## 

## Introduction

Hi there, I'm Andy Colley, a computer science teacher from Manchester, UK. Thanks for looking at these resources. If you have any feedback, suggestions or questions you can find me on Twitter at [@MrAColley](https://twitter.com/mracolley?lang=en) or email me at [andy@learningdust.com](mailto:andy@learningdust.com).

Happy coding!

## How To Use This Course

This course is designed for students with experience of the following programming techniques:

* Output
* Input
* Variables
* Simple selection
* Simple iteration
* Lists
* Basic subroutines

These techniques will be referred to, built upon and used in the lessons and tasks. It is not essential that students have direct experience with the Python language as the syntax is fairly straightforward and simple to pick up.

If you’re not sure that your students are ready to start this course, you might want to check out my ‘[Introduction To Python’](https://docs.repl.it/curriculum/introPython) course instead

This course is designed to build on the ideas learned in the introduction unit and prepare the students with coding skills roughly equivalent to those needed for GCSE level in the UK. Of course, the more students practice and are able to combine and modify their coding skills to solve more complex problems, the better they will do. Topics covered are:

* Strings - string functions and manipulation
* Numbers - data types random numbers, modulo,
* Selection - multiple conditions using AND, OR, nesting selection.
* Iteration - multiple boolean conditions, validation, combining selection and iteration
* Lists - output a range of items from a list, search a list
* Files - read from a file, write to a file
* Error Handling - Using try/catch.

I’ve split this course into seven sessions, each designed to take roughly 2-3 hours.

For each session there is a more in depth read me document to help you get started. This contains teacher notes to break down the key concepts linked to the coding skills, links to the tasks and example solutions on repl.it and the text of the tasks set. There’s also a slide deck for each week with presenter notes containing context, subject knowledge cues, possible questions to ask your students and other useful links.

For every task I’ve provided a hyperlink to a repl that students can work on. Share this with them and get them to ‘fork’ their own version. I’ve also linked to an example solution. There are often multiple ways of completing a task, so students may well have different code that works (mine often do).

## Pedagogy

I have made use of the PRIMM pedagogical approach in this course. This model follows the following structure:

| Stage | Activities | Why |
| --- | --- | --- |
| Predict | In pairs look at a piece of code printed out or on the whiteboard and ask students what they think it will do. They can write down their prediction or discuss in a group class discussion. Live coding is quite useful here or the code can be already prepared. | This activity encourages students to look for clues in the program that suggest what its function is. |
| Run | Fork the example code on repl.it and check against your prediction. | Having code that is provided has many benefits – it moves the weight of ownership of any errors from the student to the teacher, increasing confidence, and it also means that time is not spent copying in code, which can be a challenging exercise to students who struggle with literacy at any level. |
| Investigate | There are lots of different activities you can do at this stage: trace through the code, comment the code, answer questions about it, label particular concepts, highlight it, draw the flow of control, etc. Again pair work helps to encourage discussion about the nitty gritty of the program. | It takes many activities of this type, repeated in different forms in different lessons, for students to start to understand the underlying concepts in a secure way. We may tend to think that writing one selection statement correctly means that students have a good understanding of selection but really “getting” this takes some time. |
| Modify | Given a working piece of code, students are challenged to add a variety of modifications, starting very simply and having a series of exercises increasing in difficulty with larger modifications. | The transfer of ownership moves from the code being “not mine” to “partly mine” as students gain confidence by extending the function of the code. This activity obviously provides the scaffolding that students need to add small snippets of code and see their effect within a bigger program. |
| Make | Once students are confident in modifying the program that you have created, they can create their own program from scratch, which has similarities with the previous program but that they can design themselves. | Design of a new program is an important skill, and should start with planning and trying to construct a suitable algorithm. This is difficult, but does give students an opportunity to be creative and have the satisfaction of making their own program. |

Source: <https://blogs.kcl.ac.uk/cser/2017/09/01/primm-a-structured-approach-to-teaching-programming/>

The lesson tasks give you a start point and some ideas for applying the PRIMM model. Depending on your students you may need to add in more examples of your own. I wouldn’t recommend skipping tasks, especially with beginner coders, as practice is vital to embed the skills & understanding needed for independence.