1. DRY and "Do One Thing"

00:00 - 00:12

DRY (also known as "don't repeat yourself") and the "Do One Thing" principle are good ways to ensure that your functions are well designed and easy to test. Let's see how.

2. Don't repeat yourself (DRY)

00:12 - 00:42

When you are writing code to look for answers to a research question, it is totally normal to copy and paste a bit of code, tweak it slightly, and re-run it. However, this kind of repeated code can lead to real problems. In this code snippet, I load my train, validation, and test data, and plot the first two principal components of each dataset. I wrote the code for the train dataset, then copied it and pasted it into the next two blocks, updating the paths and the variable names.

3. The problem with repeating yourself

00:42 - 00:57

But one of the problems with copying and pasting is that it is easy to accidentally introduce errors that are hard to spot. If you'll notice in the last block, I accidentally took the principal components of the train data instead of the test data. Yikes!

4. Another problem with repeating yourself

00:57 - 01:20

Another problem with repeated code is that if you want to change something, you often have to do it in multiple places. For instance, if we realized that our CSVs used the column name "label" instead of "labels", we would have to change our code in six places. Repeated code like this is a good sign that you should write a function. So let's do that.

5. Use functions to avoid repetition

01:20 - 01:44

Wrapping the repeated logic in a function and then calling that function several times makes it much easier to avoid the kind of errors introduced by copying and pasting. And if you ever need to change the column "label" back to "labels", or you want to swap out PCA for some other dimensionality reduction technique, you only have to do it in one or two places.

6. Problem: it does multiple things

01:44 - 01:48

However, there is still a big problem with this function.

7. Problem: it does multiple things

01:48 - 01:50

First, it loads the data.

8. Problem: it does multiple things

01:50 - 01:52

Then it plots the data.

9. Problem: it does multiple things

01:52 - 02:07

And then it returns the loaded data. This function violates another software engineering principle: Do One Thing. Every function should have a single responsibility. Let's look at how we could split this one up.

10. Do One Thing

02:07 - 02:46

Instead of one big function, we could have a more nimble function that just loads the data and a second one for plotting. We get several advantages from splitting the load_and_plot() function into two smaller functions. First of all, our code has become more flexible. Imagine that later on in your script, you just want to load the data and not plot it. That's easy now with the load_data() function. Likewise, if you wanted to do some transformation to the data before plotting, you can do the transformation and then call the plot_data() function. We have decoupled the loading functionality from the plotting functionality.

11. Advantages of doing one thing

02:46 - 03:07

The code will also be easier for other developers to understand, and it will be more pleasant to test and debug. Finally, if you ever need to update your code, functions that each have a single responsibility make it easier to predict how changes in one place will affect the rest of the code.

12. Code smells and refactoring

03:07 - 03:26

Repeated code and functions that do more than one thing are examples of "code smells", which are indications that you may need to refactor. Refactoring is the process of improving code by changing it a little bit at a time. This process is well described in Martin Fowler's book, "Refactoring", which is a good read for any aspiring software engineer.

13. Let's practice!

03:26 - 03:33

Now you can do some refactoring of your own in the exercises!