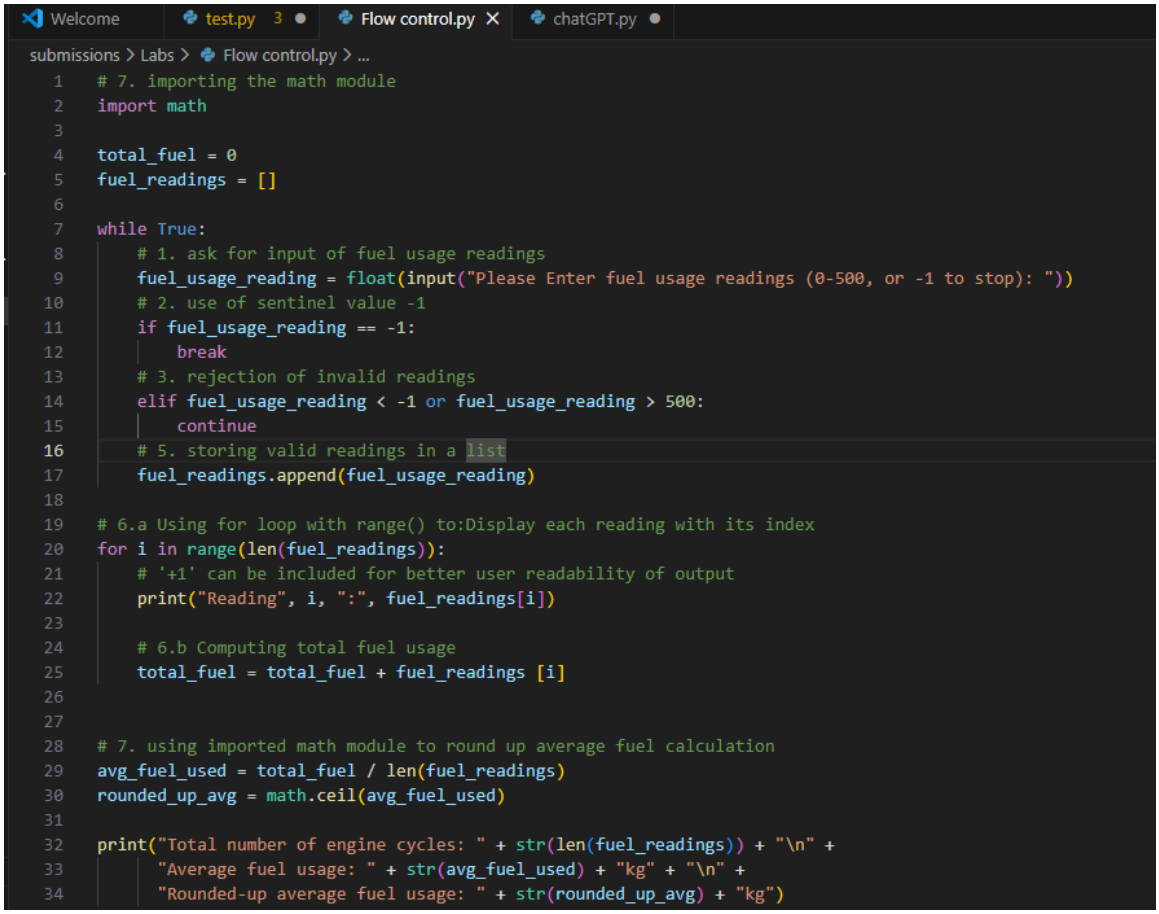
Comparison Report

ChatGPT:

```python
import math
fuel_readings = []

while True:
    reading = float(input("Enter fuel usage in kg (-1 to stop): "))

    if reading == -1:
        break

    if reading < 0 or reading > 500:
        print("Invalid reading. Must be between 0 and 500 kg.")
        continue
    fuel_readings.append(reading)

# Display readings with index and compute total
total_fuel = 0

for i in range(len(fuel_readings)):
    print(f"Cycle {i}: {fuel_readings[i]} kg")
    total_fuel += fuel_readings[i]

# Calculations
num_cycles = len(fuel_readings)

if num_cycles > 0:
    average = total_fuel / num_cycles
    rounded_avg = math.ceil(average)
else:
    average = 0
    rounded_avg = 0
# Output results
print("\nSummary")
print("Number of engine cycles:", num_cycles)
print("Average fuel usage:", average)
print("Rounded-up average fuel usage:", rounded_avg)
```

Jones':



Both programs satisfy the problem requirements and produce essentially the same outputs: collect fuel readings using a while loop with a sentinel value (-1), storing readings in a list, displaying each reading with its index using a for loop with range(), computing total fuel usage, and calculating both the average and rounded-up average fuel consumption using the math module. In addition, both approaches correctly apply **break** to terminate input and c**ontinue** to skip invalid readings.

Key differences were in structure, assumptions, and robustness. ChatGPT's solution used float inputs (with my initial source code giving errors since the input variable was set to int) and included a safeguard against division by zero when no valid readings are entered, making it more fault-tolerant. In contrast, my initial implementation assumed at least one valid reading and used integer input, which was simpler but not very tolerable (which led to the modification of the input variable to float). ChatGPT used less comments, which is a practice that would affect readability. Overall, both solutions yielded the expected results when tested. Furthermore, I was uncertain at some point on whether readings were to represent fuel used or remaining fuel (opted for the former). Additionally, displaying "Reading 0" to users reduces readability; a more user-friendly approach would start indexing from 1. To conclude the key difference between this program and real world scenarios aside from the complexity, is the manual entry of readings, as the real world programs are automated.