## Why is Unit-Testing so important?

**a) It helps you find bugs, regression bugs and sometimes also helps you to understand your code from different angles.**

b) It is a great way to spend time on something that you get paid for. But ultimately it will just slow down the development process.

## If you are starting a new ERC20 token

a) It would be best to start from scratch, just looking at the required interface

b) It is beneficial to copy and paste the already existing code from the Ethereum wiki and modify this until you like it

**c) Best is to start with an audited implementation, for example from OpenZeppelin, in order to re-use already existing code.**

## To generate a random number

a) It's good to use the block timestamp, as this is always different

b) It's good to use the block hash as this is clearly always very different

**c) It's good to use the RANDAO smart contract**

d) It's not possible to have a random number in a deterministic environment such as the Ethereum blockchain.

## When you do external calls to other smart contracts

**a) You should follow the checks-effects-interactions pattern and avoid state changes after the call**

b) You should follow the effects-checks-interactions pattern and avoid state changes before the call

c) You should follow the checks-effects-interactions pattern, which is only necessary when you do calls to contracts where a direct contract call is not possible.

## When you are programming a game like poker of battleships where you need to hide opponents values is

a) with private state variables. This way nobody else than the smart contract itself can see the information.

b) with external contracts holding those values. This way we can make sure that the information flow is following a clear logic and nobody else can access this information.

c) You can't hide anything on the blockchain, because the information is public, just the call is private which means only other smart contracts would be limited in accessing that information.

## When considering smart contracts and the blockchain it's good

a) To move all existing logic to the blockchain, so everything runs on the same system. This way it might be more complex, but easier to maintain.

**b) To move only those parts to the blockchain that really need the blockchain. This way smart contracts can be easier to read, easier to test and are not so complex.**

c) To move those parts to the blockchain that deal with Ether transfers. All other parts can remain in traditional database systems. This way only the value-transfer is on the blockchain.

## When a smart contract pays out money

a) It’s good to use a push over a pull method

**b) It’s good to use a push and a pull method to ensure that participants can get their money no matter the contract state. In addition to and pushing it should contain a withdraw method.**

c) It’s good to use only pull and no push method.

## To develop smart contracts:

a) It’s good to start with a local in-memory blockchain with unit tests but then deploy to the main-net as rapidly as possible.

**b) It’s good to start with a local in-memory blockchain with unit-tests. Then, in the next step, debug and test the smart contract on a test-net like Ropsten or Rinkeby with beta customers to iron out last issues before deploying it to the main-net.**

c) It’s good to start with a test-net with beta-customers like on the Rinkeby or Ropsten testnet, before testing it locally on an in-memory blockchain simulation such as Ganache. Then deploy it to the main-net.

## To avoid issues during Ethereum platform upgrades

a) It’s good to inform users about the updates via a newsletter

**b) It’s good to have the ability to pause a contract in order to manage the money at risk**

c) Ethereum doesn’t upgrade the platform. It’s fixed and final.