Universal Quantification

In Haskell, you can generalise types. If you label a parameter of type `a` it can represent any data type. For example if you wanted to make a function that calculated the length of a list you wouldn't make one for each different type of list:

e.g. `[Int] -> Int`, `[Char] -> Int` etc.

Instead you just write `[a] -> Int` meaning "give me a list of anything and I will return you an Int". It is almost like a wildcard, if you define a function with these general types you're basically saying that you don't mind what type that parameter is. This allows you to create more general functions. These functions are *polymorphic* since they are not specific about what data types they take in.