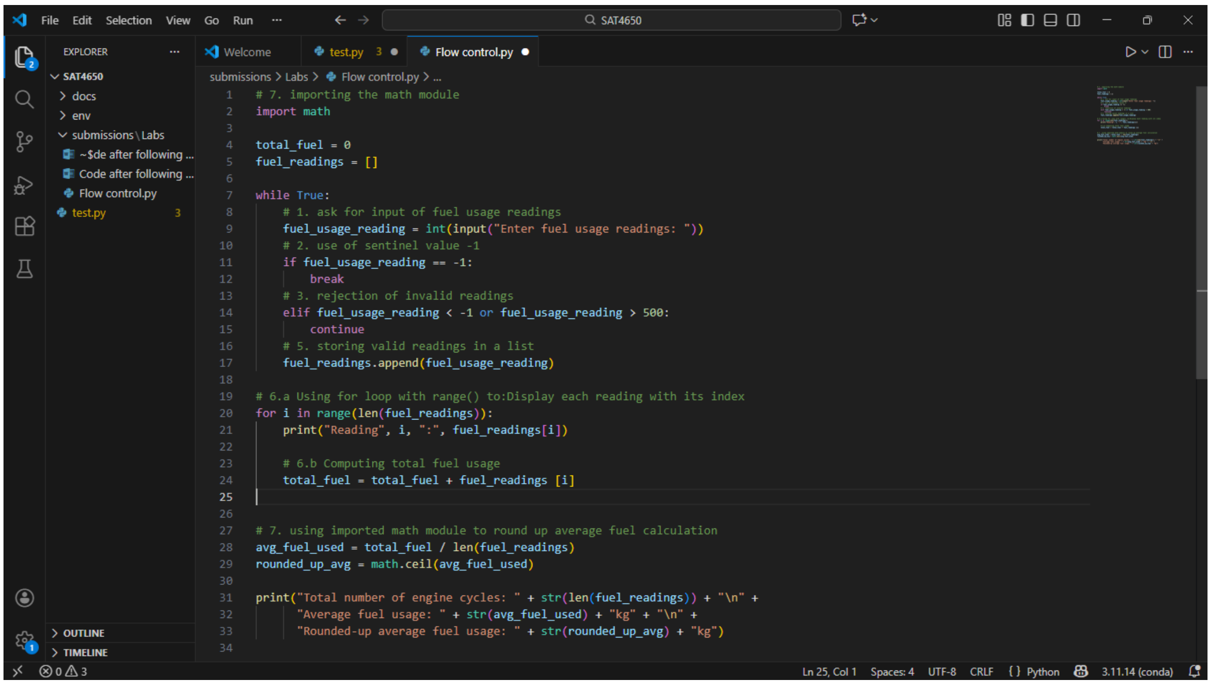
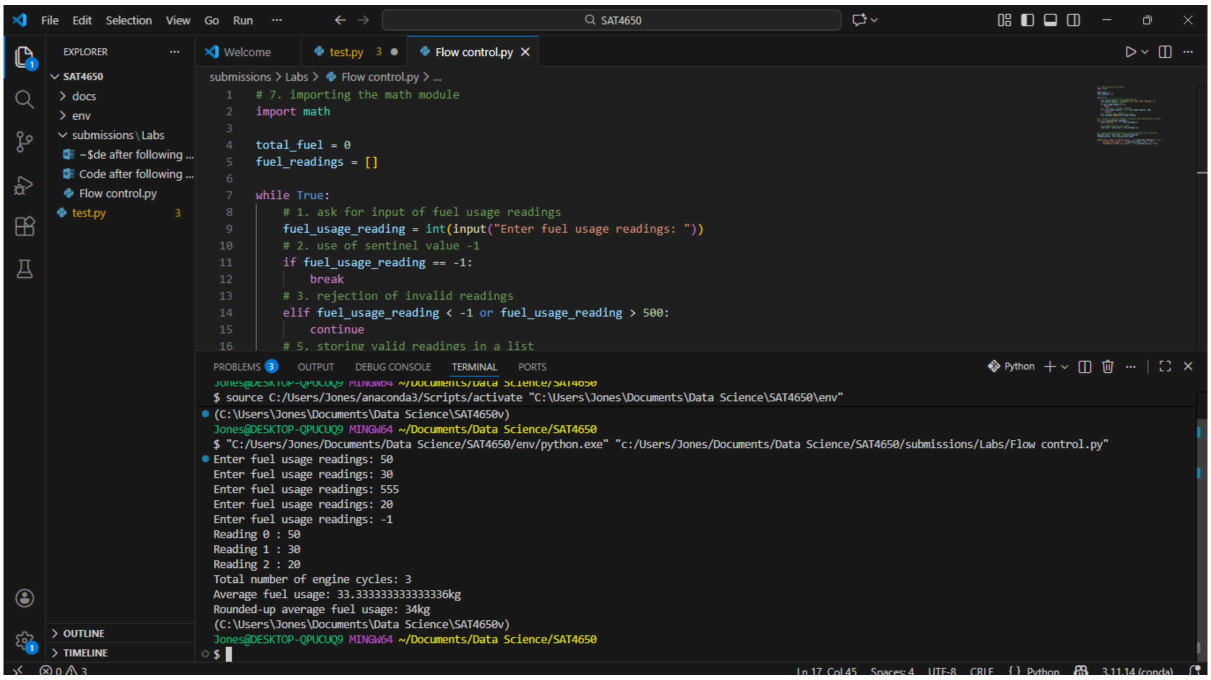
Screenshots

1. Code after answering questions (before first trial run):



```python
# 7. importing the math module
import math

total_fuel = 0
fuel_readings = []

while True:
    # 1. ask for input of fuel usage readings
    fuel_usage_reading = int(input("Enter fuel usage readings: "))
    # 2. use of sentinel value -1
    if fuel_usage_reading == -1:
        break
    # 3. rejection of invalid readings
    elif fuel_usage_reading < -1 or fuel_usage_reading > 500:
        continue
    # 5. storing valid readings in a list
    fuel_readings.append(fuel_usage_reading)

# 6.a Using for loop with range() to:Display each reading with its index
for i in range(len(fuel_readings)):
    print("Reading", i, ":", fuel_readings[i])

    # 6.b Computing total fuel usage
    total_fuel = total_fuel + fuel_readings [i]


# 7. using imported math module to round up average fuel calculation
avg_fuel_used = total_fuel / len(fuel_readings)
rounded_up_avg = math.ceil(avg_fuel_used)

print("Total number of engine cycles: " + str(len(fuel_readings)) + "\n" +
      "Average fuel usage: " + str(avg_fuel_used) + "kg" + "\n" +
      "Rounded-up average fuel usage: " + str(rounded_up_avg) + "kg")

```

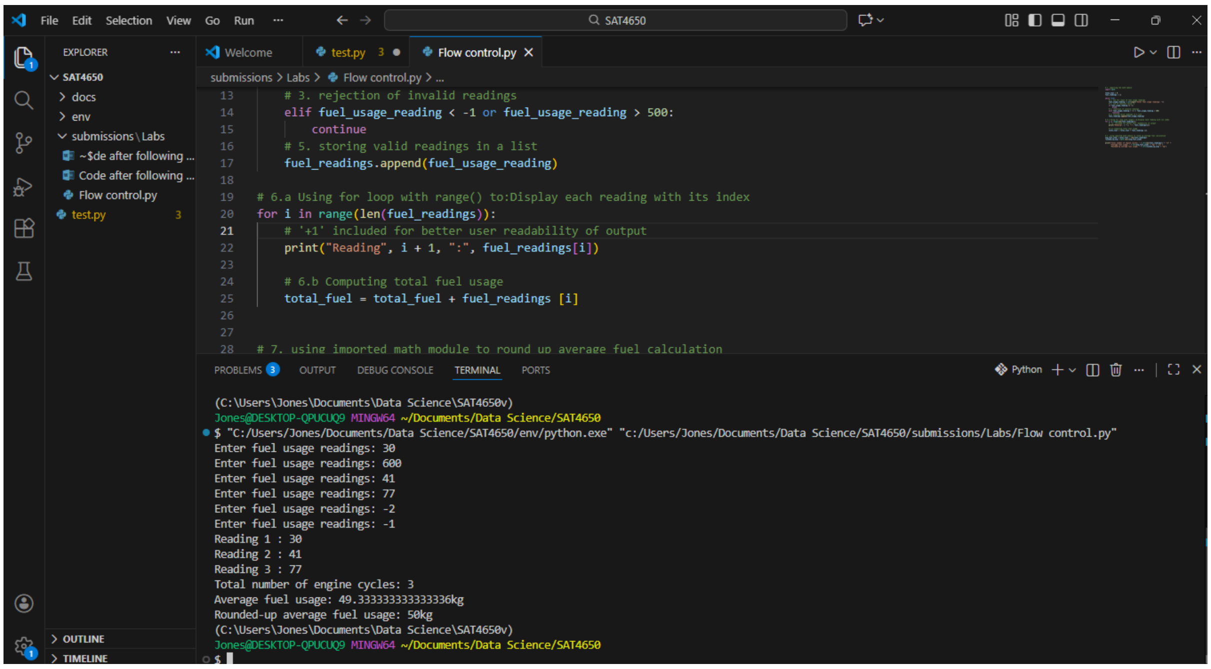
2. Code with results upon first trial of inputting sample readings (50, 30, 555, 20, -1):



```python
# 7. importing the math module
import math

total_fuel = 0
fuel_readings = []

while True:
    # 1. ask for input of fuel usage readings
    fuel_usage_reading = int(input("Enter fuel usage readings: "))
    # 2. use of sentinel value -1
    if fuel_usage_reading == -1:
        break
    # 3. rejection of invalid readings
    elif fuel_usage_reading < -1 or fuel_usage_reading > 500:
        continue
    # 5. storing valid readings in a list
```

```
Jones@DESKTOP-QPUCUQ9 MINGW64 ~/Documents/Data Science/SAT4650
$ source C:/Users/Jones/anaconda3/Scripts/activate "C:\Users\Jones\Documents\Data Science\SAT4650\env"
(C:\Users\Jones\Documents\Data Science\SAT4650v)
Jones@DESKTOP-QPUCUQ9 MINGW64 ~/Documents/Data Science/SAT4650
$ "C:/Users/Jones/Documents/Data Science/SAT4650/env/python.exe" "c:/Users/Jones/Documents/Data Science/SAT4650/submissions/Labs/Flow control.py"
Enter fuel usage readings: 50
Enter fuel usage readings: 30
Enter fuel usage readings: 555
Enter fuel usage readings: 20
Enter fuel usage readings: -1
Reading 0 : 50
Reading 1 : 30
Reading 2 : 20
Total number of engine cycles: 3
Average fuel usage: 33.333333333333336kg
Rounded-up average fuel usage: 34kg
(C:\Users\Jones\Documents\Data Science\SAT4650v)
Jones@DESKTOP-QPUCUQ9 MINGW64 ~/Documents/Data Science/SAT4650
$ 
```

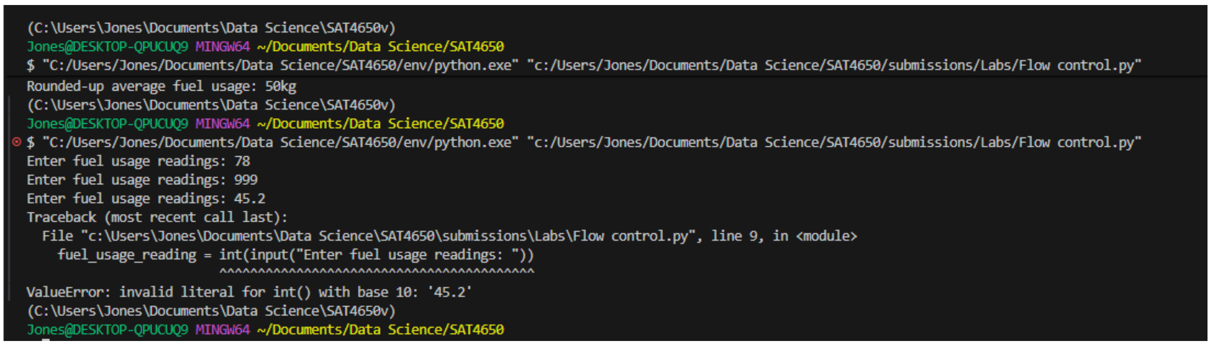
3. Test results after enhancing user readability:



```python
    # 3. rejection of invalid readings
    elif fuel_usage_reading < -1 or fuel_usage_reading > 500:
        continue
    # 5. storing valid readings in a list
    fuel_readings.append(fuel_usage_reading)

# 6.a Using for loop with range() to:Display each reading with its index
for i in range(len(fuel_readings)):
    # '+1' included for better user readability of output
    print("Reading", i + 1, ":", fuel_readings[i])

    # 6.b Computing total fuel usage
    total_fuel = total_fuel + fuel_readings [i]


# 7. using imported math module to round up average fuel calculation
```

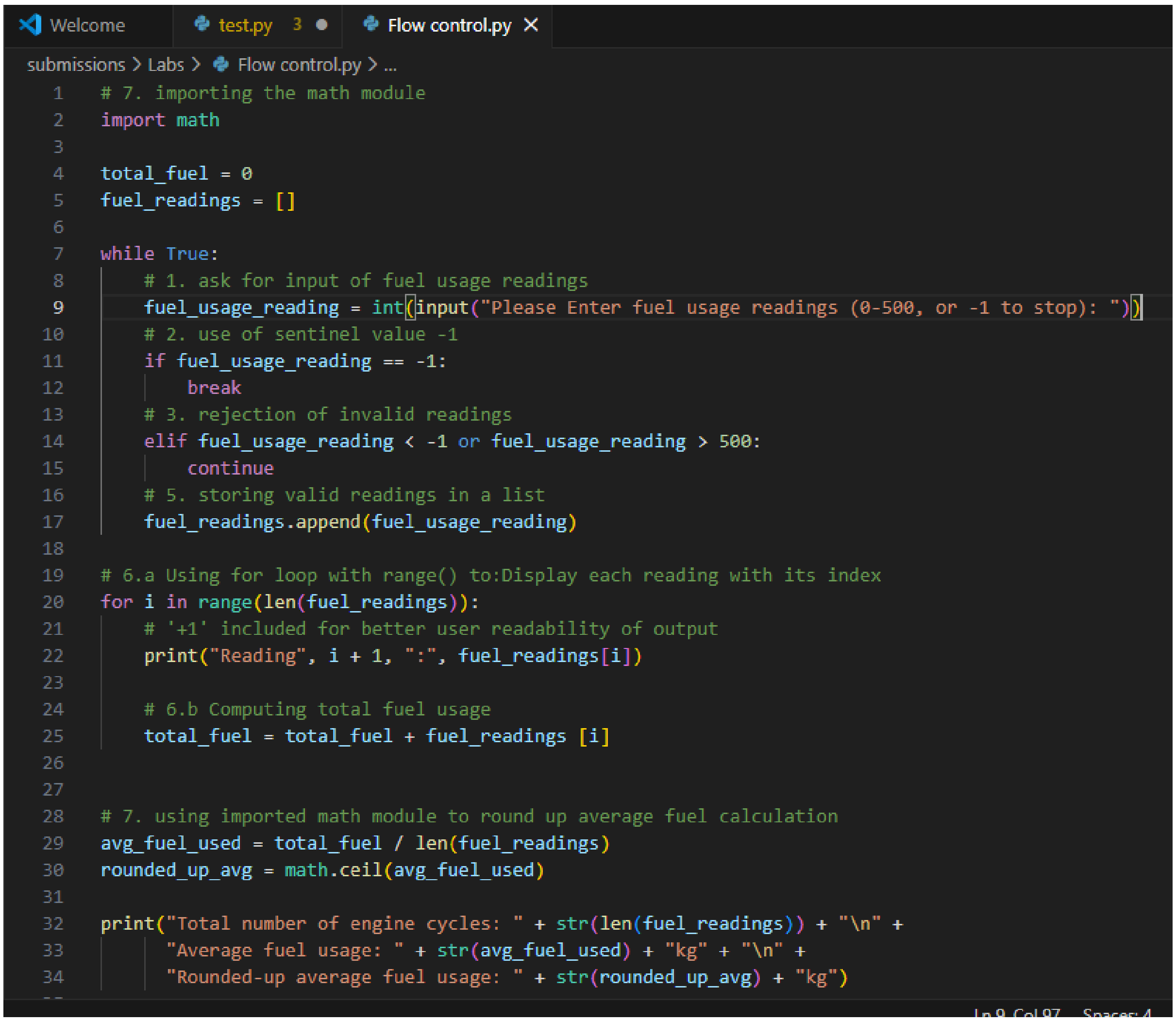
```
(C:\Users\Jones\Documents\Data Science\SAT4650v)
Jones@DESKTOP-QPUCUQ9 MINGW64 ~/Documents/Data Science/SAT4650
$ "C:/Users/Jones/Documents/Data Science/SAT4650/env/python.exe" "c:/Users/Jones/Documents/Data Science/SAT4650/submissions/Labs/Flow control.py"
Enter fuel usage readings: 30
Enter fuel usage readings: 600
Enter fuel usage readings: 41
Enter fuel usage readings: 77
Enter fuel usage readings: -2
Enter fuel usage readings: -1
Reading 1 : 30
Reading 2 : 41
Reading 3 : 77
Total number of engine cycles: 3
Average fuel usage: 49.333333333333336kg
Rounded-up average fuel usage: 50kg
(C:\Users\Jones\Documents\Data Science\SAT4650v)
Jones@DESKTOP-QPUCUQ9 MINGW64 ~/Documents/Data Science/SAT4650
$
```

4. Error after entering decimal value:



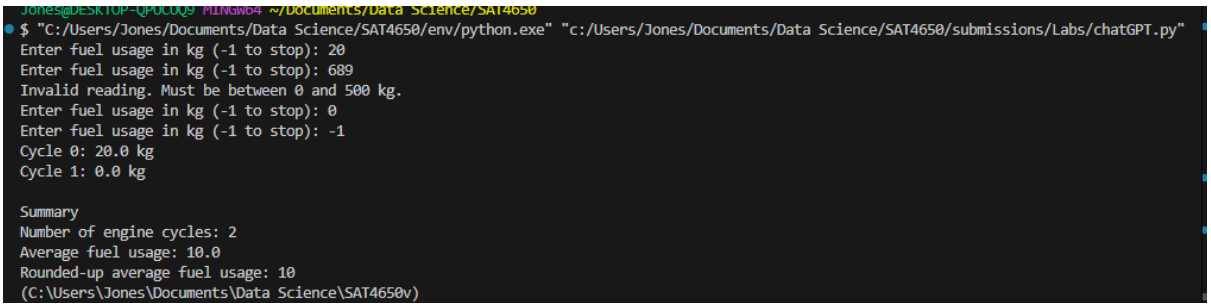
```
(C:\Users\Jones\Documents\Data Science\SAT4650v)
Jones@DESKTOP-QPUCUQ9 MINGW64 ~/Documents/Data Science/SAT4650
$ "C:/Users/Jones/Documents/Data Science/SAT4650/env/python.exe" "c:/Users/Jones/Documents/Data Science/SAT4650/submissions/Labs/Flow control.py"
Rounded-up average fuel usage: 50kg
(C:\Users\Jones\Documents\Data Science\SAT4650v)
Jones@DESKTOP-QPUCUQ9 MINGW64 ~/Documents/Data Science/SAT4650
$ "C:/Users/Jones/Documents/Data Science/SAT4650/env/python.exe" "c:/Users/Jones/Documents/Data Science/SAT4650/submissions/Labs/Flow control.py"
Enter fuel usage readings: 78
Enter fuel usage readings: 999
Enter fuel usage readings: 45.2
Traceback (most recent call last):
  File "c:\Users\Jones\Documents\Data Science\SAT4650\submissions\Labs\Flow control.py", line 9, in <module>
    fuel_usage_reading = int(input("Enter fuel usage readings: "))
                         ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
ValueError: invalid literal for int() with base 10: '45.2'
(C:\Users\Jones\Documents\Data Science\SAT4650v)
Jones@DESKTOP-QPUCUQ9 MINGW64 ~/Documents/Data Science/SAT4650
```

5. source code enhanced for user readability :



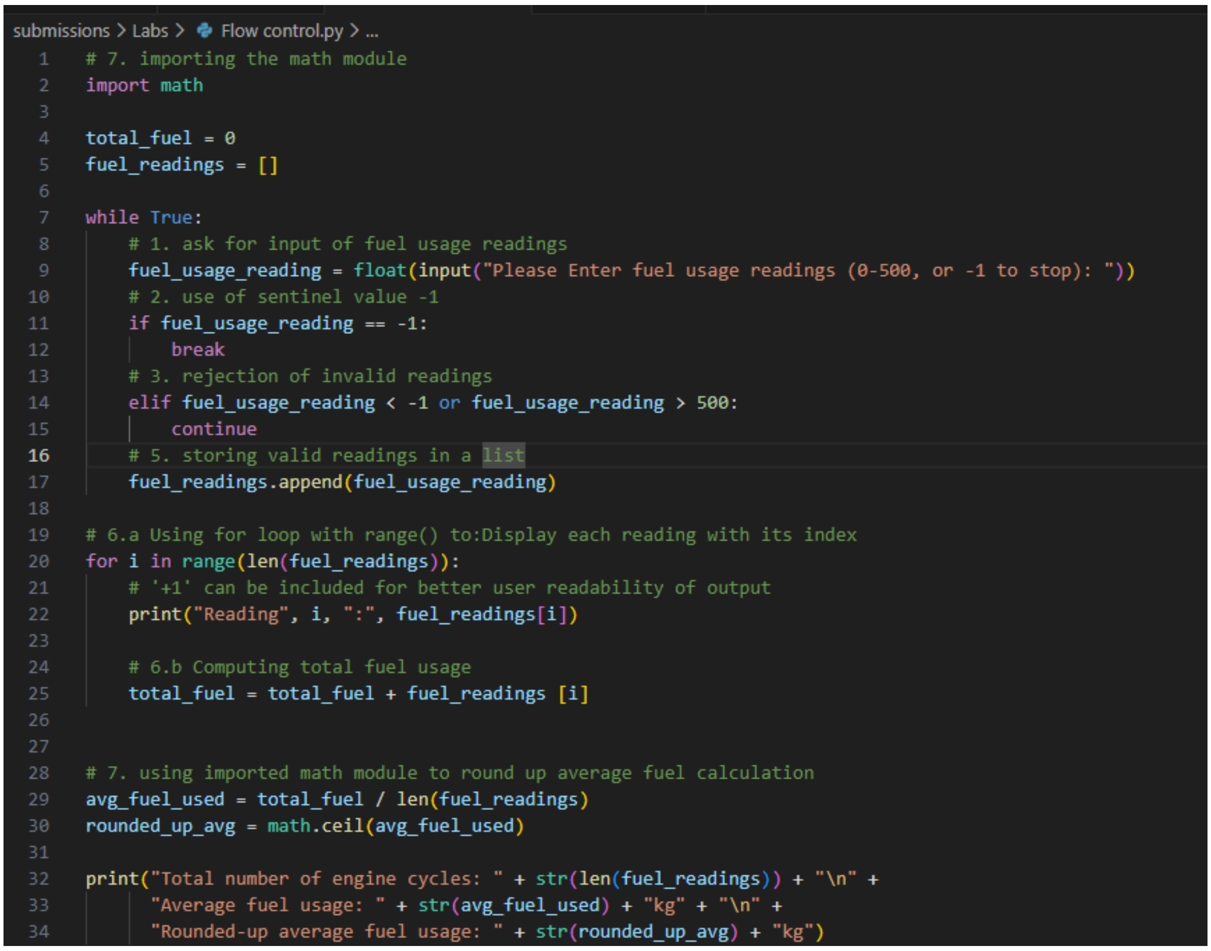
```python
# 7. importing the math module
import math

total_fuel = 0
fuel_readings = []

while True:
    # 1. ask for input of fuel usage readings
    fuel_usage_reading = int(input("Please Enter fuel usage readings (0-500, or -1 to stop): "))
    # 2. use of sentinel value -1
    if fuel_usage_reading == -1:
        break
    # 3. rejection of invalid readings
    elif fuel_usage_reading < -1 or fuel_usage_reading > 500:
        continue
    # 5. storing valid readings in a list
    fuel_readings.append(fuel_usage_reading)

# 6.a Using for loop with range() to:Display each reading with its index
for i in range(len(fuel_readings)):
    # '+1' included for better user readability of output
    print("Reading", i + 1, ":", fuel_readings[i])

    # 6.b Computing total fuel usage
    total_fuel = total_fuel + fuel_readings [i]


# 7. using imported math module to round up average fuel calculation
avg_fuel_used = total_fuel / len(fuel_readings)
rounded_up_avg = math.ceil(avg_fuel_used)

print("Total number of engine cycles: " + str(len(fuel_readings)) + "\n" +
        "Average fuel usage: " + str(avg_fuel_used) + "kg" + "\n" +
        "Rounded-up average fuel usage: " + str(rounded_up_avg) + "kg")
```

6. ChatGPT code results:



```
Jones@DESKTOP-QPUCUQ9 MINGW64 ~/Documents/Data Science/SAT4650
$ "C:/Users/Jones/Documents/Data Science/SAT4650/env/python.exe" "c:/Users/Jones/Documents/Data Science/SAT4650/submissions/Labs/chatGPT.py"
Enter fuel usage in kg (-1 to stop): 20
Enter fuel usage in kg (-1 to stop): 689
Invalid reading. Must be between 0 and 500 kg.
Enter fuel usage in kg (-1 to stop): 0
Enter fuel usage in kg (-1 to stop): -1
Cycle 0: 20.0 kg
Cycle 1: 0.0 kg

Summary
Number of engine cycles: 2
Average fuel usage: 10.0
Rounded-up average fuel usage: 10
(C:\Users\Jones\Documents\Data Science\SAT4650v)
```

7. Final source code (that displays index with fuel reading, and input variable set to float):



submissions > Labs > Flow control.py > ...

```python
# 7. importing the math module
import math

total_fuel = 0
fuel_readings = []

while True:
    # 1. ask for input of fuel usage readings
    fuel_usage_reading = float(input("Please Enter fuel usage readings (0-500, or -1 to stop): "))
    # 2. use of sentinel value -1
    if fuel_usage_reading == -1:
        break
    # 3. rejection of invalid readings
    elif fuel_usage_reading < -1 or fuel_usage_reading > 500:
        continue
    # 5. storing valid readings in a list
    fuel_readings.append(fuel_usage_reading)

# 6.a Using for loop with range() to:Display each reading with its index
for i in range(len(fuel_readings)):
    # '+1' can be included for better user readability of output
    print("Reading", i, ":", fuel_readings[i])

    # 6.b Computing total fuel usage
    total_fuel = total_fuel + fuel_readings [i]


# 7. using imported math module to round up average fuel calculation
avg_fuel_used = total_fuel / len(fuel_readings)
rounded_up_avg = math.ceil(avg_fuel_used)

print("Total number of engine cycles: " + str(len(fuel_readings)) + "\n" +
      "Average fuel usage: " + str(avg_fuel_used) + "kg" + "\n" +
      "Rounded-up average fuel usage: " + str(rounded_up_avg) + "kg")
```