Rationale

Let's study why functions are important.

We'll cover the following

Breaking your code down into discrete functions allows you to re-use code in intelligent ways, and to make your code more efficient overall. You can use functions written by other people (by linking them to your programs) and avoid having to write everything from scratch yourself.

Writing functions to perform common tasks means that you can essentially write your meta-language. For example **if** you have functions already written to perform these tasks, your C program might look like this:

```
#include <stdio.h>

int main() {
     loadSubjects();
     excludeSubject(12);
     lowpassFilterData();
     collectMeansWithBinSize(8);
     pval = performANOVA();
     printf("my hypothesis is correct, p < %.3f\n", pval);
     return 0;
}
```

The idea is that once we know what sorts of operations on our data we will wish to do, we can write a set of functions (sometimes called subroutines), to **abstract** away the details, and provide us with a sort of high-level meta-language that we can use to carry out the steps we wish to.

Abstraction

Using functions to abstract away the details also means that as long as the function inputs and outputs are known, then the user doesn't need to know the details of how the function performs its task.

Another way to think about this: let's imagine you wrote a function `lowpassFilterData()`, and it worked well. Let's say your friend came along one day and noticed that it was taking a long time to process your data, and he suggested you use a different algorithm for low-pass filtering, and he gave you a file containing the code. As long as the inputs and outputs are the same, then you can simply switch out your function for the new one, and all functionality should be the same.

You can think of functions as "black boxes" with input and output wires. As long as the wires are labeled the same, and there is the same number, you can replace one box with a newer box, and perhaps get faster (or more accurate, for example) functionality **without changing your main program**.

Just as with variables, there is a certain syntax we need to follow for defining functions. Let's look at it in the next lesson.