Coding for A-level Mathematicians

Aim:

As a mathematician teaching myself to code, I like to make sense of the tools I'm using. I appreciate that to fully grasp a concept or process it is often necessary to work with it *before* it makes sense, but I like to have at least an idea of what a thing is for, and perhaps how it fits in to my current understanding. I have thoroughly enjoyed learning to code using Python 3, and the more I learn the more I enjoy it, and the more I find I can do with it. Thus far, it is a lot like mathematics, and I think there could be some benefit to approaching the process of learning to code like we learn maths (and vice versa).

I'm going to assume a fair amount of maths knowledge, and the ability to think logically, and then give you the explanations and examples that I feel would have been most helpful for me when I was first learning. I'm also going to try and do it all through the medium of mini projects. Each new skill should be learnt because it will help solve a particular problem or simplify a particular process, and bringing in new elements only as they become necessary should make it easier to maintain motivation when learning new things.

Disclaimer:

I am an experienced teacher of A-level Maths and Further Maths, but only an enthusiastic self-taught amateur when it comes to coding. While I will try to follow best practice in terms of layout, naming conventions, readability etc, my primary aim is not to teach coding but to provide an introduction to programming for people who are, as I was once, a keen mathematician but a total beginner to coding.

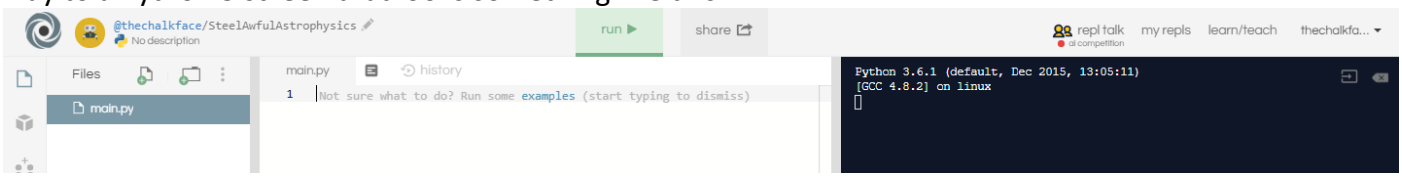
Setup:

Any beginner's guide to Python will talk you through the steps of installing this on your own operating system. You may find recommendations for different editing programs, and these can be invaluable for larger projects, but you won't be disadvantaged using a basic text editor for the types of projects we will be working on here. A great alternative to installing Python on your machine is to use an online emulator like repl.it. You can start coding in Python 3 without even logging in, but if you want to be able to save your code it's a good idea to sign up for a free account. This is likely to be the preferred option if you don't have admin rights on your computer (eg a school machine) or you want to share code with others easily.

Getting started:

The best way to learn coding is to code. I'll try to explain ideas as clearly as I can, but all of the concepts will no doubt have been explained better elsewhere. My aim here is not to write a 'How to Code' book (check out Al Sweigart's series for my own personal recommendation of a good starting point there), but to provide a series of mathematical projects that would suit and interest a mathematician learning to code.

For the sake of diving straight in, we'll assume you are coding online using repl.it, and you have found your way to a Python 3 screen that looks something like this:



How to learn:

If you're a mathematician, this should all sound very familiar. I have found the process of learning to code is a lot like learning some new mathematics. The most important similarity is that mistakes are good when learning. Certainly, bugs in an app you've created would be annoying and a dropped minus sign in an exam may lose you marks, but while *learning* to code, just like when learning maths, you will get more from identifying, dissecting and understanding your mistakes than you will from doing it right first time.

Coding for A-level Mathematicians

Aim:

As a mathematician teaching myself to code, I like to make sense of the tools I'm using. I appreciate that to fully grasp a concept or process it is often necessary to work with it *before* it makes sense, but I like to have at least an idea of what a thing is for, and perhaps how it fits in to my current understanding. I have thoroughly enjoyed learning to code using Python 3, and the more I learn the more I enjoy it, and the more I find I can do with it. Thus far, it is a lot like mathematics, and I think there could be some benefit to approaching the process of learning to code like we learn maths (and vice versa).

I'm going to assume a fair amount of maths knowledge, and the ability to think logically, and then give you the explanations and examples that I feel would have been most helpful for me when I was first learning. I'm also going to try and do it all through the medium of mini projects. Each new skill should be learnt because it will help solve a particular problem or simplify a particular process, and bringing in new elements only as they become necessary should make it easier to maintain motivation when learning new things.

Disclaimer:

I am an experienced teacher of A-level Maths and Further Maths, but only an enthusiastic self-taught amateur when it comes to coding. While I will try to follow best practice in terms of layout, naming conventions, readability etc, my primary aim is not to teach coding but to provide an introduction to programming for people who are, as I was once, a keen mathematician but a total beginner to coding.

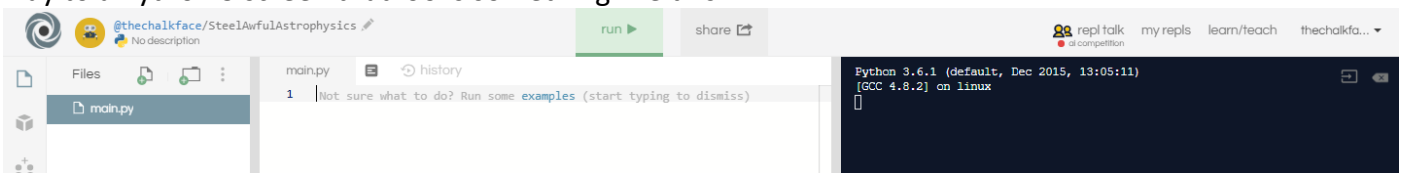
Setup:

Any beginner's guide to Python will talk you through the steps of installing this on your own operating system. You may find recommendations for different editing programs, and these can be invaluable for larger projects, but you won't be disadvantaged using a basic text editor for the types of projects we will be working on here. A great alternative to installing Python on your machine is to use an online emulator like repl.it. You can start coding in Python 3 without even logging in, but if you want to be able to save your code it's a good idea to sign up for a free account. This is likely to be the preferred option if you don't have admin rights on your computer (eg a school machine) or you want to share code with others easily.

Getting started:

The best way to learn coding is to code. I'll try to explain ideas as clearly as I can, but all of the concepts will no doubt have been explained better elsewhere. My aim here is not to write a 'How to Code' book (check out Al Sweigart's series for my own personal recommendation of a good starting point there), but to provide a series of mathematical projects that would suit and interest a mathematician learning to code.

For the sake of diving straight in, we'll assume you are coding online using repl.it, and you have found your way to a Python 3 screen that looks something like this:



How to learn:

If you're a mathematician, this should all sound very familiar. I have found the process of learning to code is a lot like learning some new mathematics. The most important similarity is that mistakes are good when learning. Certainly, bugs in an app you've created would be annoying and a dropped minus sign in an exam may lose you marks, but while *learning* to code, just like when learning maths, you will get more from identifying, dissecting and understanding your mistakes than you will from doing it right first time.