In this lecture I want to help you understand Solidity and Smart Contracts on a theoretical level, reason being, they are perhaps the two most key terms to really understand at depth.

Before we dive deeper I want to give you a brief understanding of how they’re intertwined.

At their core, Smart Contracts are programs that exist upon the Ethereum Blockchain. A Smart Contract can be programmed to run a variety of functions. Whether that’s; Transacting ERC-20 Tokens, Record Keeping, Blockchain Based Lotteries, and so on. There are many tasks!

You may have noticed I mentioned the term “programmed” not so long ago, well this is how Smart Contracts are put together. Because, a Smart Contract is just a computer program. The most common languages in which Smart Contracts are programmed, is Solidity.

Let me dive into Solidity a little more with you now.

This high-level programming language was created by several Ethereum Core contributors, such as Gavin Wood, for the purpose of Smart Contract Development.

If you’ve ever come across, or used Solidity, you’ll notice the similarity between it and JavaScript. Solidity does the process of verifying and enforcing the constraints at compile-time as opposed to the run-time.

Getting right to the point, Solidity is their to develop and execute code in the EVM, i.e. the Ethereum Virtual Machine.

Now let me switch over and go a little deeper into Smart Contracts with you.

So this very new technology, Smart Contracts, is basically a means to exchange money, shares, property, vehicles - in fact anything of value, without any need for a middle-person. Best of all, a Smart Contract helps us do this in a secure manner.

Naturally as with a traditional contract their are terms applied, whether it’s consequences for late payments and so on. But rather than this being written in text form as is with a traditional contract, these terms are automatically applied to Smart Contracts if and when programmed in.

Hence making these Smart Contracts self-executable!

So as you can tell, the actual relationship between Solidity and Smart Contracts is one of huge importance. A Smart Contract can’t be created without the use of the Solidity Programming Language - Solidity just happens to be the most popular language.

Smart Contracts and Solidity combined allow a great number of ways of allowing contracts to interrelate with other contracts.

Remember, that it’s not possible to change the code of a published Smart Contracts, reason being, it would defeat the purpose of the immutable nature of the blockchain. However, upgradeable contracts to, implementing a newly positioned version of the marked contract while fixing some bugs has been discussed. But it’s not time for that in this lecture!

Just to wrap up on this lecture, let me give you an example workflow of what a process of publishing a Smart Contract upon the Ethereum platform would look like!

1. The developer or developers would write the code of the Smart Contract using Solidity.

2. Once the code is finalised it would be uploaded to the Ethereum Virtual Machine.

3. Once the code is on the EVM, it will then be the same across all nodes.

4. Each node will then evaluate whether the conditions in the contract where met.

5. If met, the Ether will be re-distributed as was coded in. If not, the clause coded in will kick in.

And remember, this transaction could be anything. Whether it was a contract to deliver a web design project, signing the deeds of a house over, signing a vehicle over and much more!

I’ll see you in the next lecture, where I’ll be showcasing just how popular decentralised applications currently are. You may be surprised at this!