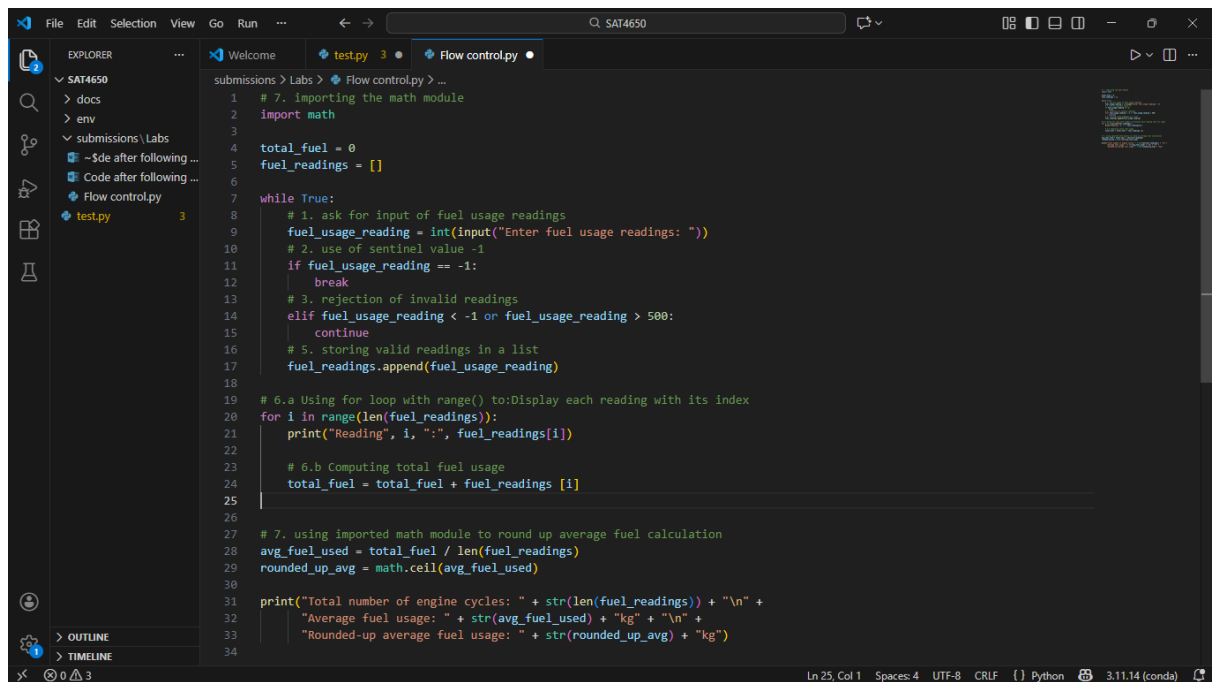


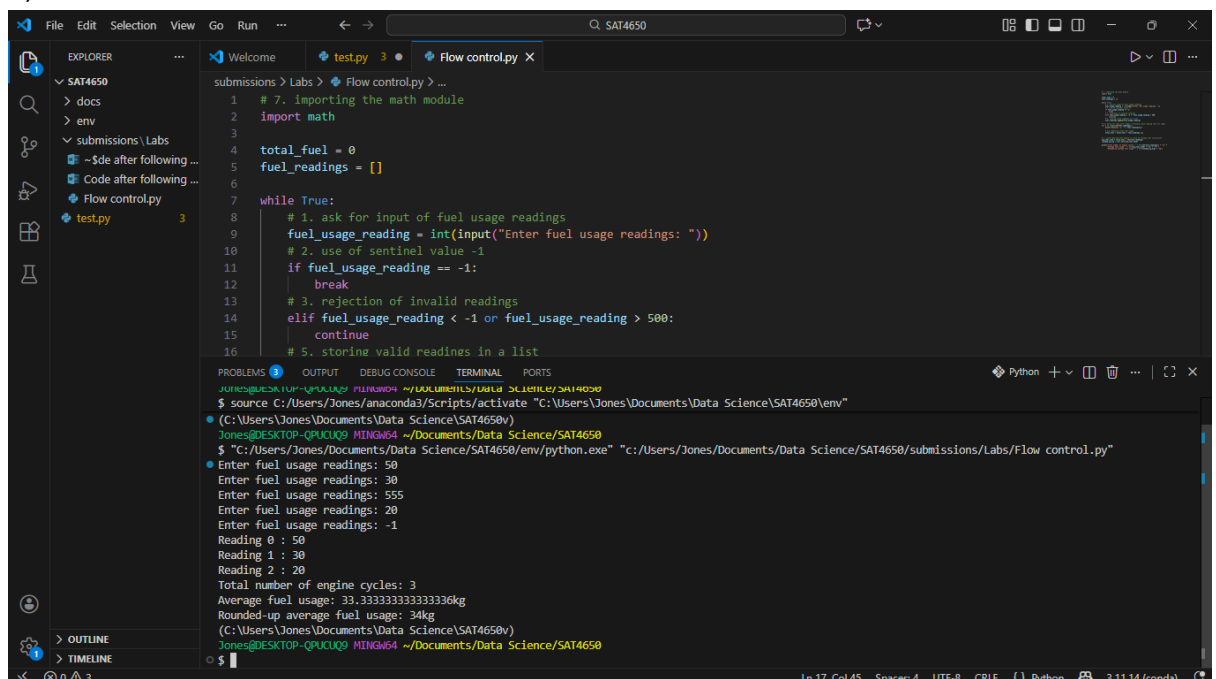
## Screenshots

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



```
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 = int(input("Enter fuel usage readings: "))
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     print("Reading", i, ":", fuel_readings[i])
22
23 # 6.b Computing total fuel usage
24 total_fuel = total_fuel + fuel_readings[i]
25
26
27 # 7. using imported math module to round up average fuel calculation
28 avg_fuel_used = total_fuel / len(fuel_readings)
29 rounded_up_avg = math.ceil(avg_fuel_used)
30
31 print("Total number of engine cycles: " + str(len(fuel_readings)) + "\n" +
32       "Average fuel usage: " + str(avg_fuel_used) + "kg" + "\n" +
33       "Rounded-up average fuel usage: " + str(rounded_up_avg) + "kg")
34
```

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

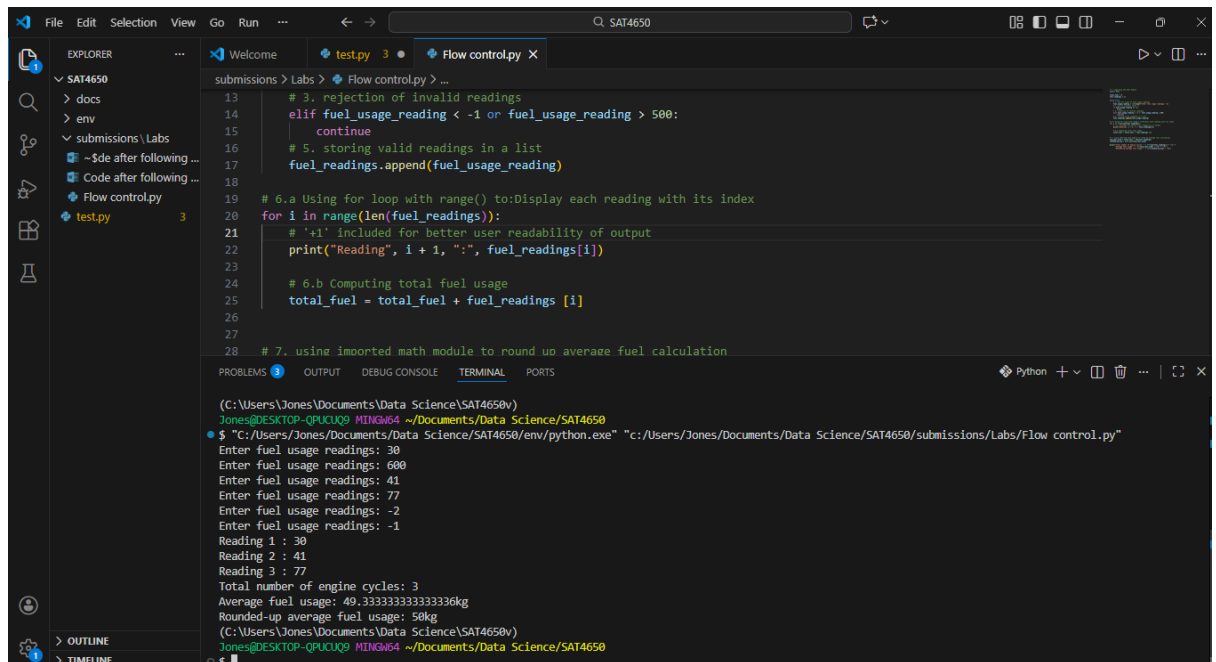


```
1 # 7. importing the math module
2 import math
3
4 total_fuel = 0
5 fuel_readings = []
6
7 while True:
8     # 1. ask for input of fuel usage readings
9     fuel_usage_reading = int(input("Enter fuel usage readings: "))
10    # 2. use of sentinel value -1
11    if fuel_usage_reading == -1:
12        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     print("Reading", i, ":", fuel_readings[i])
22
23 # 6.b Computing total fuel usage
24 total_fuel = total_fuel + fuel_readings[i]
25
26
27 # 7. using imported math module to round up average fuel calculation
28 avg_fuel_used = total_fuel / len(fuel_readings)
29 rounded_up_avg = math.ceil(avg_fuel_used)
30
31 print("Total number of engine cycles: " + str(len(fuel_readings)) + "\n" +
32       "Average fuel usage: " + str(avg_fuel_used) + "kg" + "\n" +
33       "Rounded-up average fuel usage: " + str(rounded_up_avg) + "kg")
34
```

```

$ source C:/Users/Jones/anaconda3/Scripts/activate "C:/Users/Jones/Documents/Data Science/SAT4650/env"
(C:/Users/Jones/Documents/Data Science/SAT4650)
$ python 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: 5
Average fuel usage: 33.33333333333333kg
Rounded-up average fuel usage: 34kg
(C:/Users/Jones/Documents/Data Science/SAT4650)
$
```

### 3. Test results after enhancing user readability:



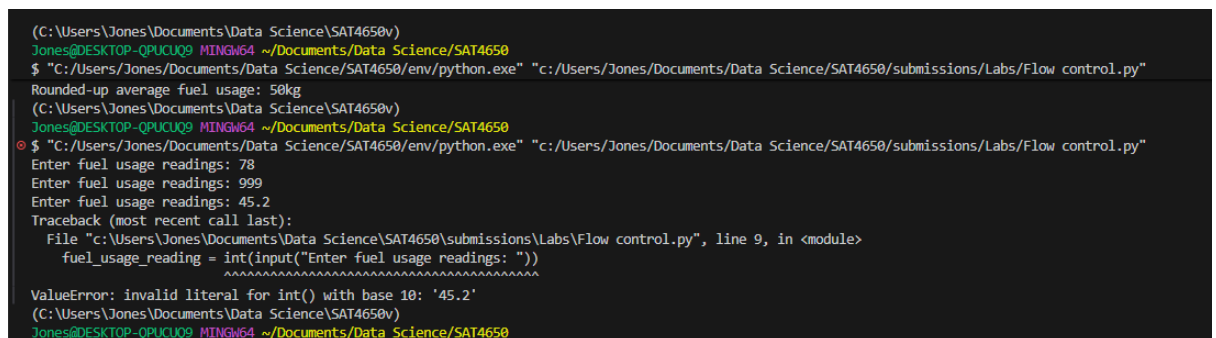
The screenshot shows a VS Code editor with a file explorer on the left, a code editor in the center, and a terminal at the bottom. The file explorer shows a project named 'SAT4650' with subfolders 'docs', 'env', and 'submissions'. The 'submissions' folder contains a 'Labs' folder with 'Flow control.py' and 'test.py'. The code editor shows the 'Flow control.py' file with the following code:

```
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' included for better user readability of output
22     print("Reading", i + 1, ":", 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
```

The terminal output shows the execution of the script:

```
(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: 40.333333333333336kg
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:



The screenshot shows a terminal window with the following output:

```
(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 :

```

Welcome | test.py 3 | Flow control.py X
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 = int(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' included for better user readability of output
22         print("Reading", i + 1, ":", 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")

```

6. ChatGPT code results:

```

Jones@DESKTOP-QP0CQ9: ~/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 > ...
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")
```