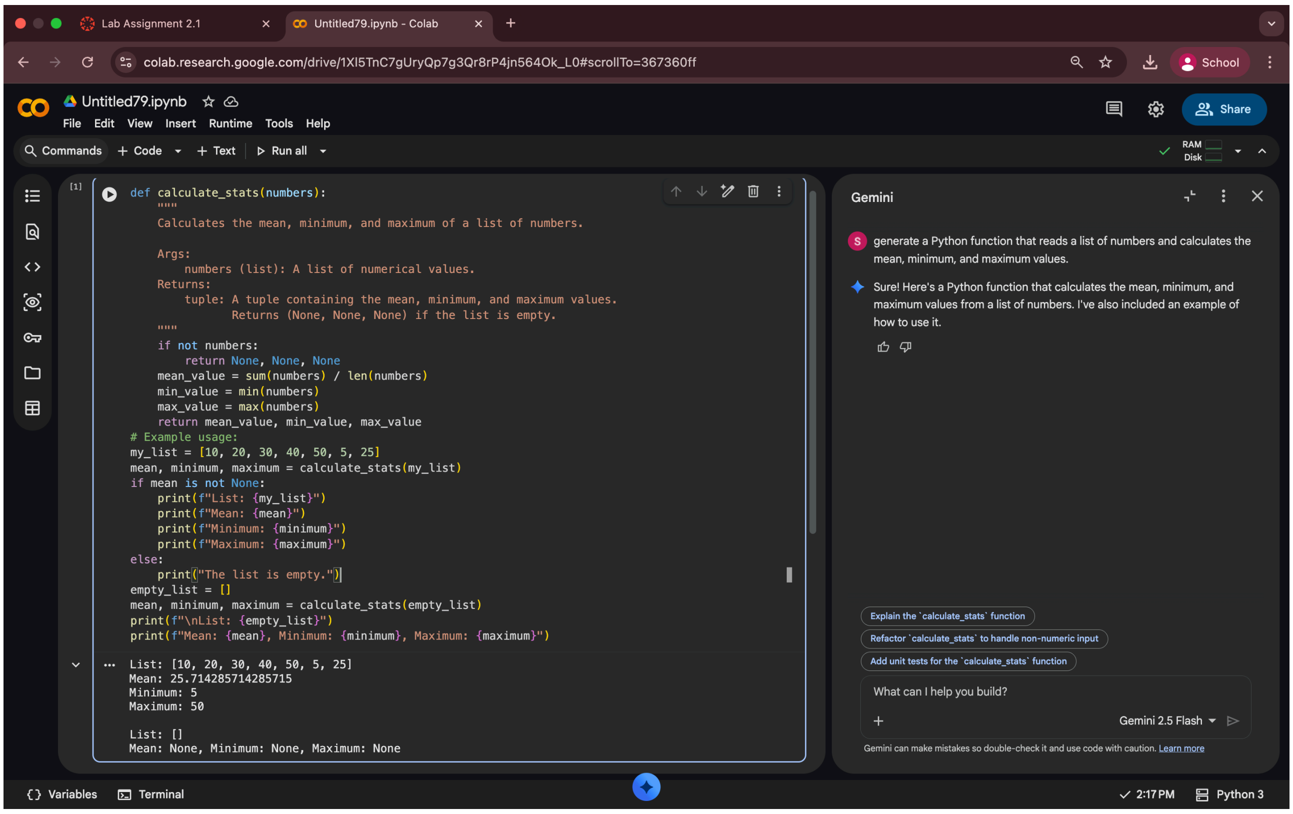
AI-assisted coding
2303A510D7(V.Sathwik Reddy)
Assignment-2.1

Task 1:
Use Google Gemini in Colab to generate a Python function that reads a list of numbers and calculates the mean, minimum, and maximum values.


Lab Assignment 2.1
Untitled79.ipynb - Colab
colab.research.google.com/drive/1Xl5TnC7gUryQp7g3Qr8rP4jn564Ok_L0#scrollTo=367360ff
School
Untitled79.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text Run all
Share
RAM Disk
Gemini
Explain the `calculate_stats` function
Refactor `calculate_stats` to handle non-numeric input
Add unit tests for the `calculate_stats` function
What can I help you build?
Gemini 2.5 Flash
Variables Terminal
2:17 PM Python 3

```
[1]
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 the mean, minimum, and maximum values.
               Returns (None, None, None) if the list is empty.
    """
    if not numbers:
        return None, None, None
    mean_value = sum(numbers) / len(numbers)
    min_value = min(numbers)
    max_value = max(numbers)
    return mean_value, min_value, max_value
# Example usage:
my_list = [10, 20, 30, 40, 50, 5, 25]
mean, minimum, maximum = calculate_stats(my_list)
if mean is not None:
    print(f"List: {my_list}")
    print(f"Mean: {mean}")
    print(f"Minimum: {minimum}")
    print(f"Maximum: {maximum}")
else:
    print("The list is empty.")
empty_list = []
mean, minimum, maximum = calculate_stats(empty_list)
print(f"\nList: {empty_list}")
print(f"Mean: {mean}, Minimum: {minimum}, Maximum: {maximum}")
```

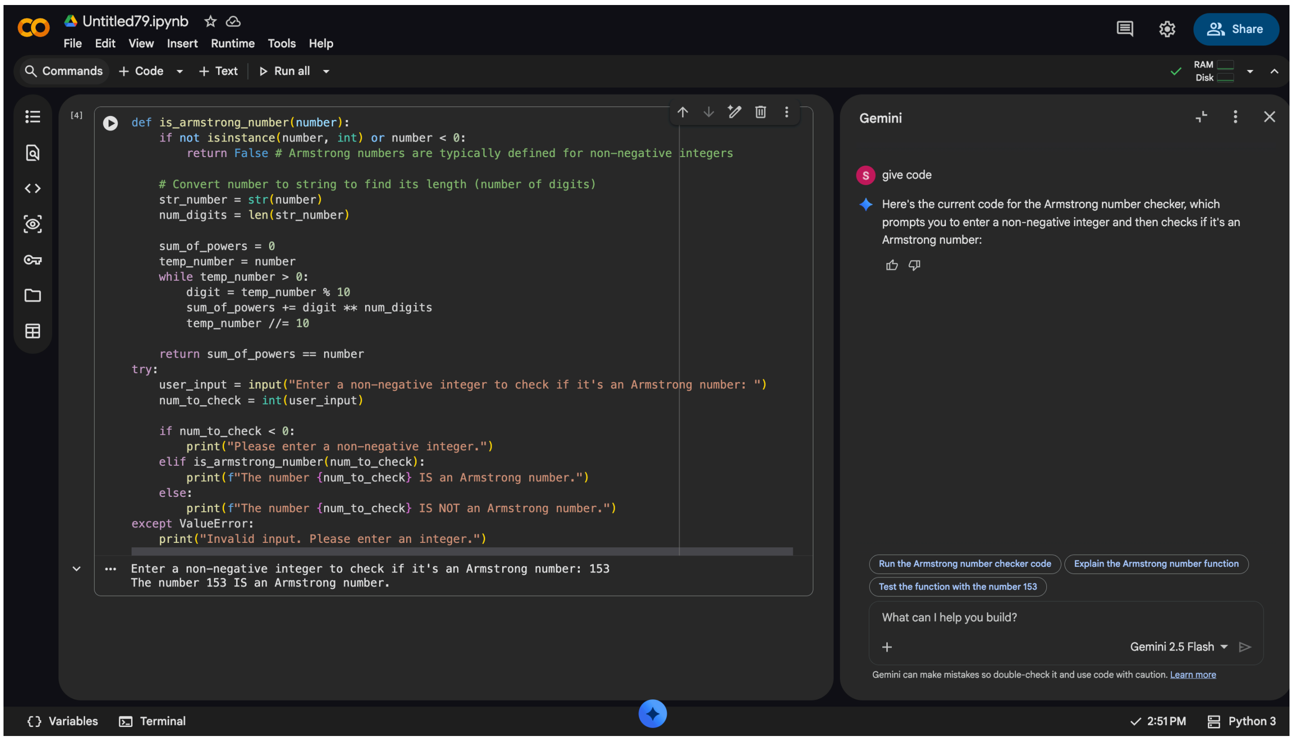
```
List: [10, 20, 30, 40, 50, 5, 25]
Mean: 25.714285714285715
Minimum: 5
Maximum: 50

List: []
Mean: None, Minimum: None, Maximum: None
```

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

Sure! Here's a Python function that calculates the mean, minimum, and maximum values from a list of numbers. 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

Task 2 :
Generate an Armstrong number checker using Gemini and GitHub Copilot.


Untitled79.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text Run all
Share
RAM Disk
Gemini
Run the Armstrong number checker code
Explain the Armstrong number function
Test the function with the number 153
What can I help you build?
Gemini 2.5 Flash
Variables Terminal
2:51 PM Python 3

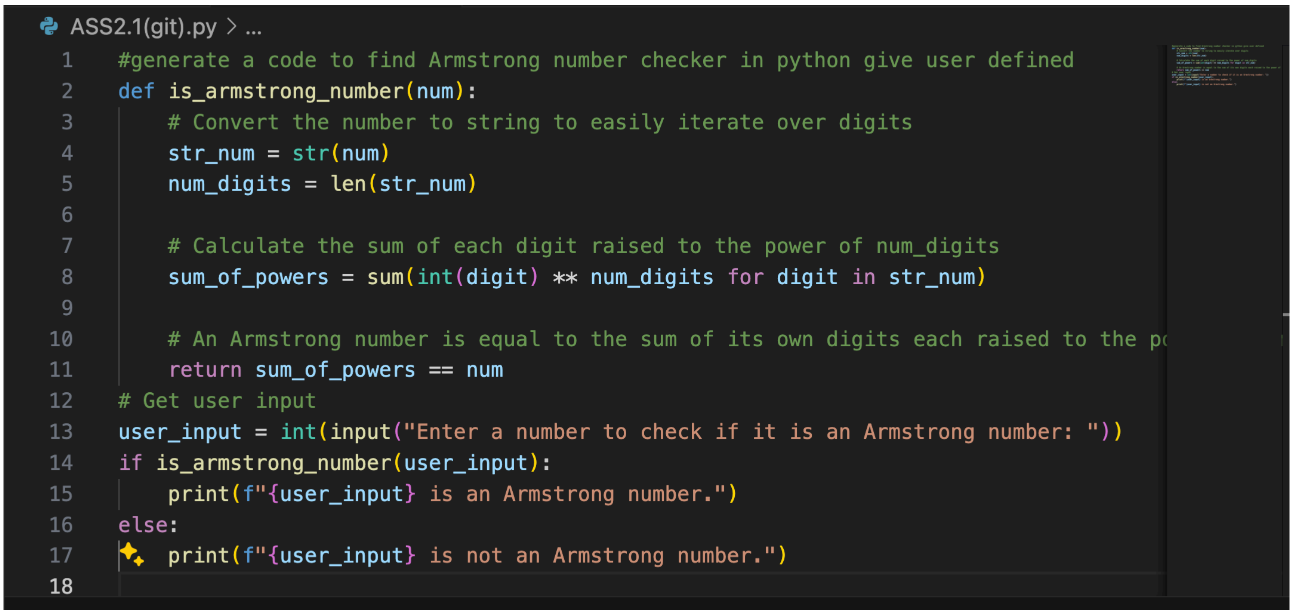
```
[4]
def is_armstrong_number(number):
    if not isinstance(number, int) or number < 0:
        return False # Armstrong numbers are typically defined for non-negative integers

    # Convert number to string to find its length (number of digits)
    str_number = str(number)
    num_digits = len(str_number)

    sum_of_powers = 0
    temp_number = number
    while temp_number > 0:
        digit = temp_number % 10
        sum_of_powers += digit ** num_digits
        temp_number //= 10

    return sum_of_powers == number
try:
    user_input = input("Enter a non-negative integer to check if it's an Armstrong number: ")
    num_to_check = int(user_input)

    if num_to_check < 0:
        print("Please enter a non-negative integer.")
    elif is_armstrong_number(num_to_check):
        print(f"The number {num_to_check} IS an Armstrong number.")
    else:
        print(f"The number {num_to_check} IS NOT an Armstrong number.")
except ValueError:
    print("Invalid input. Please enter an integer.")
```

```
Enter a non-negative integer to check if it's an Armstrong number: 153
The number 153 IS an Armstrong number.
```

S give code

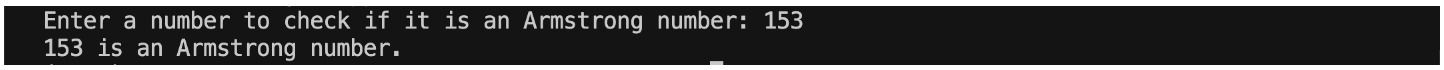
Here's the current code for the Armstrong number checker, which prompts you to enter a non-negative integer and then checks if it's an Armstrong number:

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



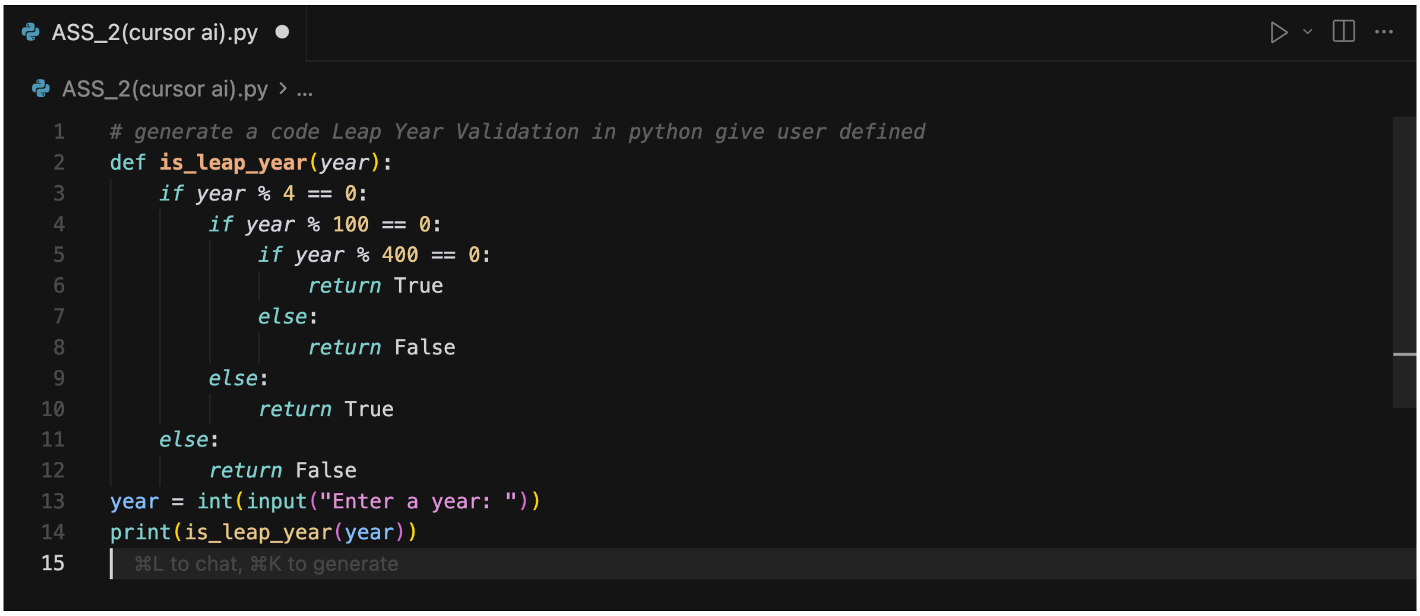
```python
#generate a code to find Armstrong number checker in python give user defined
def is_armstrong_number(num):
    # Convert the number to string to easily iterate over digits
    str_num = str(num)
    num_digits = len(str_num)

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

    # An Armstrong number is equal to the sum of its own digits each raised to the p
    return sum_of_powers == num
# Get user input
user_input = int(input("Enter a number to check if it is an Armstrong number: "))
if is_armstrong_number(user_input):
    print(f"{user_input} is an Armstrong number.")
else:
    print(f"{user_input} is not an Armstrong number.")
```

Output:



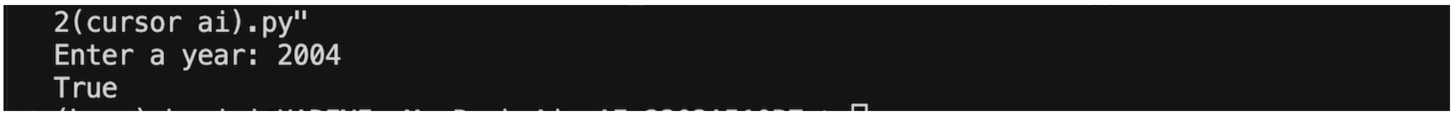
```
Enter a number to check if it is an Armstrong number: 153
153 is an Armstrong number.
```

Task 3 :
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.

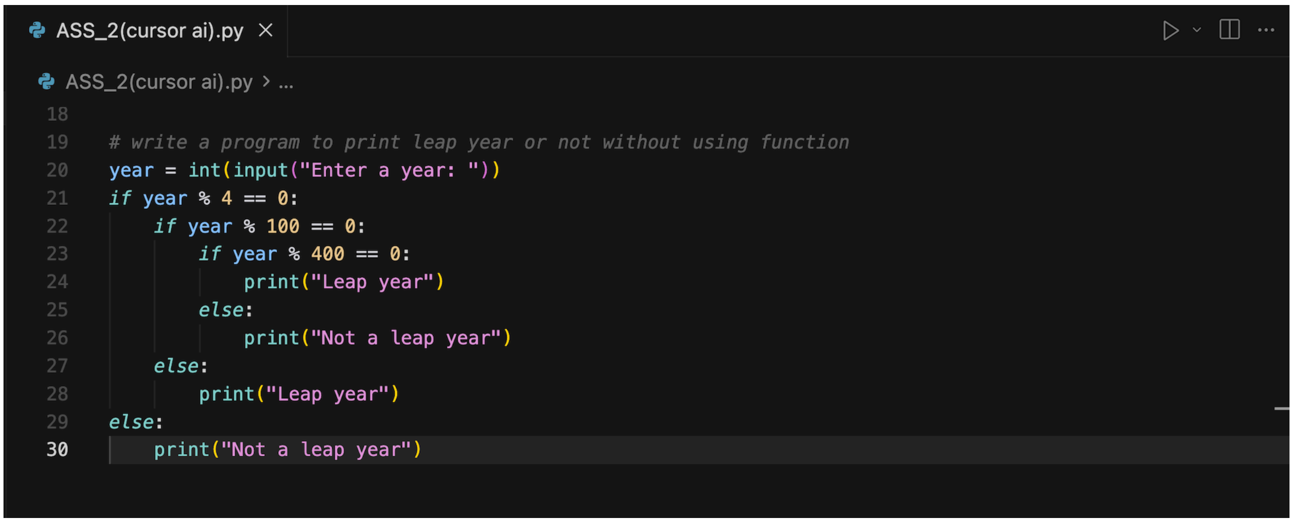


```python
# generate a code Leap Year Validation in python give user defined
def is_leap_year(year):
    if year % 4 == 0:
        if year % 100 == 0:
            if year % 400 == 0:
                return True
            else:
                return False
        else:
            return True
    else:
        return False
year = int(input("Enter a year: "))
print(is_leap_year(year))
```

Output:



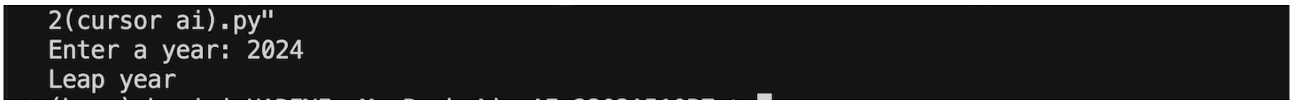
```
2(cursor ai).py"
Enter a year: 2004
True
```



ASS_2(cursor ai).py

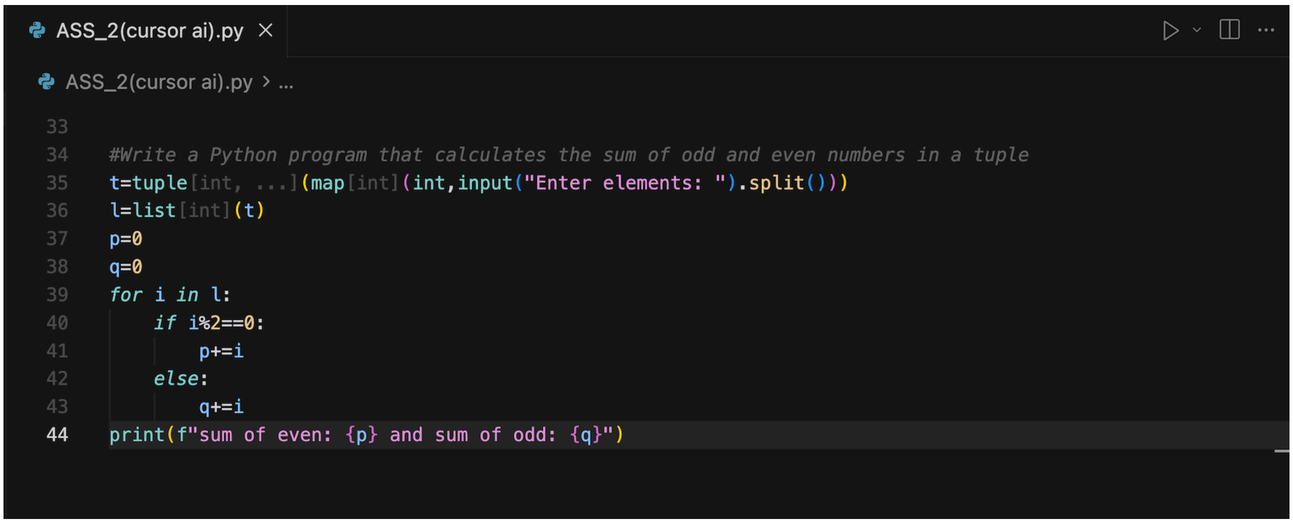
```python

# write a program to print leap year or not without using function
year = int(input("Enter a year: "))
if year % 4 == 0:
    if year % 100 == 0:
        if year % 400 == 0:
            print("Leap year")
        else:
            print("Not a leap year")
    else:
        print("Leap year")
else:
    print("Not a leap year")
```

Output:



```
2(cursor ai).py"
Enter a year: 2024
Leap year
```

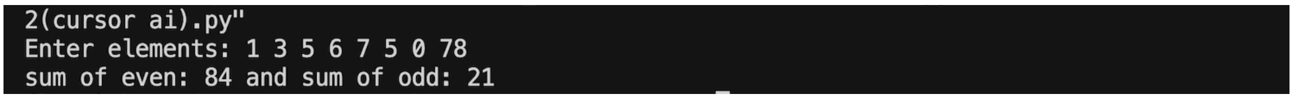
Task 4 :
Write a Python program that calculates the sum of odd and even numbers in a tuple, then refactor it using any AI tool.



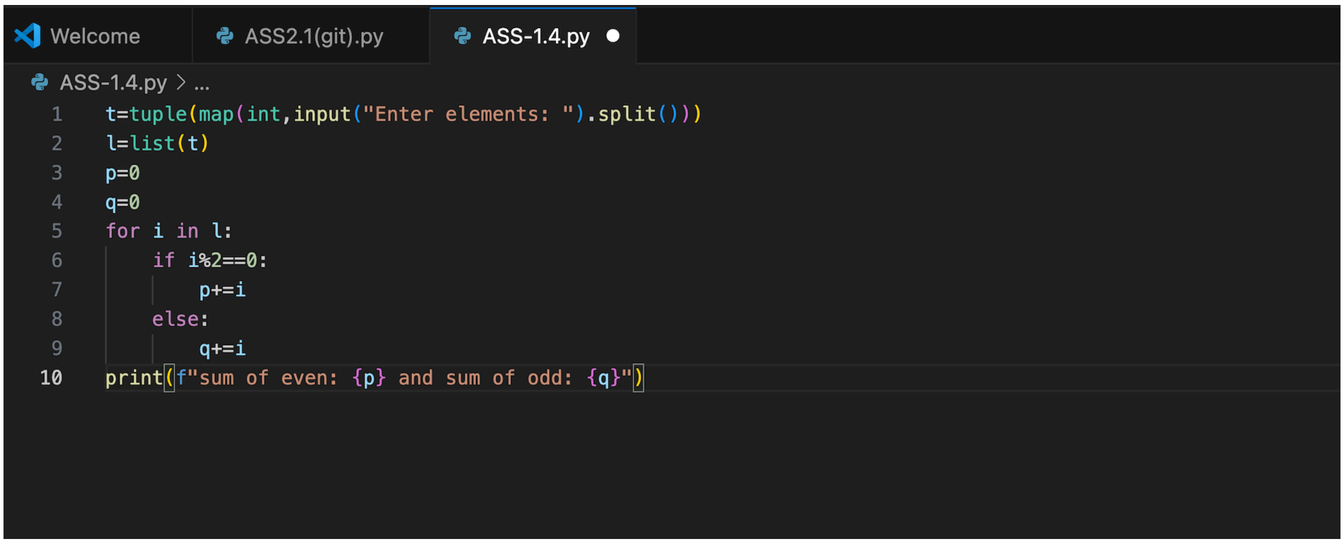
ASS_2(cursor ai).py

```python

#Write a Python program that calculates the sum of odd and even numbers in a tuple
t=tuple[int, ...](map[int](int,input("Enter elements: ").split()))
l=list[int](t)
p=0
q=0
for i in l:
    if i%2==0:
        p+=i
    else:
        q+=i
print(f"sum of even: {p} and sum of odd: {q}")
```

Output:



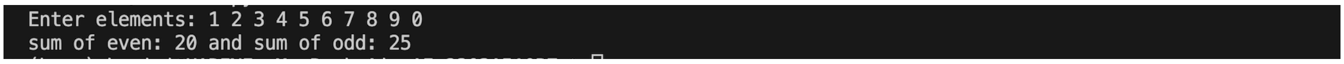
```
2(cursor ai).py"
Enter elements: 1 3 5 6 7 5 0 78
sum of even: 84 and sum of odd: 21
```



Welcome | ASS2.1(git).py | ASS-1.4.py

ASS-1.4.py > ...

```python
t=tuple(map(int,input("Enter elements: ").split()))
l=list(t)
p=0
q=0
for i in l:
    if i%2==0:
        p+=i
    else:
        q+=i
print(f"sum of even: {p} and sum of odd: {q}")
```



```
Enter elements: 1 2 3 4 5 6 7 8 9 0
sum of even: 20 and sum of odd: 25
```