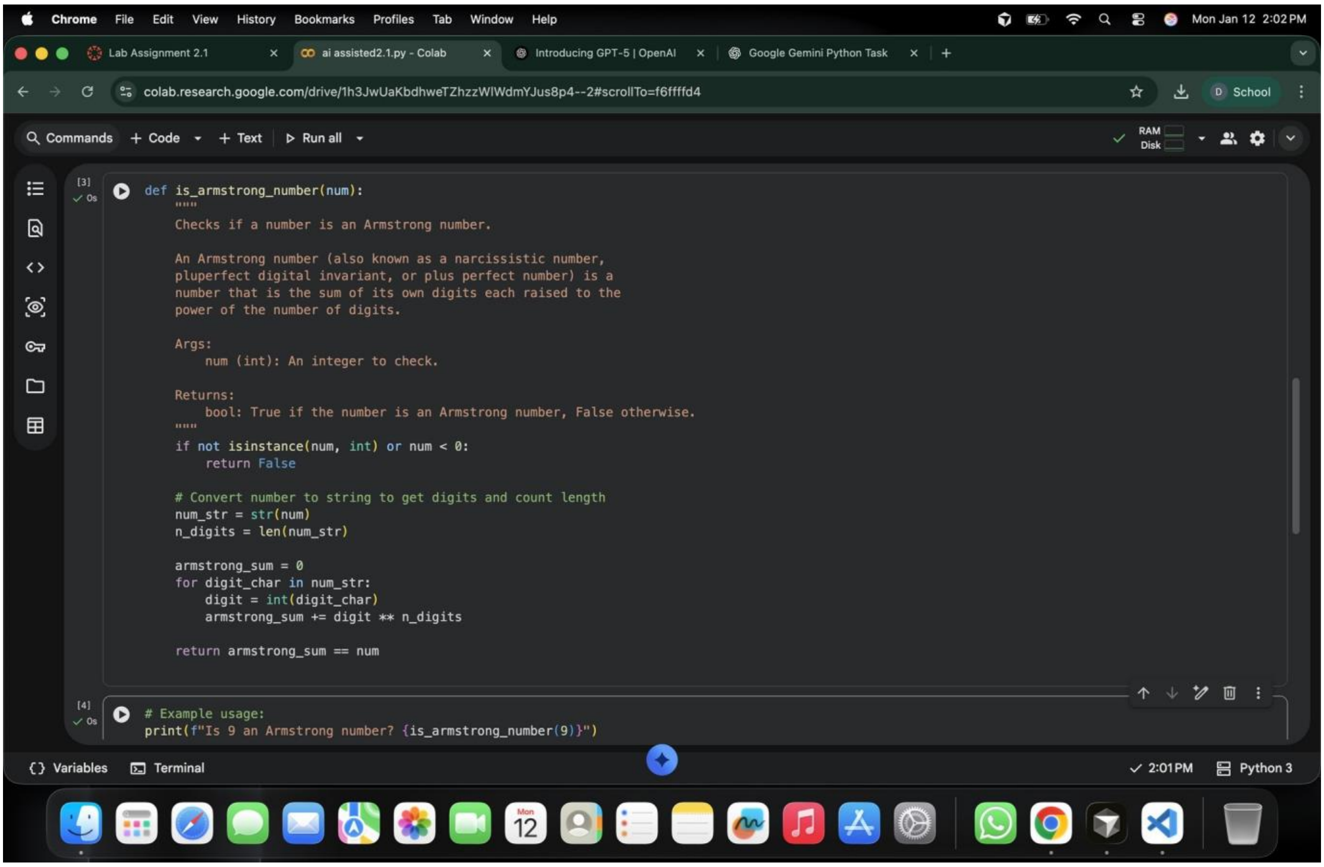

Chrome File Edit View History Bookmarks Profiles Tab Window Help
Mon Jan 12 2:01 PM
Lab Assignment 2.1
ai assisted2.1.py - Colab
Introducing GPT-5 | OpenAI
Google Gemini Python Task
colab.research.google.com/drive/1h3JwUaKbdhweTZhzzWlWdmYJus8p4--2#scrollTo=a183dfe0
School
Commands
+ Code
+ Text
Run all
RAM
Disk
Gemini
Variables
Terminal
1:59 PM
Python 3

```
def is_armstrong_number(num):
    """
    Checks if a number is an Armstrong number.

    An Armstrong number (also known as a narcissistic number,
    pluperfect digital invariant, or plus perfect number) is a
    number that is the sum of its own digits each raised to the
    power of the number of digits.

    Args:
        num (int): An integer to check.

    Returns:
        bool: True if the number is an Armstrong number, False otherwise.
    """
    if not isinstance(num, int) or num < 0:
        return False

    # Convert number to string to ge
    num_str = str(num)
    n_digits = len(num_str)

    armstrong_sum = 0
    for digit_char in num_str:
        digit = int(digit_char)
        armstrong_sum += digit ** n_

    return armstrong_sum == num
```

D Generate an Armstrong number checker

Sure, I can help you with that! Here's a Python function to check if a number is an Armstrong number, along with some examples to demonstrate its use.

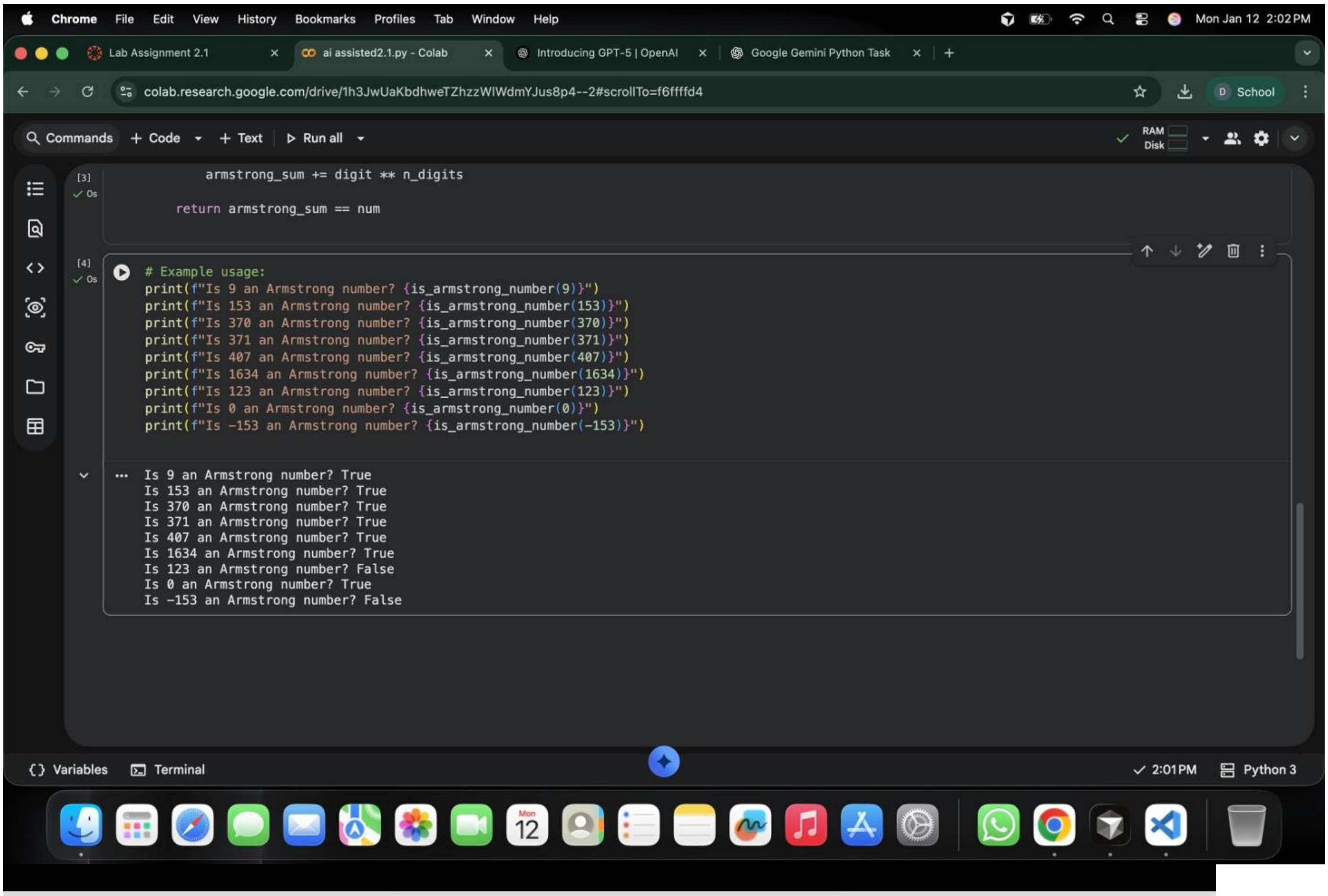
Gemini can make mistakes so double-check it and use code with caution. Learn more


Chrome File Edit View History Bookmarks Profiles Tab Window Help
Mon Jan 12 2:02 PM
Lab Assignment 2.1
ai assisted2.1.py - Colab
Introducing GPT-5 | OpenAI
Google Gemini Python Task
colab.research.google.com/drive/1h3JwUaKbdhweTZhzzWlWdmYJus8p4--2#scrollTo=f6ffffd4
School
Commands
+ Code
+ Text
Run all
RAM
Disk
[3]
0s
[4]
0s
Variables
Terminal
2:01PM
Python 3

```
def is_armstrong_number(num):
    """
    Checks if a number is an Armstrong number.

    An Armstrong number (also known as a narcissistic number,
    pluperfect digital invariant, or plus perfect number) is a
    number that is the sum of its own digits each raised to the
    power of the number of digits.

    Args:
        num (int): An integer to check.

    Returns:
        bool: True if the number is an Armstrong number, False otherwise.
    """
    if not isinstance(num, int) or num < 0:
        return False

    # Convert number to string to get digits and count length
    num_str = str(num)
    n_digits = len(num_str)

    armstrong_sum = 0
    for digit_char in num_str:
        digit = int(digit_char)
        armstrong_sum += digit ** n_digits

    return armstrong_sum == num
```

```
# Example usage:
print(f"Is 9 an Armstrong number? {is_armstrong_number(9)}")
```

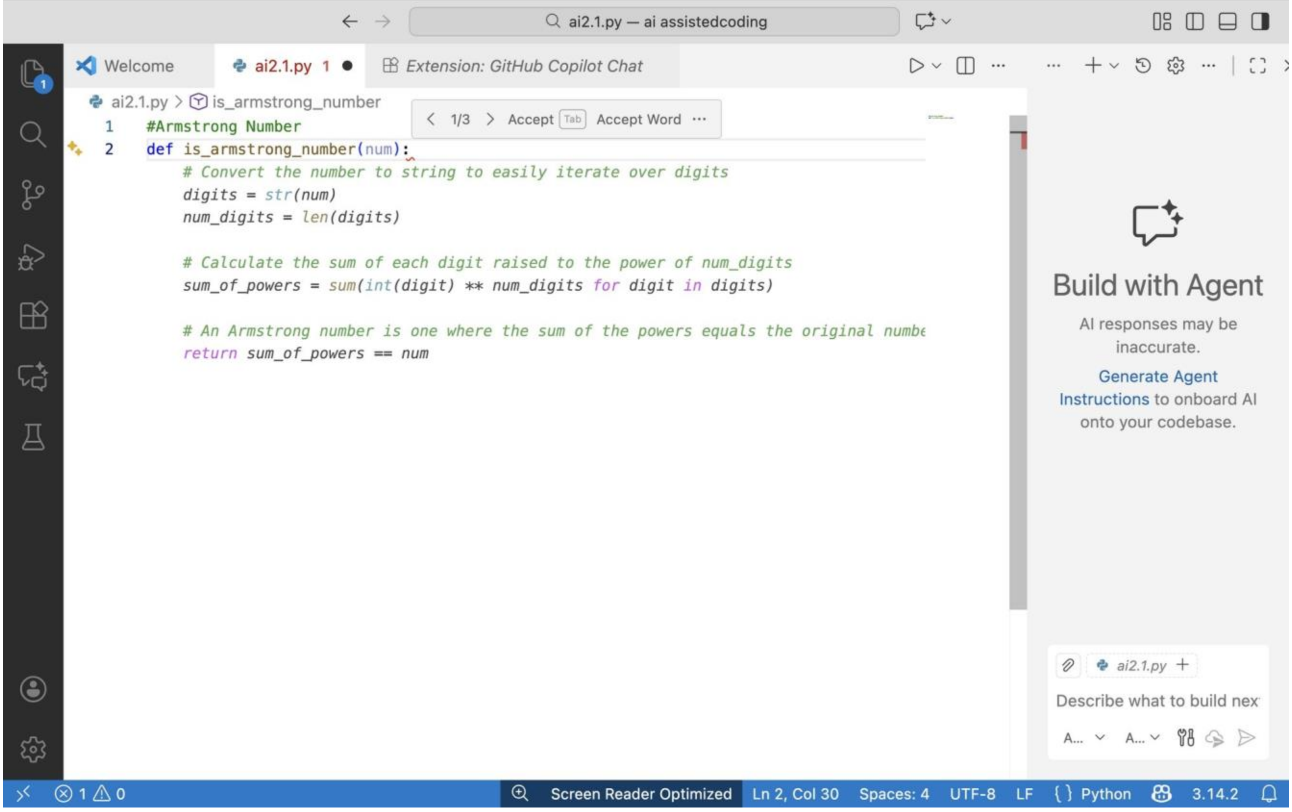
Using vs and GitHub copilot


Chrome File Edit View History Bookmarks Profiles Tab Window Help
Mon Jan 12 2:02 PM
Lab Assignment 2.1
ai assisted2.1.py - Colab
Introducing GPT-5 | OpenAI
Google Gemini Python Task
colab.research.google.com/drive/1h3JwUaKbdhweTZhzzWIWdmYJus8p4--2#scrollTo=f6ffffd4
School
Commands + Code + Text ▷ Run all
RAM
Disk
[3]
0s
[4]
0s
Variables
Terminal
✓ 2:01 PM
Python 3

```
        armstrong_sum += digit ** n_digits

    return armstrong_sum == num
```

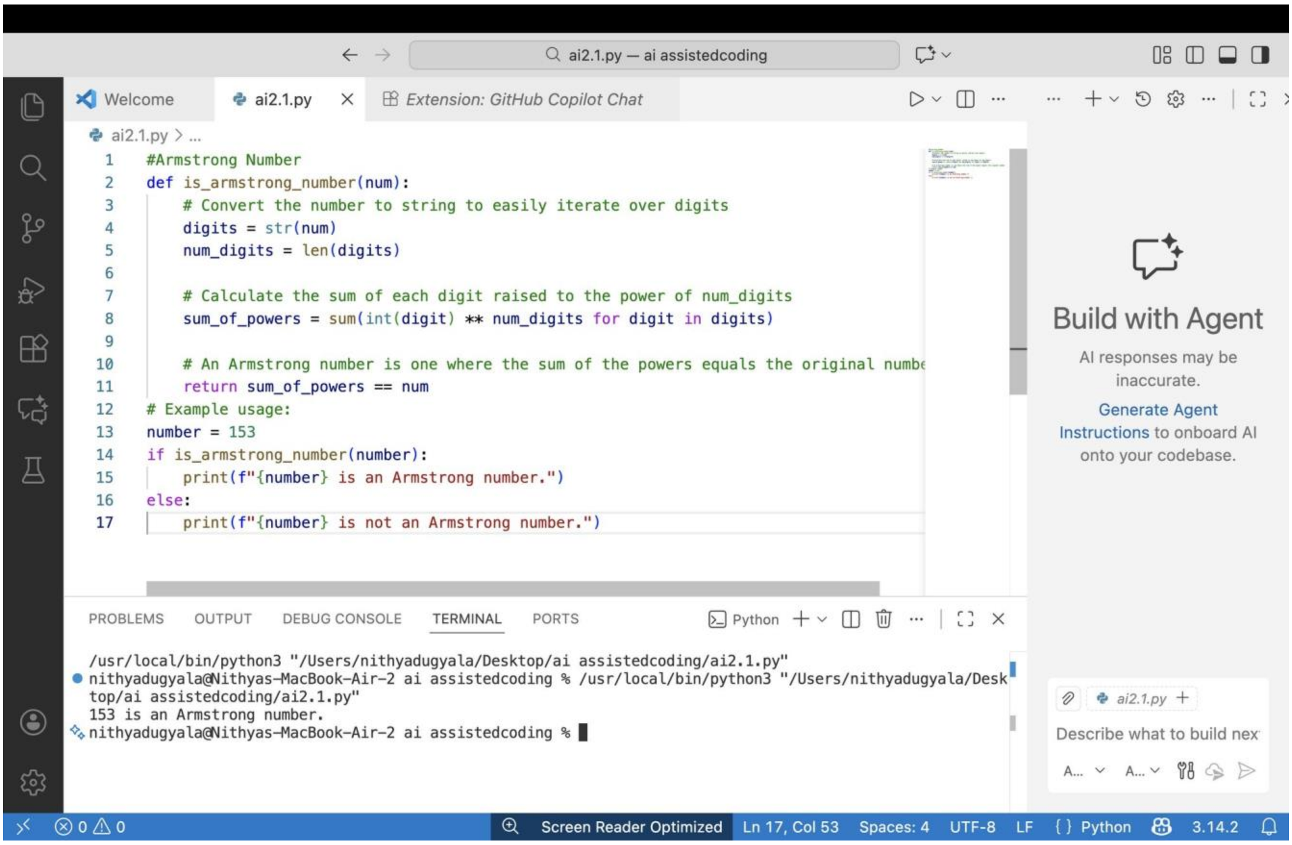
```
# Example usage:
print(f"Is 9 an Armstrong number? {is_armstrong_number(9)}")
print(f"Is 153 an Armstrong number? {is_armstrong_number(153)}")
print(f"Is 370 an Armstrong number? {is_armstrong_number(370)}")
print(f"Is 371 an Armstrong number? {is_armstrong_number(371)}")
print(f"Is 407 an Armstrong number? {is_armstrong_number(407)}")
print(f"Is 1634 an Armstrong number? {is_armstrong_number(1634)}")
print(f"Is 123 an Armstrong number? {is_armstrong_number(123)}")
print(f"Is 0 an Armstrong number? {is_armstrong_number(0)}")
print(f"Is -153 an Armstrong number? {is_armstrong_number(-153)}")
```

```
Is 9 an Armstrong number? True
Is 153 an Armstrong number? True
Is 370 an Armstrong number? True
Is 371 an Armstrong number? True
Is 407 an Armstrong number? True
Is 1634 an Armstrong number? True
Is 123 an Armstrong number? False
Is 0 an Armstrong number? True
Is -153 an Armstrong number? False
```


ai2.1.py — ai assistedcoding
Welcome
ai2.1.py 1
Extension: GitHub Copilot Chat
ai2.1.py > is_armstrong_number
1/3 Accept Tab Accept Word
Build with Agent
AI responses may be inaccurate.
Generate Agent Instructions to onboard AI onto your codebase.
ai2.1.py
Describe what to build nex
Screen Reader Optimized
Ln 2, Col 30
Spaces: 4
UTF-8
LF
{ } Python
3.14.2

```
#Armstrong Number
def is_armstrong_number(num):
    # Convert the number to string to easily iterate over digits
    digits = str(num)
    num_digits = len(digits)

    # Calculate the sum of each digit raised to the power of num_digits
    sum_of_powers = sum(int(digit) ** num_digits for digit in digits)

    # An Armstrong number is one where the sum of the powers equals the original numbe
    return sum_of_powers == num
```

Task 3: Leap Year Validation Using Cursor AI



```python
#Armstrong Number
def is_armstrong_number(num):
    # Convert the number to string to easily iterate over digits
    digits = str(num)
    num_digits = len(digits)

    # Calculate the sum of each digit raised to the power of num_digits
    sum_of_powers = sum(int(digit) ** num_digits for digit in digits)

    # An Armstrong number is one where the sum of the powers equals the original numbe
    return sum_of_powers == num
# Example usage:
number = 153
if is_armstrong_number(number):
    print(f"{number} is an Armstrong number.")
else:
    print(f"{number} is not an Armstrong number.")
```

```
/usr/local/bin/python3 "/Users/nithyadugyala/Desktop/ai assistedcoding/ai2.1.py"
nithyadugyala@Nithyas-MacBook-Air-2 ai assistedcoding % /usr/local/bin/python3 "/Users/nithyadugyala/Desk
top/ai assistedcoding/ai2.1.py"
153 is an Armstrong number.
nithyadugyala@Nithyas-MacBook-Air-2 ai assistedcoding % 
```

AI responses may be inaccurate.

Generate Agent Instructions to onboard AI onto your codebase.

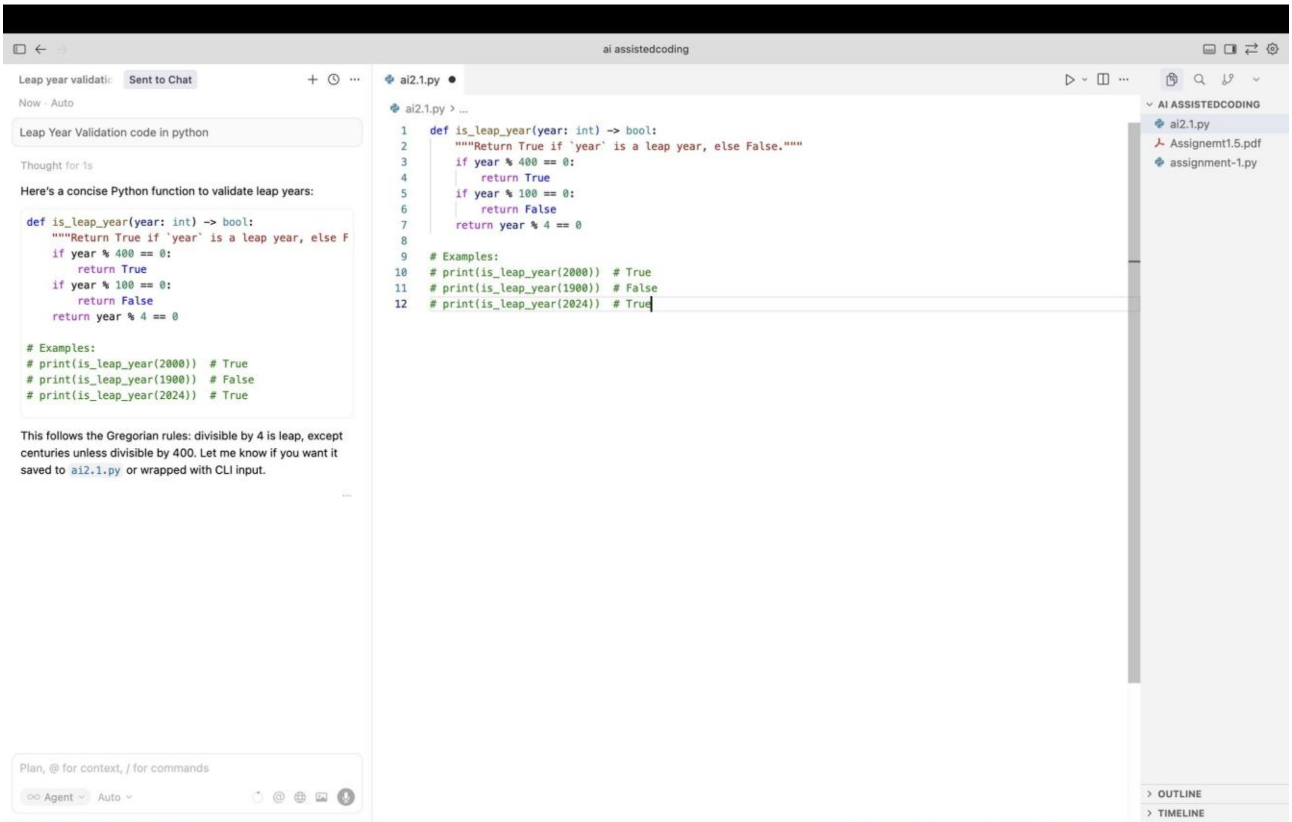
 Scenario:

You are validating a calendar module for a backend system.

 Task:

Use Cursor AI to generate a Python program that checks whether a given year is a leap year.

Use at least two different prompts and observe changes in code.



Leap Year Validation code in python

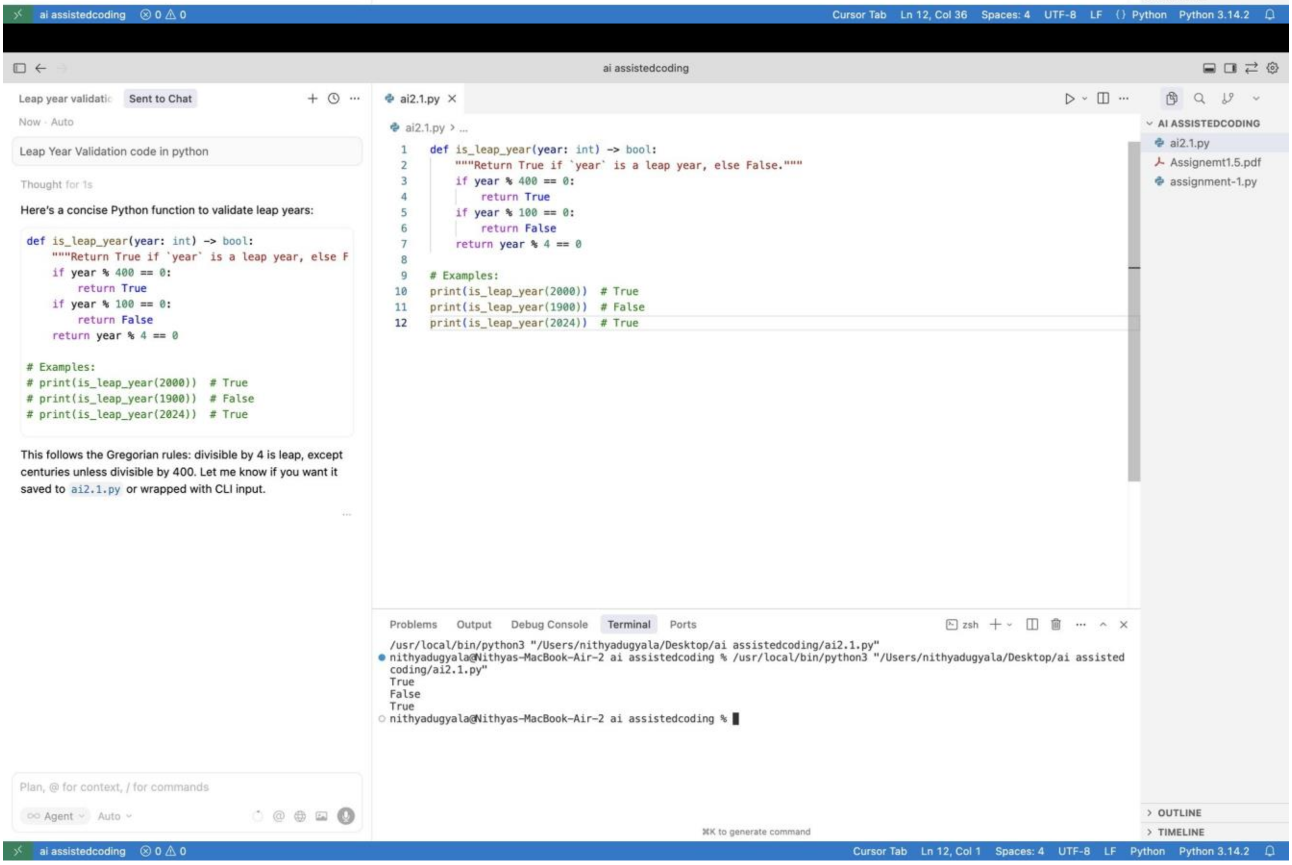
Thought for 1s

Here's a concise Python function to validate leap years:

```python
def is_leap_year(year: int) -> bool:
    """Return True if `year` is a leap year, else F
    if year % 400 == 0:
        return True
    if year % 100 == 0:
        return False
    return year % 4 == 0

# Examples:
# print(is_leap_year(2000))  # True
# print(is_leap_year(1900))  # False
# print(is_leap_year(2024))  # True
```

This follows the Gregorian rules: divisible by 4 is leap, except centuries unless divisible by 400. Let me know if you want it saved to `ai2.1.py` or wrapped with CLI input.

```python
def is_leap_year(year: int) -> bool:
    """Return True if `year` is a leap year, else False."""
    if year % 400 == 0:
        return True
    if year % 100 == 0:
        return False
    return year % 4 == 0

# Examples:
# print(is_leap_year(2000))  # True
# print(is_leap_year(1900))  # False
# print(is_leap_year(2024))  # True
```



Leap Year Validation code in python

Thought for 1s

Here's a concise Python function to validate leap years:

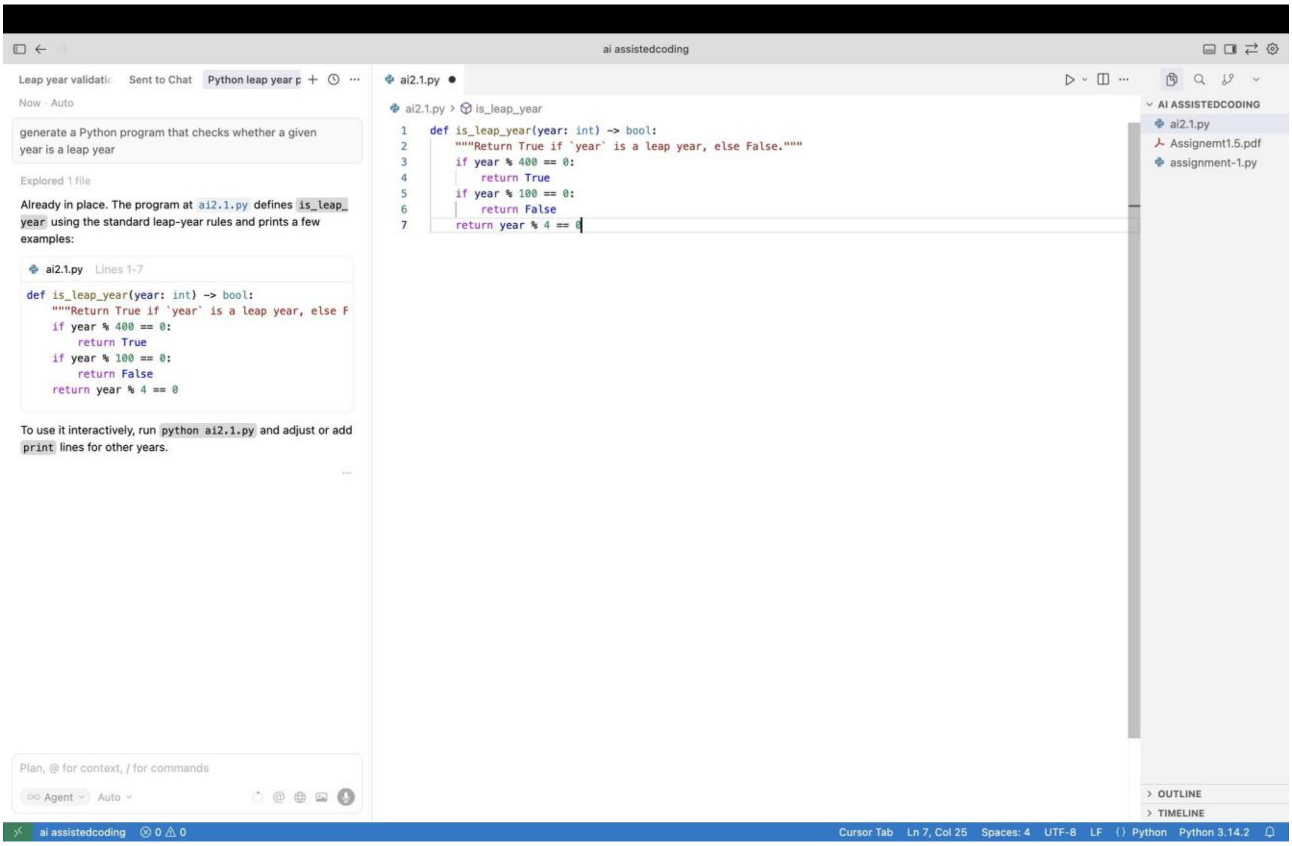
```python
def is_leap_year(year: int) -> bool:
    """Return True if `year` is a leap year, else F
    if year % 400 == 0:
        return True
    if year % 100 == 0:
        return False
    return year % 4 == 0

# Examples:
# print(is_leap_year(2000))  # True
# print(is_leap_year(1900))  # False
# print(is_leap_year(2024))  # True
```

This follows the Gregorian rules: divisible by 4 is leap, except centuries unless divisible by 400. Let me know if you want it saved to `ai2.1.py` or wrapped with CLI input.

```python
def is_leap_year(year: int) -> bool:
    """Return True if `year` is a leap year, else False."""
    if year % 400 == 0:
        return True
    if year % 100 == 0:
        return False
    return year % 4 == 0

# Examples:
print(is_leap_year(2000))  # True
print(is_leap_year(1900))  # False
print(is_leap_year(2024))  # True
```

```
/usr/local/bin/python3 "/Users/nithyadugyala/Desktop/ai assistedcoding/ai2.1.py"
nithyadugyala@Nithyas-MacBook-Air-2 ai assistedcoding % /usr/local/bin/python3 "/Users/nithyadugyala/Desktop/ai assisted
coding/ai2.1.py"
True
False
True
nithyadugyala@Nithyas-MacBook-Air-2 ai assistedcoding %
```

Other prompt:



generate a Python program that checks whether a given year is a leap year

Explored 1 file

Already in place. The program at `ai2.1.py` defines `is_leap_year` using the standard leap-year rules and prints a few examples:

ai2.1.py Lines 1-7

```
def is_leap_year(year: int) -> bool:
    """Return True if `year` is a leap year, else F
    if year % 400 == 0:
        return True
    if year % 100 == 0:
        return False
    return year % 4 == 0
```

To use it interactively, run `python ai2.1.py` and adjust or add `print` lines for other years.

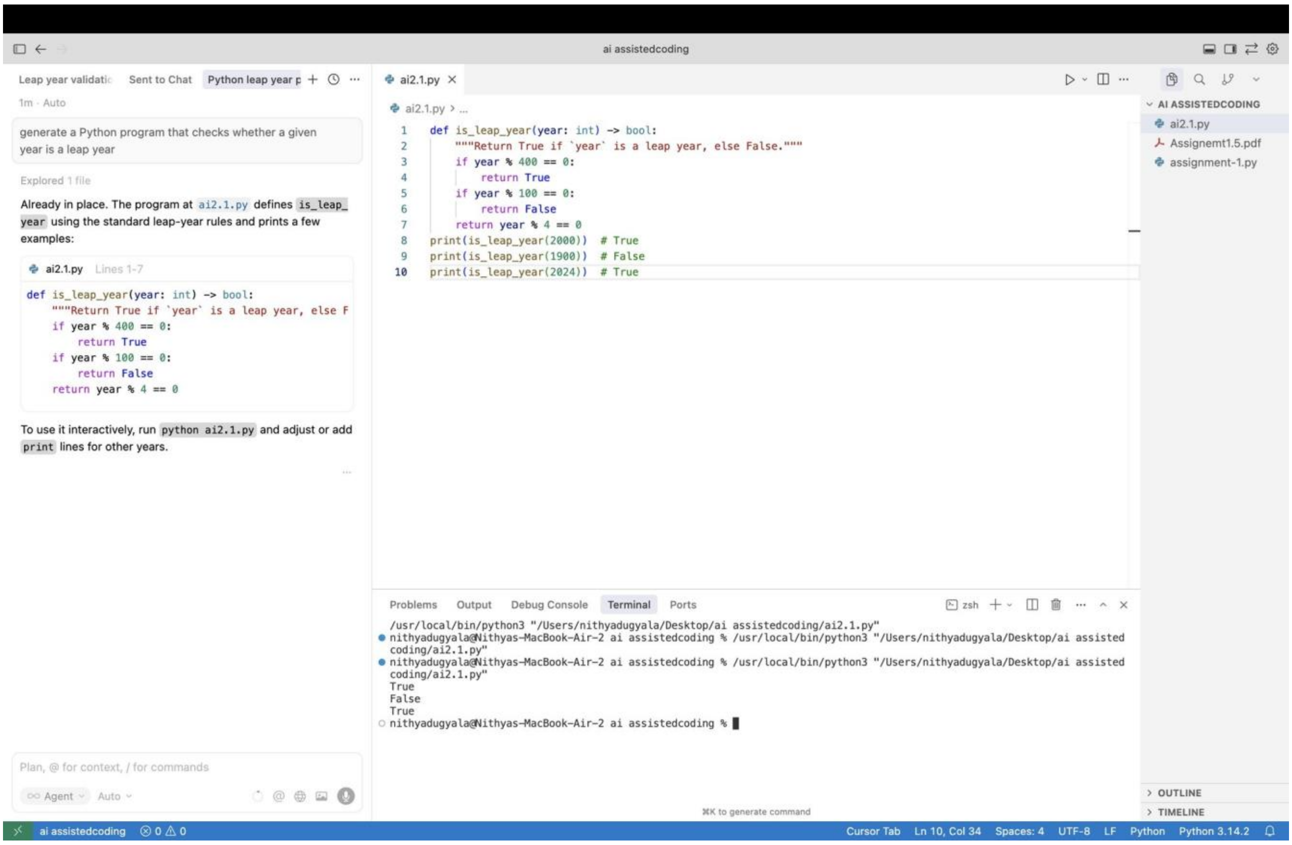
```
def is_leap_year(year: int) -> bool:
    """Return True if `year` is a leap year, else False."""
    if year % 400 == 0:
        return True
    if year % 100 == 0:
        return False
    return year % 4 == 0
```

task 4: Student Logic + AI Refactoring (Odd/Even Sum) Scenario: Company policy requires developers to write logic before using AI.

- Task:

Write a Python program that calculates the sum of odd and even numbers in a tuple, then refactor it using any AI tool. Expected Output:

- Original code
- Refactored code
- Explanation of improvements



generate a Python program that checks whether a given year is a leap year

Explored 1 file

Already in place. The program at `ai2.1.py` defines `is_leap_year` using the standard leap-year rules and prints a few examples:

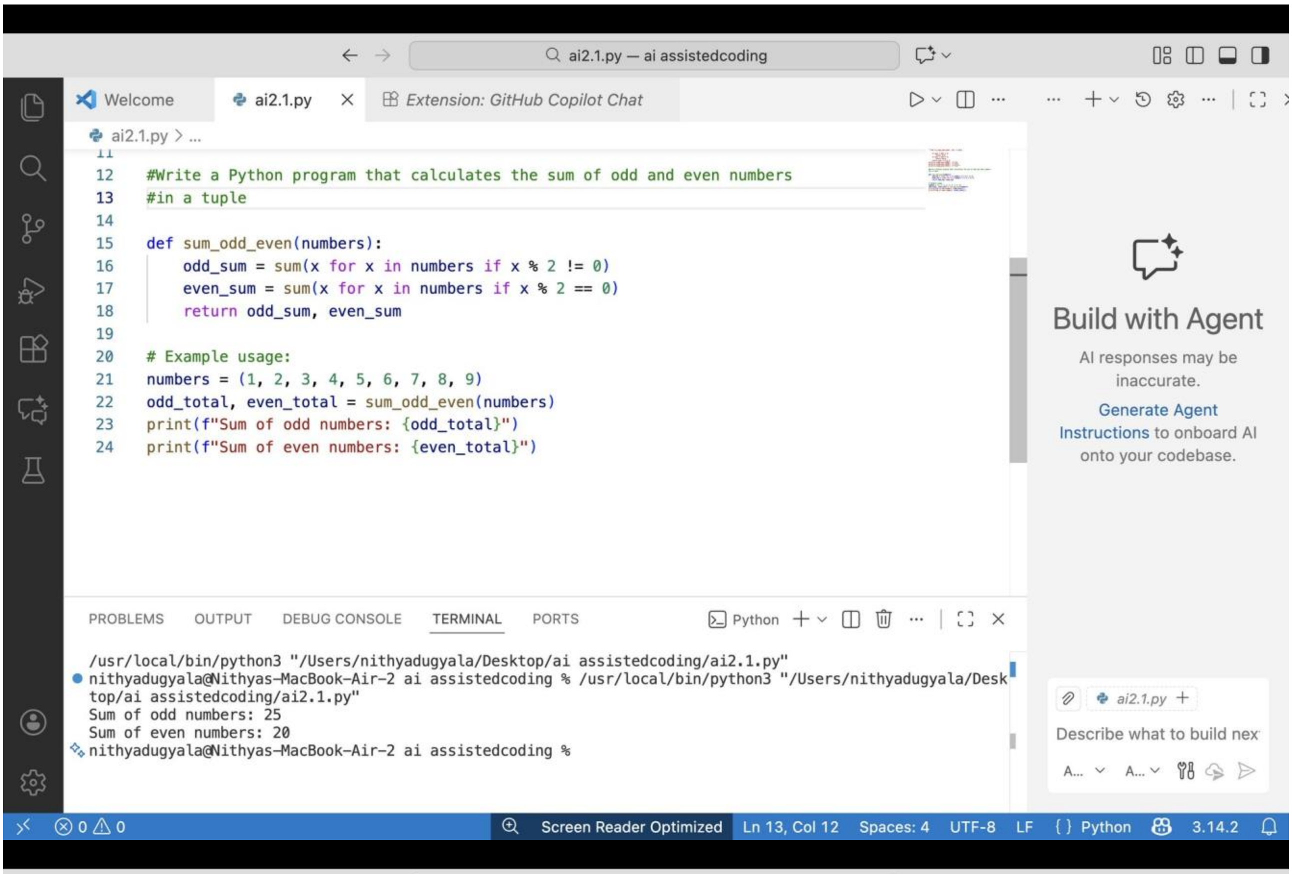
ai2.1.py Lines 1-7

```python
def is_leap_year(year: int) -> bool:
    """Return True if `year` is a leap year, else F
    if year % 400 == 0:
        return True
    if year % 100 == 0:
        return False
    return year % 4 == 0
```

To use it interactively, run `python ai2.1.py` and adjust or add `print` lines for other years.

```python
def is_leap_year(year: int) -> bool:
    """Return True if `year` is a leap year, else False."""
    if year % 400 == 0:
        return True
    if year % 100 == 0:
        return False
    return year % 4 == 0
print(is_leap_year(2000))  # True
print(is_leap_year(1900))  # False
print(is_leap_year(2024))  # True
```

```
/usr/local/bin/python3 "/Users/nithyadugyala/Desktop/ai assistedcoding/ai2.1.py"
● nithyadugyala@Nithyas-MacBook-Air-2 ai assistedcoding % /usr/local/bin/python3 "/Users/nithyadugyala/Desktop/ai assisted
coding/ai2.1.py"
● nithyadugyala@Nithyas-MacBook-Air-2 ai assistedcoding % /usr/local/bin/python3 "/Users/nithyadugyala/Desktop/ai assisted
coding/ai2.1.py"
True
False
True
○ nithyadugyala@Nithyas-MacBook-Air-2 ai assistedcoding % █
```



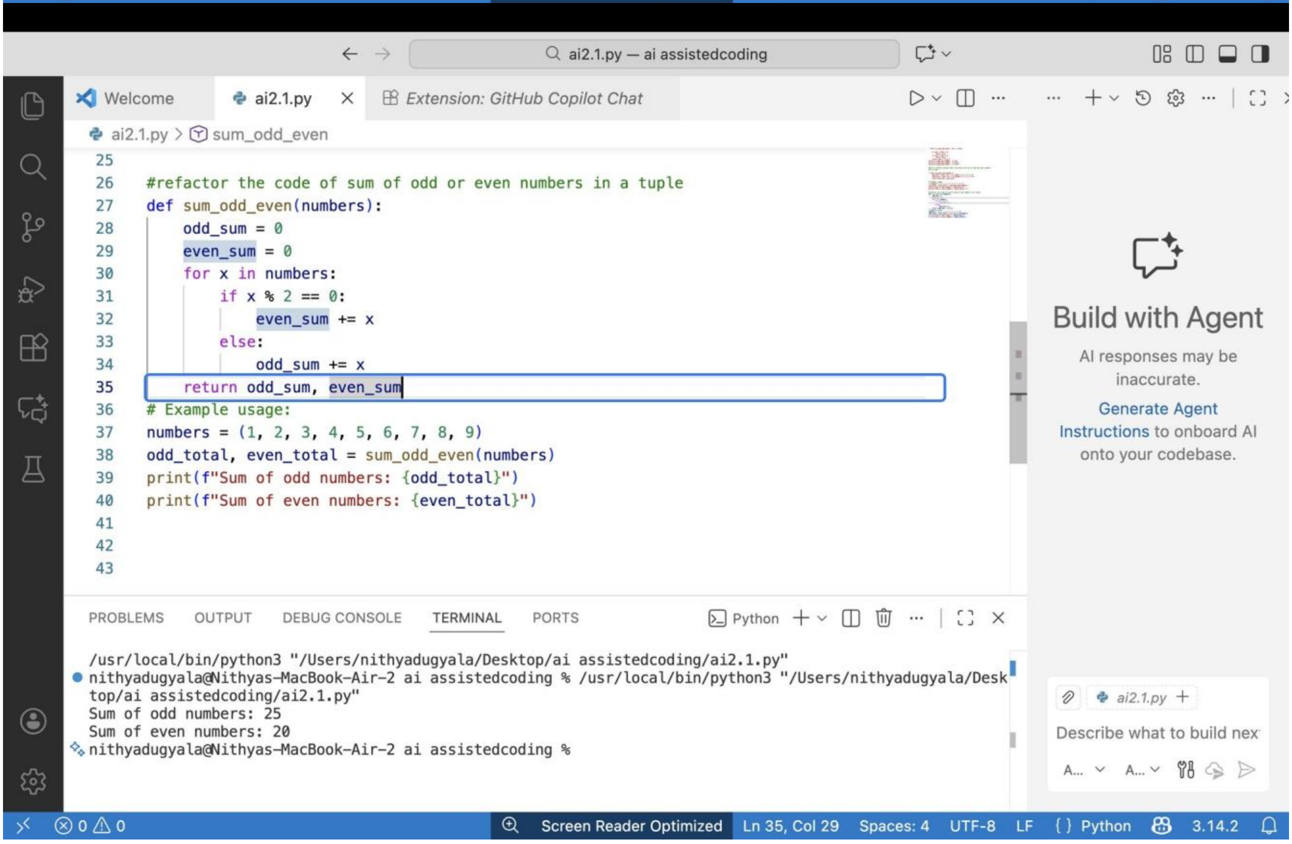
AI responses may be inaccurate.

Generate Agent Instructions to onboard AI onto your codebase.

Describe what to build nex

```python
#Write a Python program that calculates the sum of odd and even numbers
#in a tuple

def sum_odd_even(numbers):
    odd_sum = sum(x for x in numbers if x % 2 != 0)
    even_sum = sum(x for x in numbers if x % 2 == 0)
    return odd_sum, even_sum

# Example usage:
numbers = (1, 2, 3, 4, 5, 6, 7, 8, 9)
odd_total, even_total = sum_odd_even(numbers)
print(f"Sum of odd numbers: {odd_total}")
print(f"Sum of even numbers: {even_total}")
```

```
/usr/local/bin/python3 "/Users/nithyadugyala/Desktop/ai assistedcoding/ai2.1.py"
nithyadugyala@Nithyas-MacBook-Air-2 ai assistedcoding % /usr/local/bin/python3 "/Users/nithyadugyala/Desk
top/ai assistedcoding/ai2.1.py"
Sum of odd numbers: 25
Sum of even numbers: 20
nithyadugyala@Nithyas-MacBook-Air-2 ai assistedcoding %
```



AI responses may be inaccurate.

Generate Agent Instructions to onboard AI onto your codebase.

Describe what to build nex

```python

#refactor the code of sum of odd or even numbers in a tuple
def sum_odd_even(numbers):
    odd_sum = 0
    even_sum = 0
    for x in numbers:
        if x % 2 == 0:
            even_sum += x
        else:
            odd_sum += x
    return odd_sum, even_sum
# Example usage:
numbers = (1, 2, 3, 4, 5, 6, 7, 8, 9)
odd_total, even_total = sum_odd_even(numbers)
print(f"Sum of odd numbers: {odd_total}")
print(f"Sum of even numbers: {even_total}")



```

```
/usr/local/bin/python3 "/Users/nithyadugyala/Desktop/ai assistedcoding/ai2.1.py"
nithyadugyala@Nithyas-MacBook-Air-2 ai assistedcoding % /usr/local/bin/python3 "/Users/nithyadugyala/Desk
top/ai assistedcoding/ai2.1.py"
Sum of odd numbers: 25
Sum of even numbers: 20
nithyadugyala@Nithyas-MacBook-Air-2 ai assistedcoding %
```