



Carbon Footprint Calculator

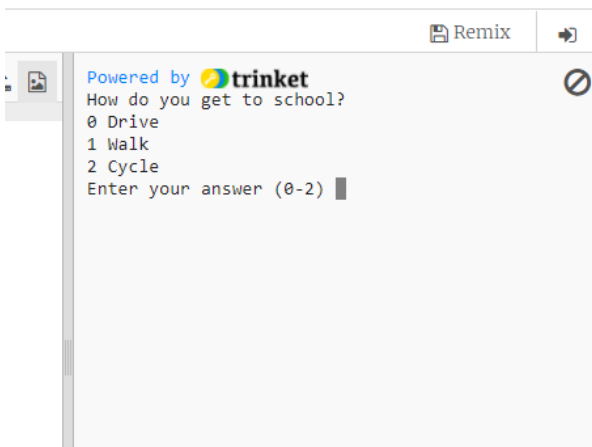
Create your own tool to measure your personal carbon output.



INTRODUCTION

What you will make

You will make a simplified carbon footprint calculator that can help people to understand their personal impact.



What you will learn

- Loop through data in a list.
- Sum values based on user input.
- Dynamically format strings.
- Validate input.

Code Club Australia recognises the Traditional Custodians of the land across Australia and their continuing connection to land, cultures, and communities. Australia's traditional owners are the world's first innovators.

What you will need

HARDWARE

A computer capable of accessing Trinket online. You do not need a Raspberry Pi to complete this project.

DOWNLOADS

Offline starter project
bit.ly/mhcarboncalc

Additional notes for educators

Here is a link to the completed project
<https://trinket.io/python/bd3506101e>

Check out our [blog post](#) for this project with tips, curriculum and supporting material at [medium.com/@codeclubau](#)

1. LIST QUESTIONS AND ANSWERS

First, we want to ask the user some questions, and provide them with the possible answers that will allow us to measure their carbon footprint.



- Open the starter project at bit.ly/mhcarboncalc
- On the left is your code window, on the right is the result of your code. You can click the “Run” button to run your code. You’ll notice that the first 2 lines have already been written for you.
- The first line tells us to use version 3 of Python, the second gets the data for our carbon calculator.



```
1  #!/bin/python3
2  from data import questions
3
4  |
```



- Our data consists of a series of questions, with each of the answers given a value for how much carbon we’re using.
- Let’s start by listing out all the questions. To do this, we’ll write a ‘for’ loop that will give us each of the questions in turn.



```
1  #!/bin/python3
2  from data import questions
3
4  ▾ for question in questions:
5
```



- We want to write out each question. To do this, we’ll use the ‘print’ command.
- The questions are already in the data.py tab so we don’t need to write them out.




```
1  #!/bin/python3
2  from data import questions
3
4  ▾ for question in questions:
5      print(question['question'])
```



- Run your program using the ‘Run’ button. In your Result window, you should see a list of questions.



Powered by  trinket
How do you get to school?
How often do you eat beef?
What do you set your aircon to in summer?



- For each question, we want to list out the answers. Add a second loop that outputs all the possible answers to the question.



```
4  ▾ for question in questions:
5      print(question['question'])
6
7  ▾ for answer in question['answers']:
8      print(answer['answer'])
```

- We want to number our answers to give the user something easy to type in. First, we need to set an initial value for our number.
- When programming we often use the letter 'i' to indicate a counting number.



```

4 for question in questions:
5     print(question['question'])
6     i = 0
7     for answer in question['answers']:
8         print(answer['answer'])

```

- Next, we need to increase the value of 'i' every time we go around our answer for loop.



```

4 for question in questions:
5     print(question['question'])
6     i = 0
7     for answer in question['answers']:
8         print(answer['answer'])
9         i += 1

```

- Now we can add this to the text of our answer.
- Run your program. You should now get all of the questions with numbered answers.



```

4 for question in questions:
5     print(question['question'])
6     i = 0
7     for answer in question['answers']:
8         print(i, answer['answer'])
9         i += 1

```

2. TOTALLING IT UP

Next, we will get all the user's responses and total up their emissions.

- Now that we're asking the user a question, we need to get their answer.
- Add the following code to ask the user for their answer and store it in the variable named "response".
- Make sure the new code is in the right indentation.
- For the question, we want to get the carbon amount from the relevant answer. Add the code to print out the amount of carbon.



```

6     i = 0
7     for answer in question['answers']:
8         print(i, answer['answer'])
9         i += 1
10
11     response = input("Enter your answer")

```

```

10
11     response = input("Enter your answer")
12     print(question['answers'][response]['carbon'])

```

- Run your program. What happens when you try and put in a response? You probably get an error saying this:

```
TypeError: list indices must be integers, not str on line 12 in main.py
```

- The reason we get this error is because the input command gives us a string (text) instead of an integer (number). We need to convert our response into a number before we can use it.
- Add this code and run your program again. That's better!

```
10  
11 response = input("Enter your answer")  
12 response = int(response)  
13 print(question['answers'][response]['carbon'])
```

- Instead of showing the amount of carbon after each step, we want to keep a running tally and display it after we've asked all the questions.
- First, we need to initialise our total to 0 before we start our question loop.



```
1 #!/bin/python3  
2 from data import questions  
3  
4 total = 0  
5 for question in questions:  
6     print(question['question'])
```

- Instead of printing out the amount of carbon used after each question, we want to add it to the total.

```
12 response = input("Enter your answer")  
13 response = int(response)  
14 total += question['answers'][response]['carbon']
```

- Finally, we want to print this out after we've finished looping through the questions.

```
13 response = int(response)  
14 total += question['answers'][response]['carbon']  
15  
16 print(total)
```

- Note that we don't have any spaces in front of line 16. The spaces are called 'indenting' and they tell Python that a command is in the loop. We don't want our print statement to be in the loop, so we don't want it to be indented. Make sure this code is right against the edge.
- Run your program. You should be able to answer all of the questions and get an answer at the end.

- Currently, the program is just spitting out a number. Let's make that number a bit more user friendly.
- Update your print line to the following to give your user some context to the number:

```
14 total += question['answers'][response]['carbon']
15
16 print("You emit approximately", total, "kg of CO2 each year.")
```

- Run your program again. You should now get the total CO2 emitted displayed in a readable format.

3. INPUT VALIDATION

You have a working CO2 calculator, but what happens if the user enters a number that the program isn't expecting, or something that isn't a number at all? Did you see an error called "IndexError"? In the next step we'll validate the user input so that if they enter something incorrect, it doesn't crash the whole program.

- The first step in validation is to tell the user what is expected.
- Let's tell our user what is expected of them when you ask them for their answer.

```
10 i += 1
11
12 response = input("Enter your answer (0-{})".format(i))
13 response = int(response)
```

- The 'format' command might be a little confusing, but for now all you need to know is that it replaces '{}' with the number stored in 'i'.

- Run your program. You might notice that the range given is one larger than the number of options. Oh no! We can fix this substituting "i-1" instead of "i" in format.

```
10 i += 1
11
12 response = input("Enter your answer (0-{})".format(i-1))
13 response = int(response)
```

- Run your program again. That's better!

- Now we're telling the user what to enter, they should get it right, but just because they should get it right, it doesn't mean they will! Everyone makes mistakes, and some people will deliberately try and break programs (that's one of the ways that hackers can break into computers). We still need to check that the input is valid.
- Let's start by checking that a number is actually entered before we tell python to convert it into a number.
- (Note that we've indented line 14 to the next level to tell Python that it's inside the if statement)

```

12 response = input("Enter your answer (0-{})".format(i-1))
13 if response.isdigit():
14     response = int(response)
15 total += question['answers'][response]['carbon']

```

- If the user enters something that isn't a digit, we should tell them that they need to enter a number.

```

13 if response.isdigit():
14     response = int(response)
15 else:
16     print("That's not a number!")
17
18 total += question['answers'][response]['carbon']

```

- Next, we need to validate the number, to ensure that it's in the right range, only then will we add the number to the total. Add the 'if' below, and indent the existing 'total +=' line.

```

16     print("That's not a number!")
17
18 if response >= 0 and response < i:
19     total += question['answers'][response]['carbon']
20
21 print("You emit approximately", total, "kg of CO2 each year.")

```

- Run your program.
- What happens if the user enters something wrong? It doesn't break the program anymore, but it just goes on to ask the next question, and so won't give an accurate result. We want to ask the user the question again and again until they give a valid response.

- To achieve this, we'll use a while loop. This will keep looping through until the condition is satisfied (in this case, we get a valid response). Start by initialising a Boolean variable.
- We will assume a response is not valid until we've checked everything required to tell us that it is.

```

10     i += 1
11
12     valid = False
13     response = input("Enter your answer (0-{}).format(i-1))
14     if response.isdigit():
15         response = int(response)
16     else:
17         print("That's not a number!")

```

- Our while loop will run all of our validations.
- (Note that we've indented everything up to line 21 to another level. You can do this by highlighting all of the code between lines 14-21 that you want to indent and pressing the TAB key.)

```

12     valid = False
13     while not valid:
14         response = input("Enter your answer (0-{}).format(i-1))
15         if response.isdigit():
16             response = int(response)
17         else:
18             print("That's not a number!")
19
20         if response >= 0 and response < i:
21             total += question['answers'][response]['carbon']
22
23     print("You emit approximately", total, "kg of CO2 each year.")

```

- If you run your program now, you will be stuck in an endless loop, because we never set valid to True. Let's do that if our number is between the correct range.

```

17     else:
18         print("That's not a number!")
19
20     if response >= 0 and response < i:
21         valid = True
22
23     total += question['answers'][response]['carbon']
24
25     print("You emit approximately", total, "kg of CO2 each year.")

```

- Note that we've moved line 23 out of the if statement and the while loop by decreasing the indent by two levels. We did this by using the backspace at the beginning of the line.



- If the user enters a number outside of the correct range, we should tell them what they've done wrong

```
20     if response >= 0 and response < i:  
21         valid = True  
22     else:  
23         print("Please enter a number between 0 and", i-1)  
24  
25     total += question['answers'][response]['carbon']
```

- Run your program. It should now be working and correctly validating user responses.

Challenges:

On repeat

- Can you loop through the entire program so that multiple people can use the carbon calculator without having to reset it every time?

Advanced Challenge: Graph the result

- Can you graph the carbon footprints of your classmates? If you want to graph it in Python, check out the [Popular Pets Code Club](#) project at.

Congratulations you're a Moonhack changemaker!

Don't forget to talk to an adult about registering your participation at moonhack.com

