You've all now had the experience of trying to read and understand some pretty unclear code. I'd like to talk briefly about how to make your code *not* be unclear.

Variable and function names

Given names If the task you are given involves any algebraic names (N, x, etc.), use precisely those names in your code to represent precisely those quantities.

Related names If you have a variable that is related to the above, use a related name. If you have a maximum value for N, call it something like `Nmax`.

Long names If you can't see a good name for something, try a sentence. e.g. `number_of_chains_to_average`.

No name Sometimes when using a `for` loop, for instance for averaging, you don't actually care about which one you're working on. You can make your code a bit more clear by giving the customary (in Python) name `_` to the variable.

```
mean_x_squared = 0
for _ in range(number_chains_to_average):
    x = compute_chain_x(N)
    mean_x_squared = mean_x_squared + x**2
mean_x_squared = mean_x_squared/number_chains_to_average
```

By *not* giving this variable a name, we can make it clear that it doesn't matter. Fewer names to parse makes code simpler.

No variable If a variable is not used at all, just remove it from the code. Every assignment adds to the difficulty in reading the code.

Comments Comments are one way to add an explanation of what your code is doing. They are an approach of last resort, to be used when your code is confusing even with optimal variable names. The major problem with comments is that frequently the code will end up changing and leaving the comments out of date. Once you put comments into your code, it becomes important to keep updating them when you touch the code, which is a nuisance, and if you fail to make those changes, then the code ends up being *harder* to read because of the comments leading the reader astray.

Nice functions A final suggestion is that *sometimes* your code can be made more clear by defining a function. The key is to ensure that your function has a role that is easy to explain, with well-defined inputs and output, and a great name.