

Comparison Report

ChatGPT:

```
1  import math
2  fuel_readings = []
3
4  while True:
5      reading = float(input("Enter fuel usage in kg (-1 to stop): "))
6
7      if reading == -1:
8          break
9
10     if reading < 0 or reading > 500:
11         print("Invalid reading. Must be between 0 and 500 kg.")
12         continue
13     fuel_readings.append(reading)
14
15     # Display readings with index and compute total
16     total_fuel = 0
17
18     for i in range(len(fuel_readings)):
19         print(f"Cycle {i}: {fuel_readings[i]} kg")
20         total_fuel += fuel_readings[i]
21
22     # Calculations
23     num_cycles = len(fuel_readings)
24
25     if num_cycles > 0:
26         average = total_fuel / num_cycles
27         rounded_avg = math.ceil(average)
28     else:
29         average = 0
30         rounded_avg = 0
31
32     # Output results
33     print("\nSummary")
34     print("Number of engine cycles:", num_cycles)
35     print("Average fuel usage:", average)
36     print("Rounded-up average fuel usage:", rounded_avg)
```

Jones':

```
Welcome | test.py 3 | Flow control.py X | chatGPT.py •
submissions > Labs > Flow control.py > ...
1  # 7. importing the math module
2  import math
3
4  total_fuel = 0
5  fuel_readings = []
6
7  while True:
8      # 1. ask for input of fuel usage readings
9      fuel_usage_reading = float(input("Please Enter fuel usage readings (0-500, or -1 to stop): "))
10     # 2. use of sentinel value -1
11     if fuel_usage_reading == -1:
12         break
13     # 3. rejection of invalid readings
14     elif fuel_usage_reading < -1 or fuel_usage_reading > 500:
15         continue
16     # 5. storing valid readings in a list
17     fuel_readings.append(fuel_usage_reading)
18
19     # 6.a Using for loop with range() to:Display each reading with its index
20     for i in range(len(fuel_readings)):
21         # '+1' can be included for better user readability of output
22         print("Reading", i, ":", fuel_readings[i])
23
24         # 6.b Computing total fuel usage
25         total_fuel = total_fuel + fuel_readings[i]
26
27
28     # 7. using imported math module to round up average fuel calculation
29     avg_fuel_used = total_fuel / len(fuel_readings)
30     rounded_up_avg = math.ceil(avg_fuel_used)
31
32     print("Total number of engine cycles: " + str(len(fuel_readings)) + "\n" +
33           "Average fuel usage: " + str(avg_fuel_used) + "kg" + "\n" +
34           "Rounded-up average fuel usage: " + str(rounded_up_avg) + "kg")
```

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 **continue** 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.