Ethereum has 2 types of accounts:

1. EOA externally owned – has a public -private key linked to it
2. Smart contract accounts

State variables are stored in blockchain are declared inside a contract and not inside a function.

Variables inside a function are not state variable and they are not stored on blockchain.

Two types of function:

1. Create txn (updates state variable etc)
2. Do not create txn (view and pure type(are identifiers in a function)). view: Only reads the state variable from SC. Pure: not even read the state variable in SC

Key word “storage” or “memory”:

Storage means input is something already stored in the contract. While memory means local argument.

2 ways to get state variables:

1. Write your own functions.
2. Let solidity create it for you. Solidity creates getter for public state variables.

Smallest unit of ether is wei 10^18 = 1 ether

Gwei= 10^9 wei

Before mentioning wei or gwei or eth in solidity, one should pass literals i.e values. Variables are not permitted.

e.g

int x=10;

x eth; is invalid but 10 eth is valid

**Txn fees = Gas used \* Gas price**

Txn with higher gas will be processed faster and unspent gas will be refunded.

If a txn runs out of gas, changes to the state variable are reverted but you still have to pay for the gas

Q/A

Bytecode is the opcode / instruction to EVM

ABI- When an external application or another smart contract wants to interact with the blockchain, it needs to have some knowledge of a smart contract’s interface such as a way to identify a method and its parameters. This is facilitated by the Ethereum Application Binary Interface (ABI).

It is similar to the Application Program Interface (API), which is essentially a representation of the code’s interface in Human readable form or a high-level language. In the EVM the compiled code being stored as binary data and the human-readable interfaces disappear and smart contract interactions have to be translated into a binary format that can be interpreted by the EVM. ABI defines the methods and structures that you can simply use to interact with that binary contract (just like the API does), only on a lower level. The ABI indicates to the caller the needed information (functions signatures and variables declarations) to encode such that it is understood by the Virtual Machine call to the byte code(contract). This process is called **ABI encoding**.

An Ethereum smart contract is byte code deployed on the Ethereum blockchain. There could be several functions in a contract. An ABI is necessary so that you can point to the specific function call that you want to evoke and also get a guarantee that the function will return data in the format you are expecting.   
**The first four bytes of a transaction payload sent to a contract is usually used to distinguish which function in the contract to call.**

(ABI has the methods and variables of source code )which acts as a bridge between sc and app/other sc.

When we perform a txn or in other words call some function of ABI which modifies state variable then does it affect byte code, because bytecodes are immutable .

So are variables are not present in byte code they are just memory addresses and when a update a state variable the mem address is updated and not the byte code, hence this byte code may remain unchanged / immutable.

2. About the license commented line of source code.