What is an API?

Application Programming Interfaces (APIs) help different pieces of software talk to each other. When you write a program, you typically use a bunch of existing libraries for the programming language of your choice. These libraries provide APIs in the form of ***external*** or ***public*** functions, classes, and methods that other code can use to get their job done without having to create a lot of repeated code.

And not only that, APIs can also be used by other pieces of software, even if they were written in a completely different programming language. For example, Cloud services use APIs that your programs can communicate with by making web calls. What's special about an API? What makes it different to any other function that you would write in your own code?

If you look at the library's code, you'll find it has many functions that we're not meant to use directly from our code. These ***internal*** or ***private*** functions, classes, and methods do important work, but they're there to support the functions that are published by the library. You probably don't have time to dig in to understand every little bit of how the code works, but you need to know how to interact with the library to do useful work. An API is sort of like a promise. Even if the library's internal code changes, you expect the function to keep accepting the same parameters and returning the same results. That provides a stable ***interface*** to write your code with. That's an API!

Library authors are free to make improvements and changes to the code *behind* the interface, but they shouldn't make changes to the way the functions are called or the results they provide. Because this would break the code that depends on that library. When a library author needs to make a ***breaking change*** to an API, then they need to have a plan in place for communicating that change to their users. That's why breaking changes *should* be saved for major version increments of a library.

When you choose a certain library to use with your code, the first step is to get familiar with its API. You'll need to look at how the functions are called, what inputs they expect, and what outputs they'll return.