

Screenshots

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

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

# 5. storing valid readings in a list
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):

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

# 5. storing valid readings in a list
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")
```

JONES@DESKTOP-QPUCUQ: MINGW64 ~/Documents/Data Science/SAT4650\$ source C:/Users/Jones/anaconda3/Scripts/activate "C:/Users/Jones/Documents/Data Science/SAT4650/env"
JONES@DESKTOP-QPUCUQ: 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: 33.33333333333336kg
Rounded-up average fuel usage: 34kg
(C:/Users/Jones/Documents/Data Science/SAT4650)v
JONES@DESKTOP-QPUCUQ: MINGW64 ~/Documents/Data Science/SAT4650\$

3. Test results after enhancing user readability:

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows a project structure under "SAT4650" with files like "docs", "env", "submissions\Labs", "Flow control.py", and "test.py".
- Code Editor:** Displays the content of "Flow control.py". The code reads fuel usage readings from the user, rounds them up to the nearest integer, and calculates the total fuel usage.
- Terminal:** Shows the command line output of running the script. It prompts for fuel usage readings (30, 600, 41, 77, -2, -1) and prints the total fuel usage as 50kg.

```
# 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\SAT4650)
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: 49.33333333333336kg
Rounded-up average fuel usage: 50kg
(C:\Users\Jones\Documents\Data Science\SAT4650)
Jones@DESKTOP-QPUCUQ9 MINGW64 ~/Documents/Data Science/SAT4650
```

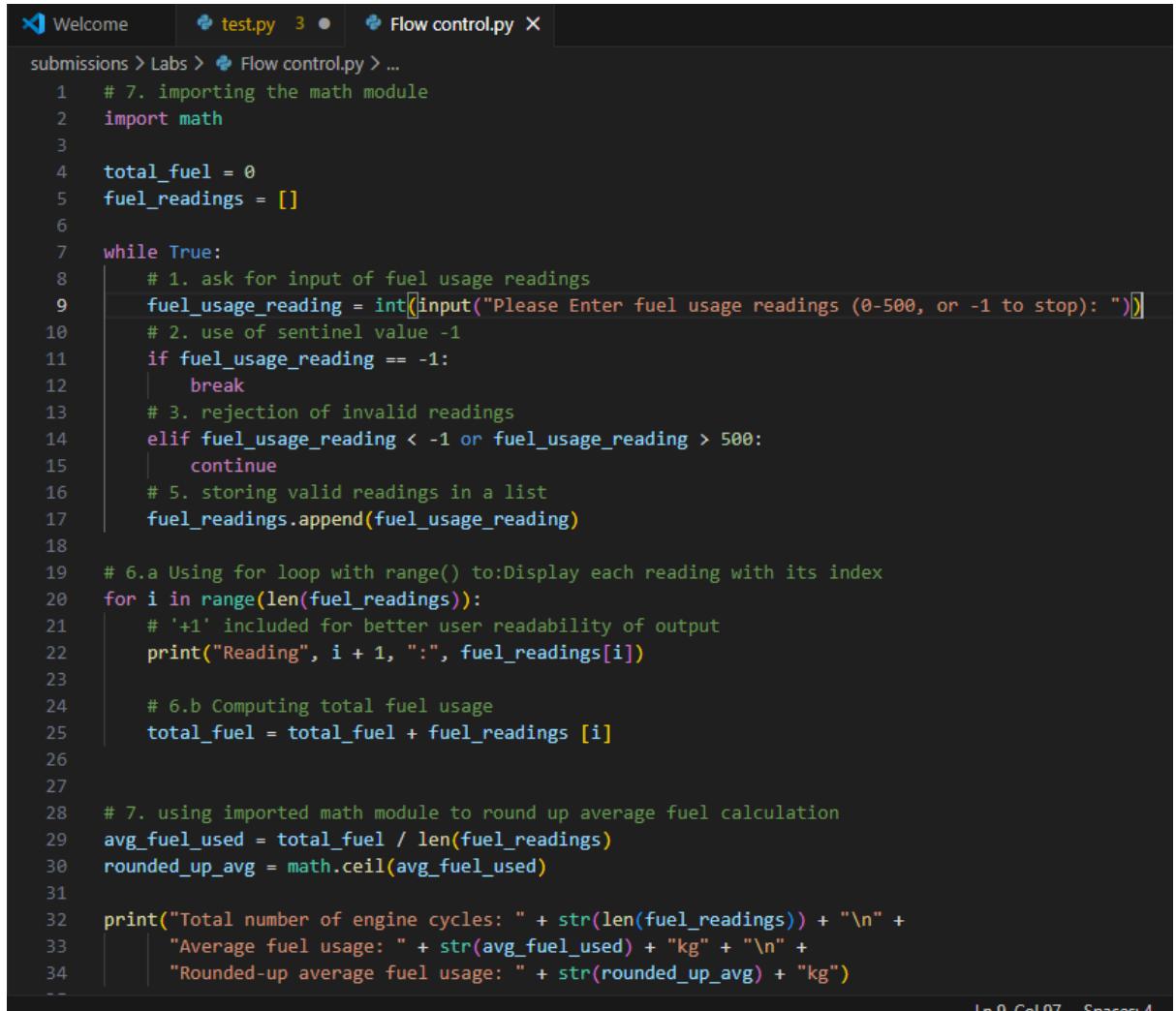
4. Error after entering decimal value:

The screenshot shows a terminal window with the following details:

- Terminal Output:** Shows a user attempting to enter fuel usage readings. When they enter "45.2", the script raises a `ValueError: invalid literal for int() with base 10: '45.2'`.

```
(C:\Users\Jones\Documents\Data Science\SAT4650)
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\SAT4650)
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\SAT4650)
Jones@DESKTOP-QPUCUQ9 MINGW64 ~/Documents/Data Science/SAT4650
```

5. source code enhanced for user readability :

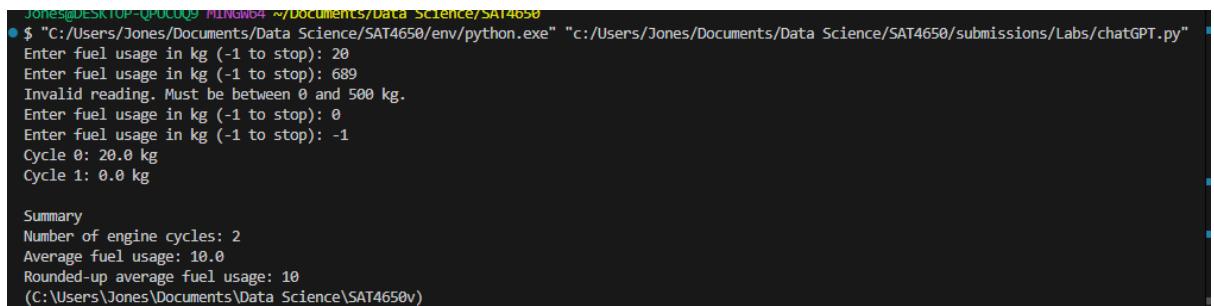


The screenshot shows a code editor window with two tabs: "test.py" and "Flow control.py". The "test.py" tab is active and displays the following Python code:

```
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")
```

In 9 Col 97 Spacing 4

6. ChatGPT code results:



The screenshot shows a terminal window with the following output:

```
Jones@DESKTOP-QPUCQ9 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 > ...
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")
```