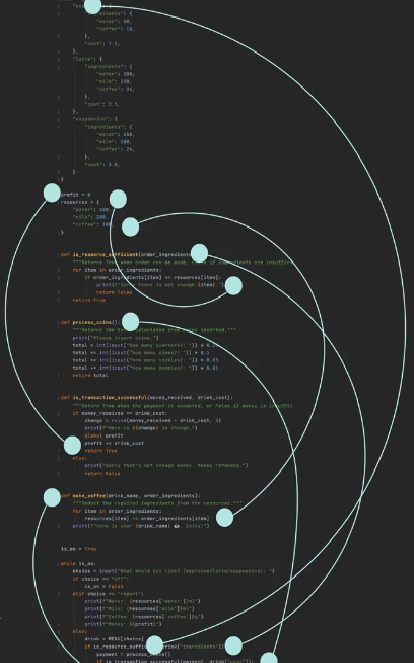
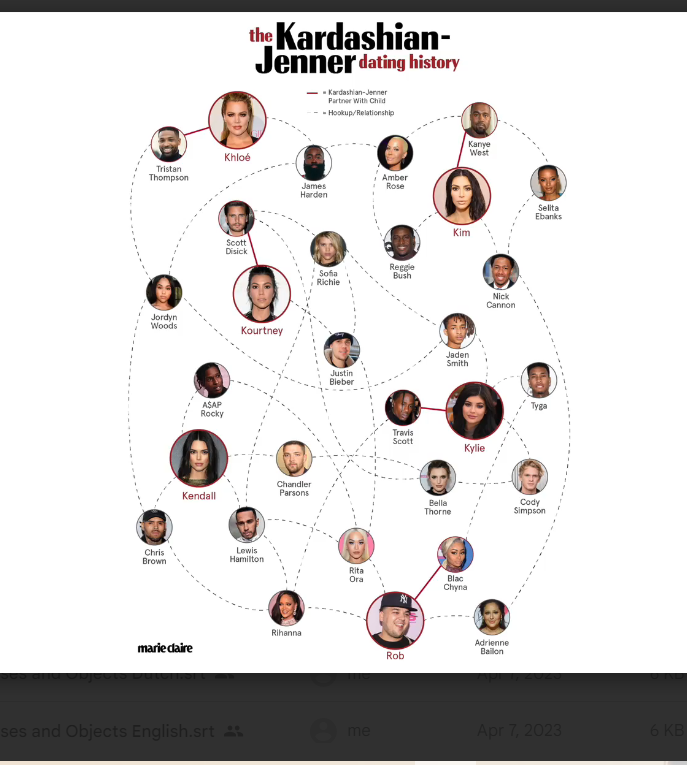
Now today, we're going to be pulling out the big guns, we're going to be learning about Object Oriented Programming otherwise known as OOP. Why are we learning about it? Well, if you think back to yesterday's project, some of you I'm sure at some point during the project, would have had this feeling of what on earth is going on with my code? And the reason is because our code is getting more complex, it's trying to do a lot of things, and it's trying to manage a whole bunch of relationships. So one function is changing a variable, and then that same function is doing something else to another variable. And at some point, the logic of our code starts looking very, very spaghetti-like.



And it's at this point where it's really hard to track and remember what's actually going on in our code. So this style of programming is called Procedural Programming where we set up procedures or functions that do particular things. And then one procedure leads to another procedure, and all in all, the computer's mostly working from top to bottom and then jumping out into a function as needed. Procedural programming is one of the earliest paradigms of Programming. In fact, back in the days when we had older languages like Fortran and COBOL, they rely pretty much exclusively on procedural programming like what we've been doing. But the increase in complexity, the increase in the number of relationships that we need to remember and manage starts making it look a bit like the dating history of the Kardashians.

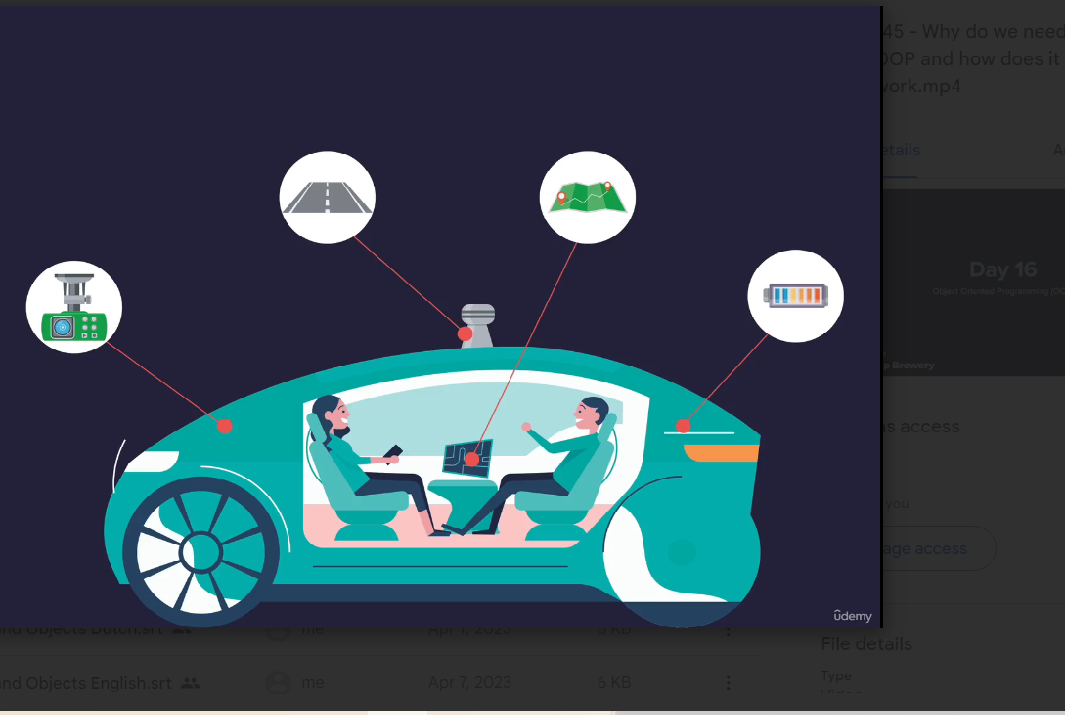


And it gets very, very confusing because everybody has been in touch with somebody else. Now, the kind of relationships I prefer are one to one, me and my couch.

So how can we maintain a simple relationship in our code while being able to write more and more complex projects? Well, this is where the Object-Oriented paradigm comes in really handy.



Now let's imagine that you're tasked with creating the program for a self- driving car. Now, as you can imagine, this is a pretty complex project and it's many, many notches above the coffee machine that we've been struggling with so far. But what if you broke it down? What is a self-driving car? What are the different components that make up a self-driving car? Well, it probably needs some sort of camera module to keep track of what's on the road and to recognize what's on the road, it will probably need a form of lane detection to know if we're actually within the lane or if we need to turn off or if we need to park, and it will need some way of navigating so that when the user says, I want to go to the bank, they identify the branch they want to go to, and then the navigation gets set up and the car knows how to get there. And finally, you'll probably need some form of fuel management, right? What should happen when the fuel gets low? Should it go and automatically recharge at a specified point? Now I'm obviously simplifying this task. There's a lot, a lot more that goes into making an automated car. But we've already managed to break it down into several modules that we can think about tackling, right?

But imagine if you have a whole team who are all working on this project and within that team, there's sub-teams who are working on each of these different modules. Well then by splitting up this big complex task into separate modules, then we can all work on this car simultaneously, massively improving our productivity, making it much quicker to eventually build all the software for this car. Plus on top of that, a lot of these modules are reusable. So if it just so happens that the next year we're tasked with building a drone, well, a lot of those programming modules that we built, like the camera module, the fuel management, the navigation, that's all going to be incredibly useful in our drone delivery software as well.