**How to write good code**

When you are just starting out you tend to be interested in just getting the code to work. However after a while you probably start to think, *Am I writing good code?* So what does ‘good’ code look like. Here are some guidelined which will help you be a good coder

1. **Make sure it works**, if it runs without errors and gets you the result it is good code. Remember that the computer does not care about the style of the code, it simply executes what is there. However this is only the first step in writing good code, but it is an essential first step. A lot of coders write very bad code simply to get things done quickly and then go back and refactor their code to make it better.
2. **Make sure it is readable**. Make your variable and function names meaningful. This is the key to making your code maintainable. When you go back to your code in six month’s time or when some-else reads your code they must understand what is going on. Good meaningful names are a huge help, a far greater help than comments. Most coding projects are done by teams and good readable code is essential so that others in the team can understand what you are doing.
3. **Follow conventions of the language**, which means other programmers can easily read your code. All programmers develop their own style of writing code, however often languages also have a preferred style to the language (or sometime teams have a preferred style). It is better to adjust your style as this will help others to read and understand your code. Python has a style guide in PEP 8, you should read it and follow it when you can.
4. **DRY principle (Don't Repeat Yourself).** If you are writing the same code in two places, you probably need a function for that. Good code is often made up of a huge number of small functions which are self-contained and then stitched together in the main program flow. This makes your code much more robust since all the parts can be fully tested and also allows you to reuse those functions in other parts of the program or other programs.
5. **Don't nest too deeply**. The classic case is many if statements inside each other. If you are doing this then probably you could think of a better way. Also it might look clever to have a complex line of code but it is often difficult for you and others to read. A quick test is that you should be able to read your code by quickly scanning it. If you have to stop and think through what is happening, then you code is probably too complex.
6. **Structure matters**. You should think about how your program is designed. Any non trivial program will have many lines of code and how this is put together matters. When you are trying to maintain it or develop new features you often need to think about where the processing for this should occur. This decision is all about what structure you have for you programs. A good structure means that it is easy to find the code and know where to place new features, a bad structure means that code is all over the place.

**How to get better**:

Practise and read other people's code, learn to use the libraries and most of all think about how your code looks and is organised.