2️⃣

Basics II

@June 8, 2022

1- Decision Making In Solidity

```
contract DecisionMaking {
  function validateCode(uint code) public view returns (bool) {
    if (code == 1234) {
      return true;
    } else {
      return false;
    }
  }
}
```

2- Scope

- **local**
  - declared inside a function
  - not stored on the blockchain
- **state**
  - declared outside a function
  - stored on the blockchain
- **global** (provides information about the blockchain)

```
contract C {

    uint public data = 10; // state variable

    function x() public returns (uint) {
        data = 30; // local variable
        return data;
    }
```

```
    function y() public view returns (uint) {
        return data;
    }
}
```

3- Visibility

State Variable Visibility

- `public` allows other contracts to read their values, but not modify them.
- `internal` state variables can only be accessed from within the contract they are defined in and in derived contracts. They cannot be accessed externally. This is the default visibility level for state variables.
- `private` state variables are like internal ones but they are not visible in derived contracts.

```
contract StateVisibilities {
  uint data = 5; //internal
  uint private data2 = 10;
  uint public data3 = 20;
}
```

> Making something `private` or `internal` only prevents other contracts from reading or modifying the information, but it will still be visible to the whole world outside of the blockchain.

Function Visibility

There are 2 types of function calls:

1. **External** requires an actual EVM message call
2. **Internal** do not

Visibility Identifiers for Functions:

- `public` means that the defining contract, inheriting contracts, EOAs, and other smart contracts can all access it.

- `private` is the most restrictive visibility. Private functions are like internal ones but they are not visible in derived contracts.
- `external` functions are part of the contract interface, which means they can be called from other contracts and via transactions. An external function cannot be called internally.
- `internal` functions can only be accessed from within the current contract or contracts deriving from it. They cannot be accessed externally.

4- Comparison & Check for Conditions

The `require` Solidity function **guarantees** validity of conditions that cannot be detected before execution. It checks **inputs**, **contract state variables** and **return values** from calls to external contracts.

5- Strings & Bytes

We need to define where we are going to keep the string data while declaring it. Even for functions, we need a declaration for the storage.

Finding string lengths in Solidity is not usual as in other languages since it's expensive. Instead, we convert a string into bytes format, and find the length afterwards.

```
contract Strings {
    string favouriteColour;

    function setColour(string memory _colour) public {
        favouriteColour = _colour;
    }

    function returnColour() public view returns (string memory) {
        return favouriteColour;
    }

    function getLength() public view returns (uint) {
        bytes memory temp = bytes(favouriteColour);
        return temp.length;
    }
}
```