Demystifying programming

Introduction

When looking at the day to day advances of technology, it may not always feel like we are making leaps and strides. However, if you imagine where we were 100 or even 20 years ago, the rise of computers truly is remarkable. Let's look back slightly over 50 years ago to 1969, when Apollo 11 first landed humankind on the moon. 🚀🌙

Onboard Apollo 11 was a guidance computer with 4,096 bytes of RAM. RAM stands for “Random Access Memory” and stores temporary information. To put that into context, a single alphabetical character, like the letter “a”, takes up 1 byte of storage, meaning that 4,096 characters could be stored in RAM.

Let's compare that to a modern smartphone, which normally has around 4GB of RAM. 4GB is roughly equivalent to four *billion* (4,000,000,000) bytes. A modern smartphone can store over one million times more data than the computer in Apollo 11!

Memory isn’t the only thing that matters to a computer. The Apollo 11 computer could perform operations at a speed of 0.043 MHz. The processor in a high-end mobile phone might run at a speed of around 2500 MHz. If we compare these numbers, we can see that the phone in your pocket could have over 100,000 times the processing power of the computer that landed humans on the moon 50 years ago. 😳

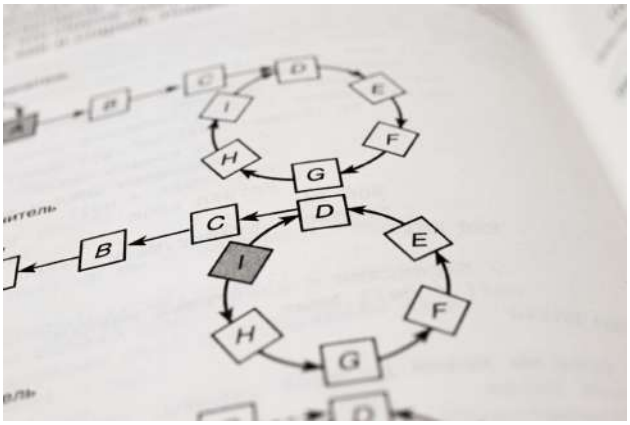
So, does that mean that my mobile phone can fly a spaceship to the moon? Weeell, not quite. At least not by itself. Computers can not act entirely on their own (at least not yet); they rely on a programmer to give them instructions. So while your mobile phone has the storage and the processing power necessary to do so, it won't know how to fly the Apollo 11 to the moon unless you write a ***program*** to tell it what to do.

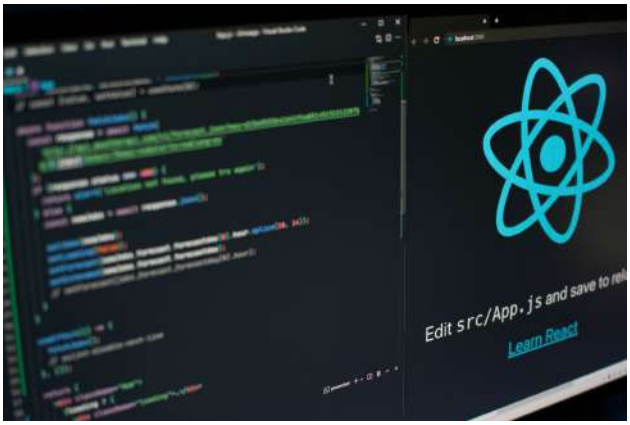
What is a program?

Put simply, a program is a set of instructions that tell a computer what to do.

- Programs follow simple linear paths:
  do A then do B
- Or they can have more complex structures, like outlining to the computer how to make its own decision:
  do either A or B based on some information O

We usually write a program because we want our computer to complete a specific task. Whenever we want to complete a task in real life, we typically break it down into a series of steps that we can follow. In programming terms, what these steps are and how we move between them is referred to as an algorithm.





You already use algorithms in your everyday life, maybe without realising it. The most obvious example would be following a recipe while cooking 👩‍🍳, but it also extends to almost everything we do.

For example, let's say that you get a notification on your mobile phone that someone has sent you a message 📲. After receiving this information, you need to unlock your phone, navigate to the messaging app, open it, read the message, and then decide on an appropriate response based on its content. This series of steps is an example of an algorithm to accomplish the task of responding to a message notification.

Programming is about thinking of algorithms to accomplish a task and then translating that algorithm into something that the computer can read. This translated algorithm is what we call a program.

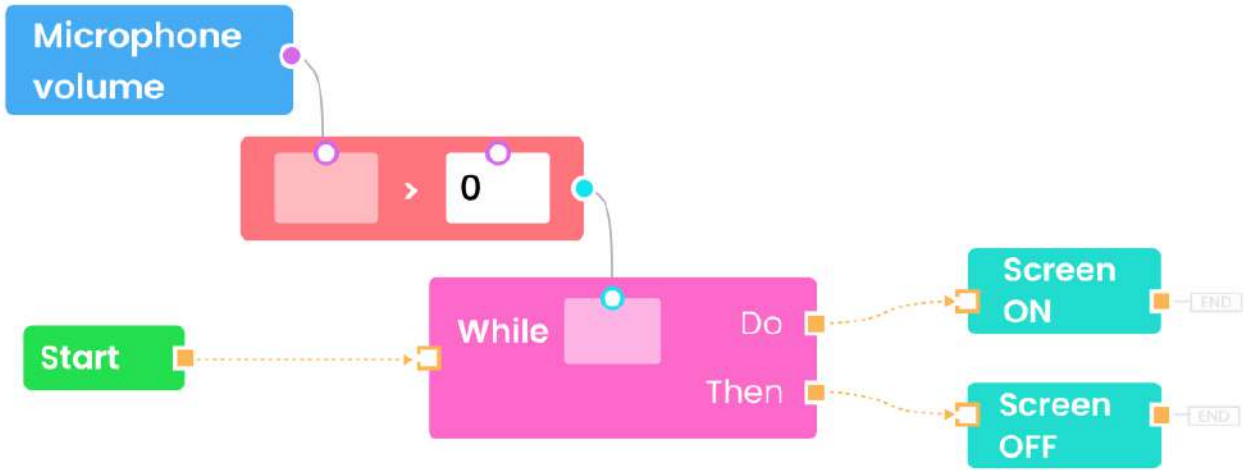
How do we write one?

Programs need to be written in unique languages that computers can read. Just like there are multiple languages that we can write instructions for humans in, like Te Reo or English, there are also numerous languages in which we can write computer programs.



Computers are very picky in what they can read. Your algorithm needs to be translated correctly to the programming language you are using. Like any other language, programming languages have their own unique syntaxes that need to be followed.

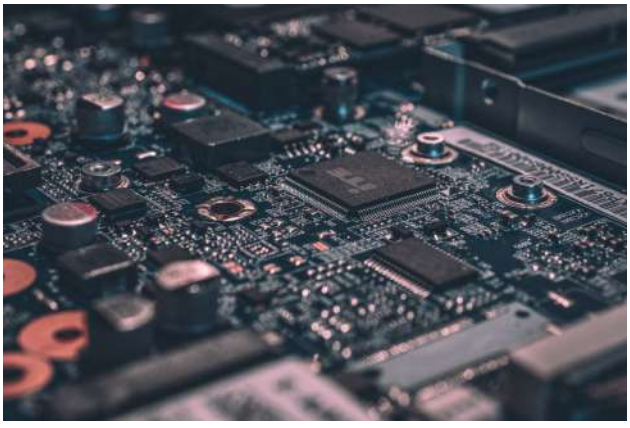
In this Project, you will be using a visual language called Flow. Flow is very similar to creating flow charts and requires very little syntax memorisation, so it is an ideal language for learning the fundamental principles of programming.

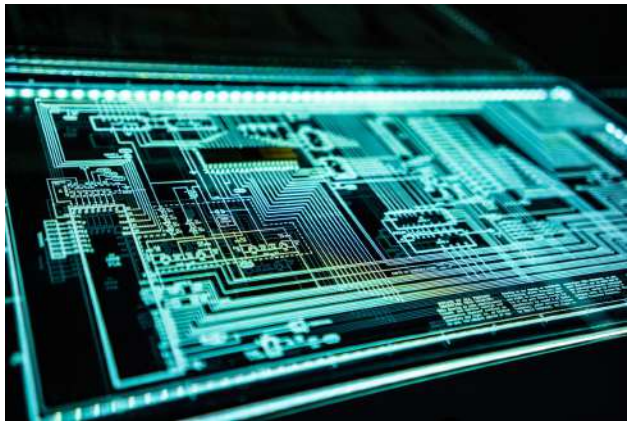


Although there are many different programming languages, the core concepts of programming – logical thinking, problem-solving and translating algorithms – are identical between them. Once you have mastered these with Flow, picking up any other language becomes much easier.

How do computers read code? (optional extra)

The code that you write is not actually the same code that the computer reads. It first needs to go through a series of conversions.





Although it is not necessary to understand how this works to complete this Project, if you do want to know more then check out the following video in your own time:

How do computers read code?