**Important: Note on learning by building**

**Tutorials vs Real-World Smart-Contract Building**

When it comes to building solidity projects, things may seem a bit too linear or straightforward when you watch a demo or read a tutorial. Looking at a video where Patrick streamlines a project from start to finish and his code compiling from the first try 99.9% of the time might give you the wrong idea of how this process normally goes.

Keep in mind that Patrick built this project or a close version of it using Solidity + Brownie, Solidity + Hardhat and now Solidity + Foundry and probably updated them multiple times to adjust for different changes in Solidity versions, VRF versions and so on. When building something completely new, Patrick, like any other smart contract developer, doesn't do it seamlessly or in one go.

Normally, when you start building a new project, you will write 1-2 functions and then try to compile it ... and BAM, it doesn't compile, you go and fix that and then you write a couple of tests to ensure the functionality you intend is there ... and some of these tests fail. Why do they fail? You go back to the code, make some changes, compile it again, test it again and hopefully everything passes. Amazing, you just wrote your first 1-2 functions, your project will most likely need 10 more. This might look cumbersome, but it's the best way to develop a smart contract, far better than trying to punch in 10 functions and then trying to find out where's the bug that prevents the contract from compiling. The reason why Patrick is not testing every single thing on every single step is, as you've guessed, the fact that the contract will be refactored over and over again, and testing a function that will be heavily modified two lessons from now is not that efficient.

***You won't develop smart contracts without setbacks. And that is ok!***

***Setbacks are not indicators of failure, they are signs of growth and learning.***