Inheritance in Solidity

- A contract acts like a class. A contract can inherit from another contract known as **the base contract** to share a common interface;
- The general inheritance system is very similar to Python's, especially concerning multiple inheritance;
- Solidity supports multiple inheritance including polymorphism. Multiple inheritance introduces problems like the "diamond problem" and should be avoided;
- When a contract inherits from multiple contracts, **only a single contract is created** on the blockchain, and the code from all the base contracts is copied into the created contract;
- All function calls are virtual, which means that the most derived function is called, except when the contract name is explicitly given;
- When deploying a derived contract the base contract's constructor is automatically called;
- is keyword is used when declaring a new derived contract;

Abstract Contracts

- An abstract contract is the one with at least one function that is not implemented and is declared using the abstract keywork;
- You can mark a contract as being abstract even though all functions are implemented;
- **An abstract contract cannot be deployed;**

Interfaces

- Interfaces are similar to abstract contracts, but they cannot have any functions implemented;
- **Interfaces can be inherited;**
- Interfaces have further restrictions:
  - They cannot inherit from other contracts, but they can inherit from other interfaces;
  - All declared functions must be external;
  - They cannot declare a constructor;
  - They cannot declare state variables;
- An interface is created using the interface keyword instead of contract;