



klaytn

# KLAYTN DEV BOOTCAMP

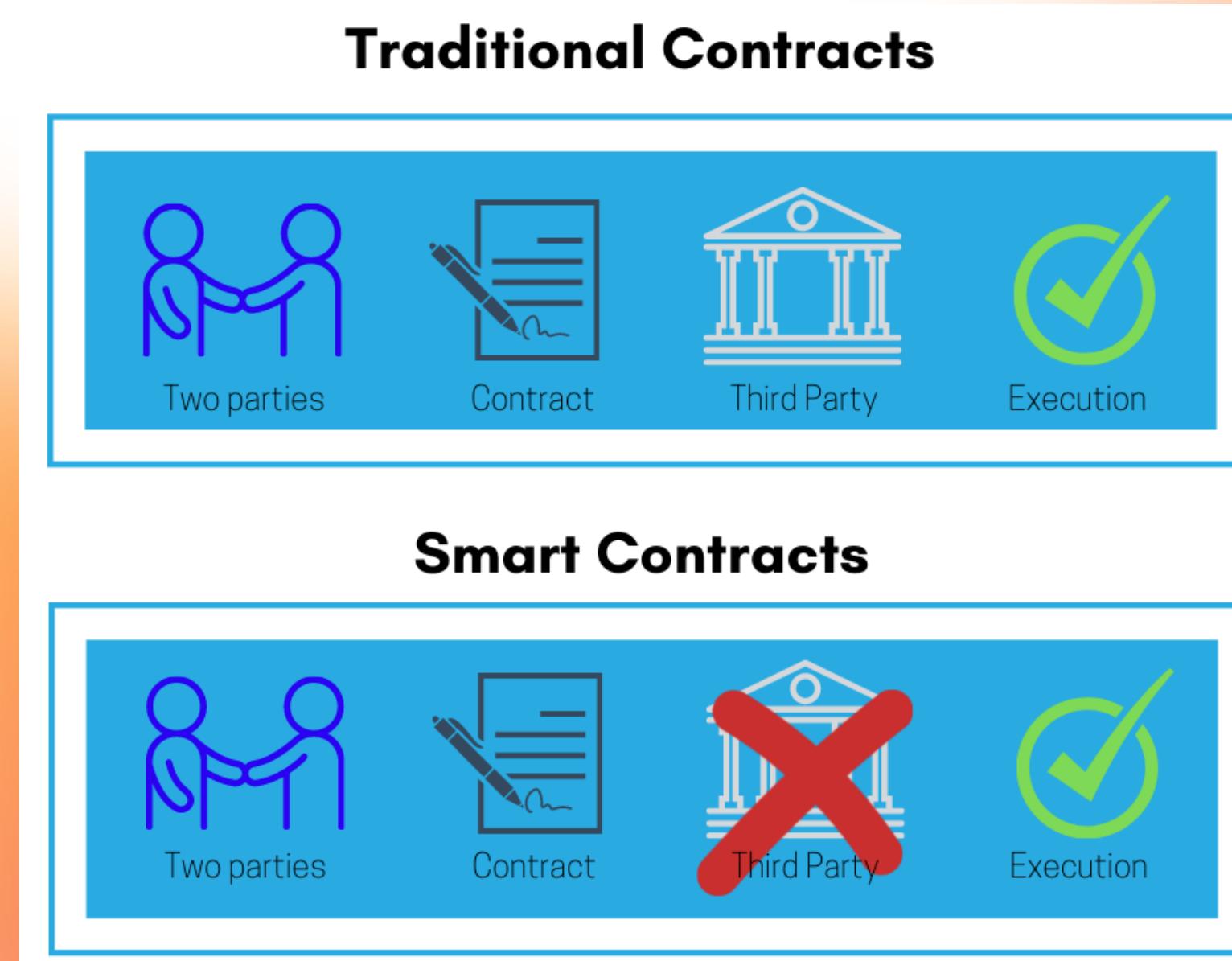


# Buổi 2:

# Giới thiệu Hợp đồng thông minh và Dapp



# HỢP ĐỒNG THÔNG MINH



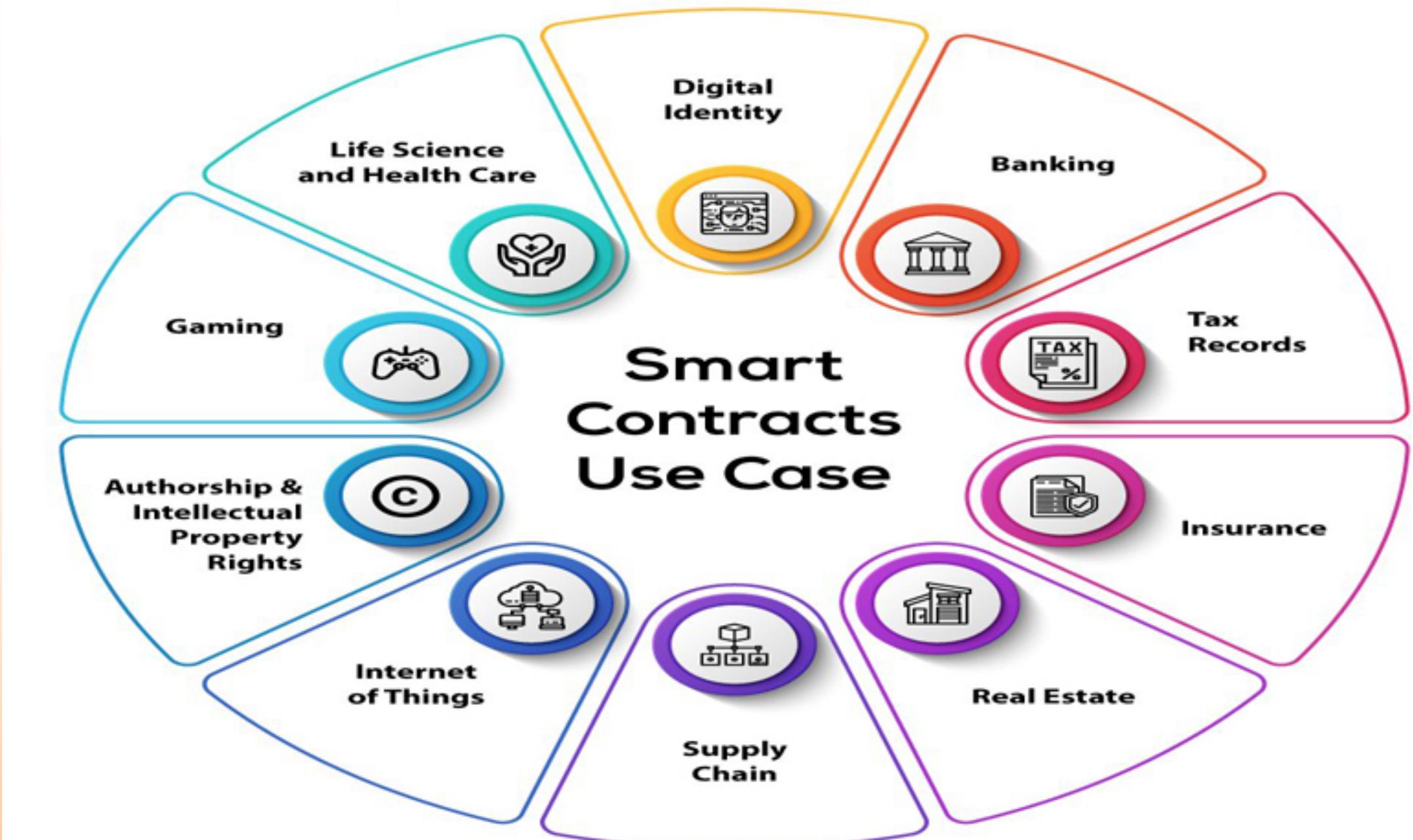
- Hợp đồng thông minh (Smart contract) là các **chương trình chạy trên blockchain**. Hợp đồng thông minh cũng giống như một **hợp đồng kỹ thuật số** bị bắt buộc thực hiện bởi một bộ quy tắc cụ thể. Các quy tắc này do bộ mã máy tính xác định trước mà **tất cả các nút (node) trong mạng đều phải sao chép và thực thi** các quy tắc đó.
- Smart Contract chỉ là một đoạn mã chạy trên một hệ thống phân tán, cho phép tạo ra **các giao thức Permissionless** (không cần trao quyền).

## CÁCH THỨC HOẠT ĐỘNG

### How Smart Contracts work?

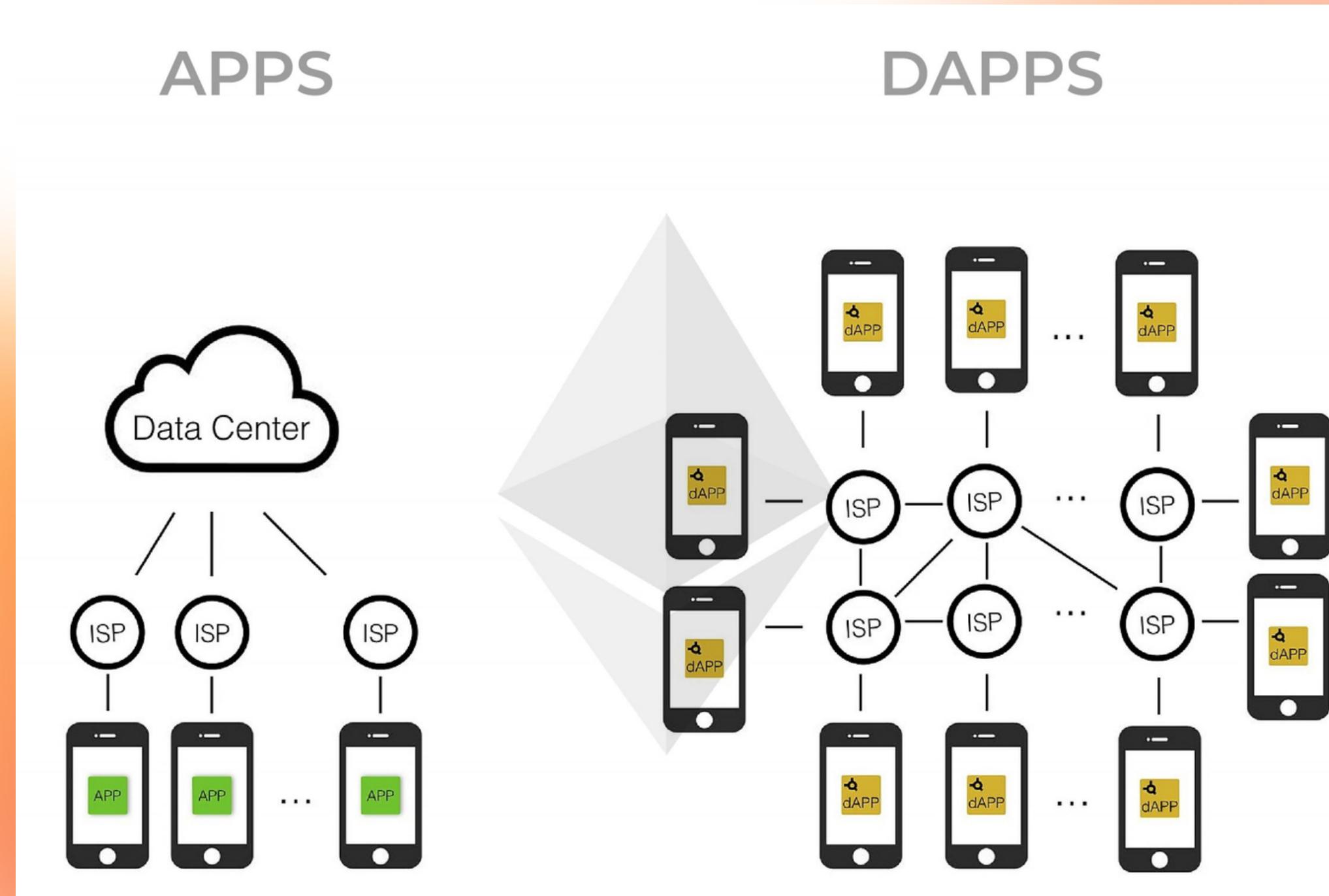


- Hợp đồng thông minh hoạt động như một **chương trình tất định**. Nó sẽ thực thi một tác vụ cụ thể trong trường hợp thỏa mãn các điều kiện nhất định.
- Hợp đồng thông minh chịu trách nhiệm **thực thi và quản lý các hoạt động** diễn ra trên blockchain khi những người dùng (address) tương tác với nhau



*Một số ứng dụng của Hợp đồng thông minh*

## ỨNG DỤNG PHI TẬP TRUNG (DAPP)



- Ứng dụng phi tập trung (Dapp) là các chương trình hoặc ứng dụng kỹ thuật số **chạy trên chuỗi khối hoặc mạng máy tính ngang hàng**.
- Các ứng dụng này về cơ bản được **phân cấp ở mức độ lớn** và **không thể bị kiểm soát bởi một cơ quan duy nhất**.

## TÍNH NĂNG CỦA DAPP



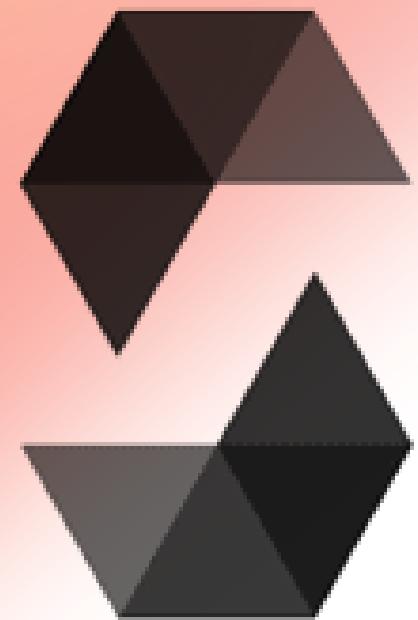
- **Mã nguồn mở:** Mã nguồn của ứng dụng có sẵn cho tất cả mọi người.
- **Quyền riêng tư:** Để triển khai và tương tác với DApp, user không cần cung cấp danh tính ở thế giới thực.
- **Phi tập trung và an toàn hơn nhờ mã:** Tất cả thông tin của DApp đều được bảo mật bằng mã và được lưu trữ trên một blockchain công khai, phi tập trung, được vận hành bởi nhiều người dùng (hoặc các node).
- **Vận hành một cách độc lập:** Dapp chạy độc lập mà không cần sự tham gia của bên thứ ba.

## CÁC DAPP TRÊN MẠNG KLAYTN



## NGÔN NGỮ SOLIDITY

- Solidity là một **ngôn ngữ bậc cao** do team Ethereum Network chịu trách nhiệm phát triển.
- Ngôn ngữ này thường được ứng dụng trong quá trình xây dựng - phát triển các Smart Contract **tự động hóa các giao dịch trên Blockchain hỗ trợ EVM**.
- Ngoài ra, Solidity còn là ngôn ngữ lập trình **theo hướng Contract**.



# SOLIDITY

## MỘT SỐ NGÔN NGỮ KHÁC

### Vyper

- Có nguồn gốc từ Python 3.
- Thường được dùng cho Máy ảo Ethereum (EVM), giống như Solidity.
- Vyper đã loại bỏ nhiều chức năng OOPS trong Solidity và hỗ trợ các tính năng khác như vòng lặp vô hạn, gọi đệ quy...
- Nhờ vậy, các vấn đề bảo mật phát sinh cũng được tránh khỏi nhờ các tính năng này.

### Yul

- Là ngôn ngữ biên dịch trung gian thành byte code cho máy ảo EVM.
- Là ngôn ngữ cấp thấp, tương tự Assembly.
- Có thể tích hợp viết trong Solidity.



## CÔNG CỤ HỖ TRỢ

- **Remix Online IDE:** là một công cụ online hỗ trợ các bạn muốn lập trình Smart Contract ngay lập tức. Remix hỗ trợ viết Smart contract và triển khai bằng Klaytn Plugin.
- **Hardhat:** Hardhat là một môi trường phát triển để biên dịch, triển khai, test và debug smart contract với nhiều plugin/libraries hỗ trợ.
- **Truffle:** Truffle là một công cụ để phát triển và triển khai smart contract trên blockchain. Truffle cung cấp các tính năng như tích hợp với các môi trường phát triển, các công cụ để test và triển khai smart contract. Hiện Klaytn hỗ trợ đến Truffle bản v5.0.26.
- **Kaikas:** Là một browser extension wallet cho mạng Klaytn.
- **Klaytnscope:** Là một block explorer trên mạng Klaytn (giống Etherscan).



# KIỂU DỮ LIỆU

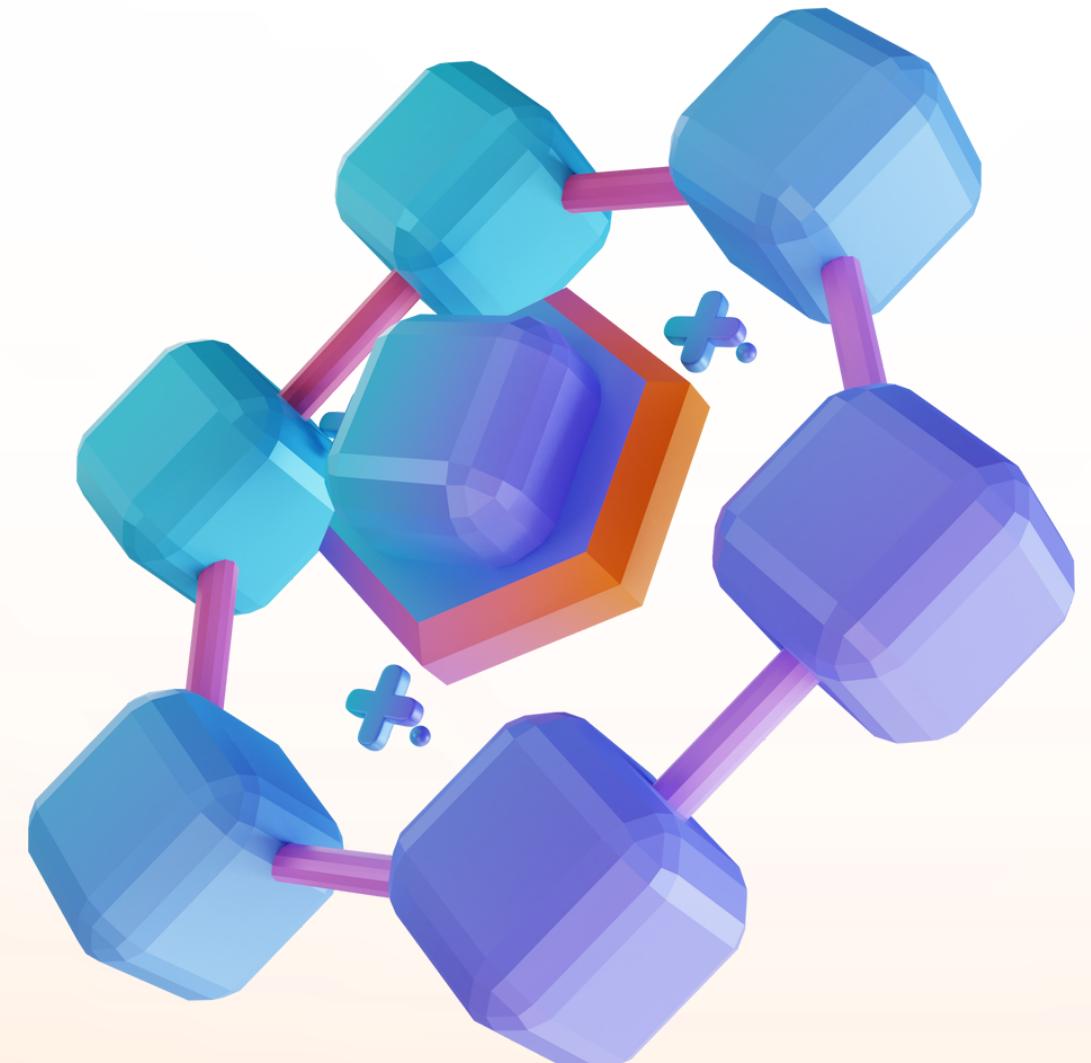
## Số nguyên (Integer)

Kiểu dữ liệu này có thể lưu trữ từ 8 đến 256 bit giá trị:

- **signed**: bao gồm số âm và số dương: **int**
- **unsigned**: chỉ gồm các số nguyên dương: **uint**

Kiểu dữ liệu số nguyên có thể thực hiện các phép toán sau:

- So sánh `<=`, `<`, `==`, `!=`, `>=`, `>`
- Phép toán bit `&` (and), `|` (or), `^` (bitwise exclusive), `~` (bitwise negation)
- Toán tử số học `+` (cộng), `-` (trừ), `*` (nhân), `/` (chia), `%` (lấy dư), `**` (lũy thừa)



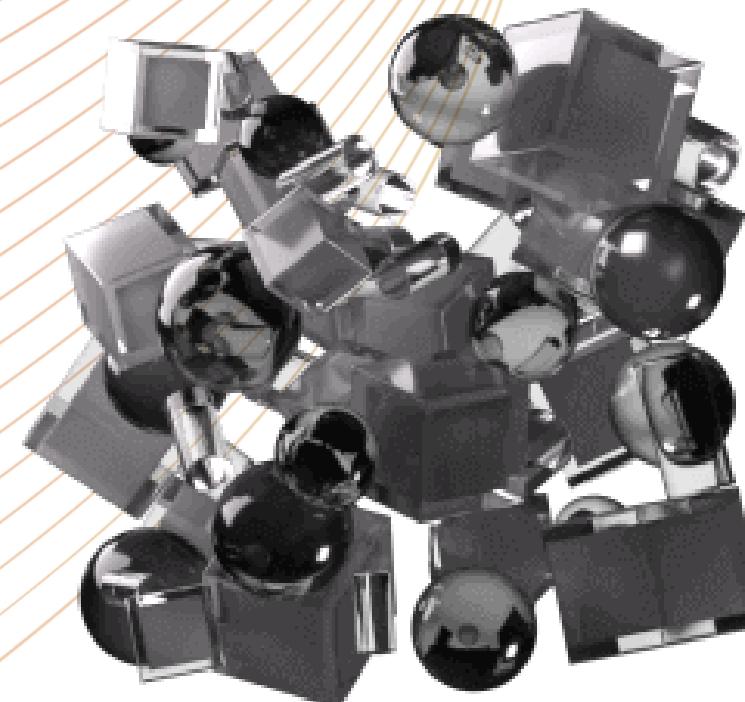
## KIỂU DỮ LIỆU

### Boolean

- Khai báo bằng từ khóa bool. Chúng chỉ có 2 giá trị là true hoặc false
- Booleans thường được sử dụng trong so sánh các điều kiện
- Và cũng có thể sử dụng như tham số hàm và định nghĩa kiểu dữ liệu trả về

### String

- Chuỗi ký tự được dùng ở hầu hết các contract, và được bọc bởi nháy đơn hoặc nháy kép.



## KIỂU DỮ LIỆU

### Address

- Address là kiểu dữ liệu sử dụng 20-byte thể hiện địa chỉ tài khoản Ethereum.
- Address payable là địa chỉ mà bạn có thể gửi Ether tới, và nó bao gồm hai phương thức transfer và send

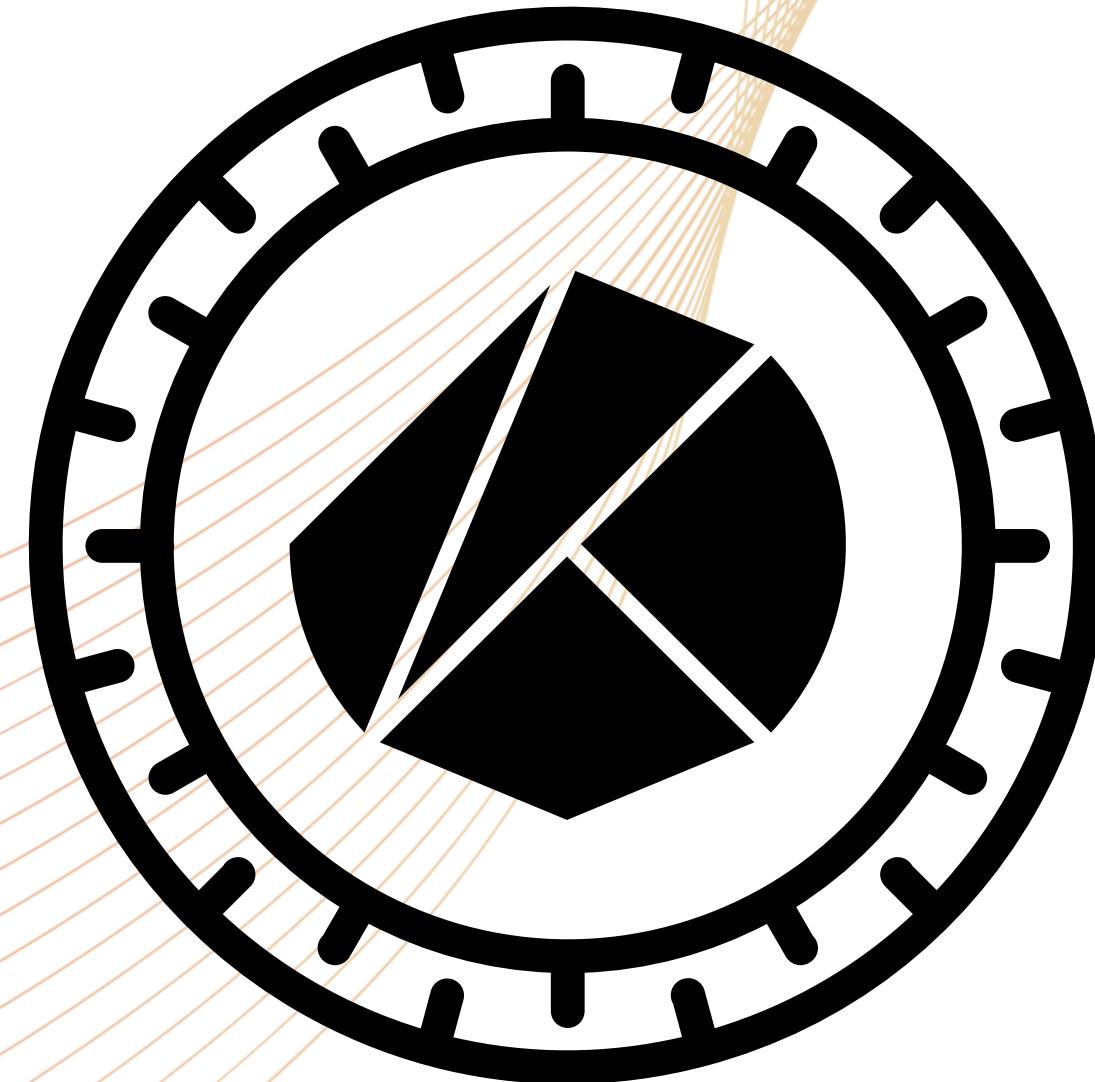
### Enums

- Trong Solidity, bạn có thể sử dụng enums để tạo kiểu dữ liệu tự định nghĩa.



## BIẾN

- **State Variables:** Biến lưu trữ tạm thời trong bộ nhớ contract.
- **Local Variables:** Biến có giá trị khi một hàm thực thi.
- **Global Variables:** Biến tồn tại trong không gian chung của blockchain



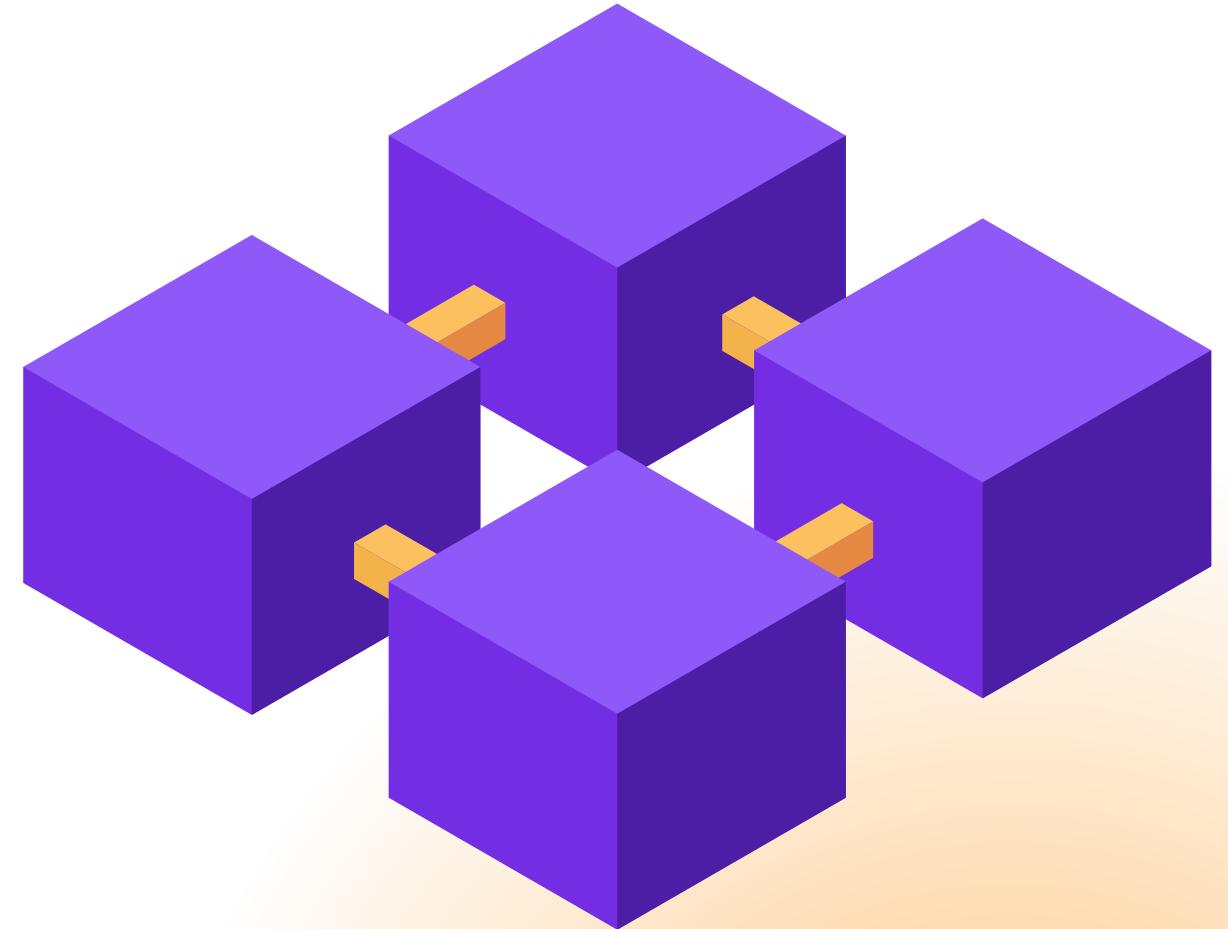
Tên	Giải thích
block.number (uint)	Số block hiện tại
block.timestamp (uint)	Thời gian hiện tại tính theo giây (theo chuẩn epoch)
msg.sender (address payable)	Địa chỉ gọi đến contract
msg.value (uint)	Giá trị wei gửi đến contract
now (uint)	Thời gian hiện tại của block
tx.origin (address payable)	Địa chỉ thực hiện giao dịch

*Các biến global thường dùng*



## PHẠM VI CÁC BIẾN

- **Public:** Giống với các ngôn ngữ hướng đối tượng khác, biến public được truy cập từ trong contract hoặc bên ngoài, khi định nghĩa biến là public một method get tự động được tạo ra để truy xuất thông tin của biến từ bên ngoài contract.
- **Internal:** Biến Internal được truy cập từ Contract chứa nó và các Contract kế thừa
- **Private:** Biến private chỉ được truy cập ở Contract mà nó được định nghĩa



## LƯU TRỮ BIẾN

memory:

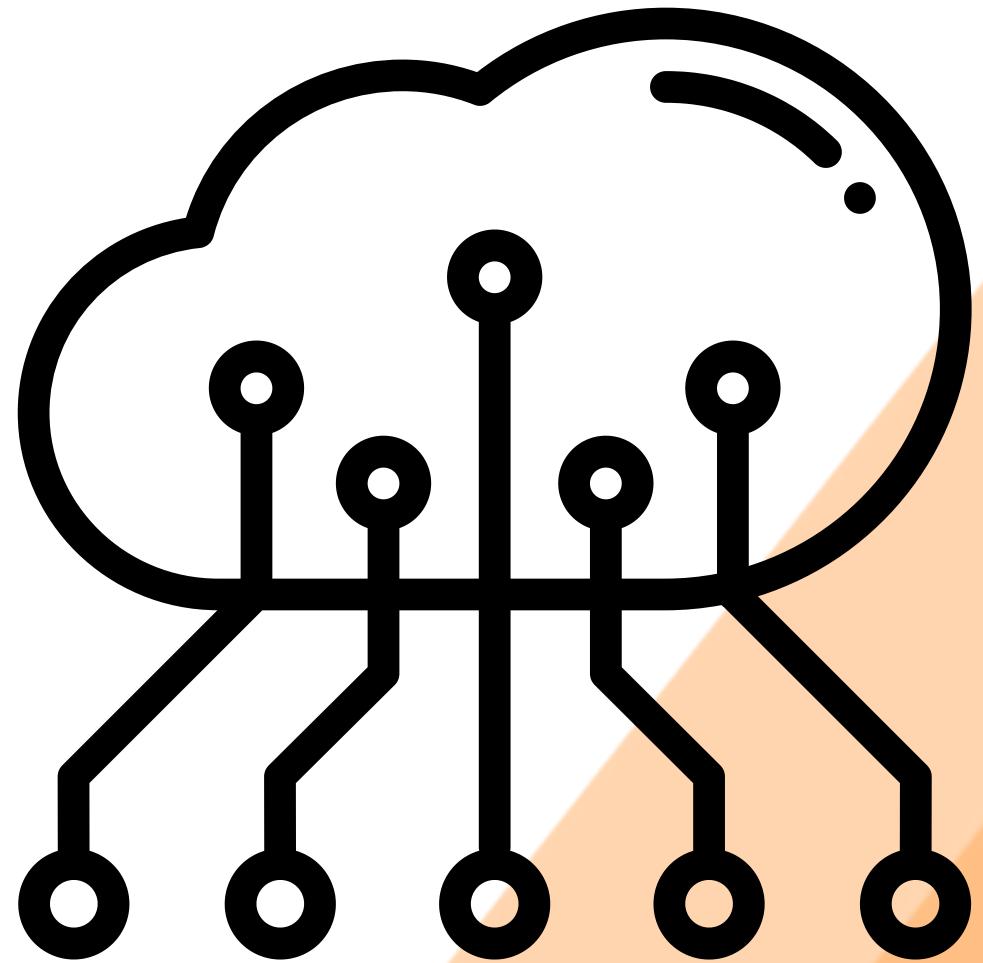
- Vị trí lưu là các đối số của hàm.
- Bị giới hạn với hàm gọi bên ngoài.

storage:

- Vị trí lưu trữ các biến trạng thái.
- Tồn tại giới hạn trong thời gian contract.

calldata:

- Vị trí lưu trữ các đối số của hàm.
- Bắt buộc đối với các tham số của hàm bên ngoài, nhưng cũng có thể được sử dụng cho các biến khác.
- Bị giới hạn với hàm gọi bên ngoài.

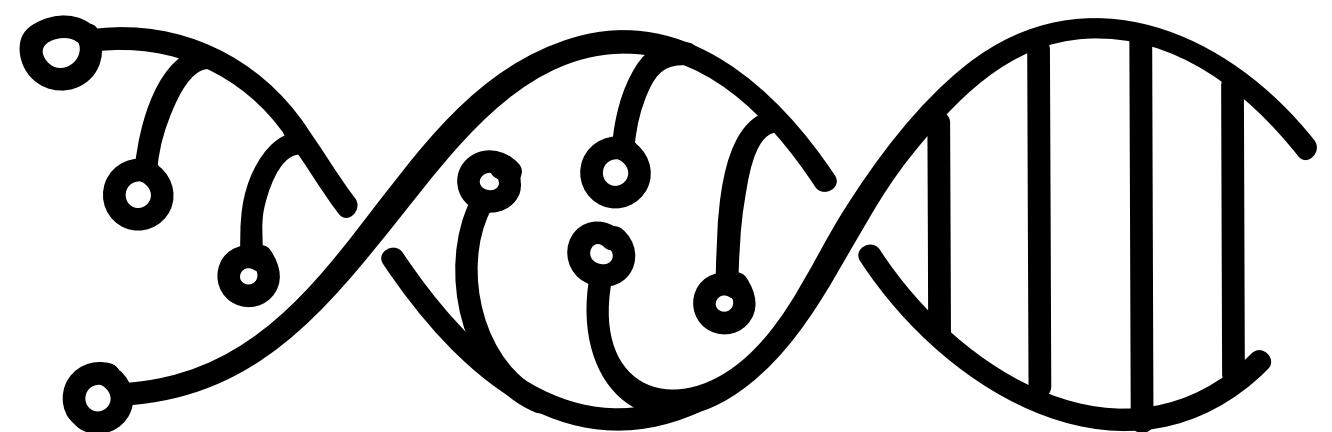


## HÀM

Hàm là các đoạn mã thực hiện một nhiệm vụ cụ thể. Các hàm có thể lấy bất kỳ số lượng biến nào làm đầu vào và nó có thể trả về một giá trị cụ thể. Với Solidity, bạn thậm chí có thể trả về nhiều giá trị từ một hàm.

### Cú pháp:

```
function <tên_func>(<type> <var_name>,...) <scope> <behavior> returns(<type>  
<var_name>)  
{...}
```



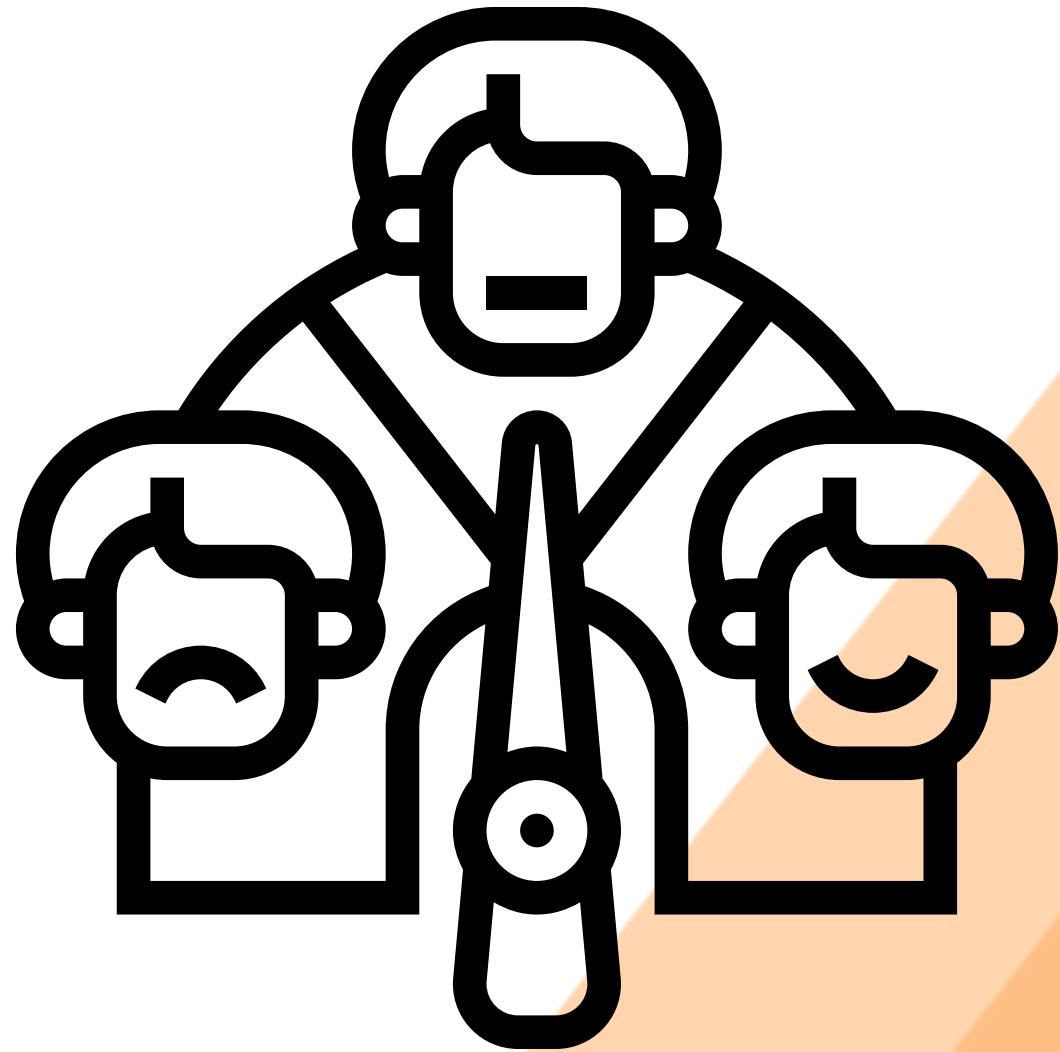
## PHẠM VI HÀM

- **Public:** hàm được gọi từ trong contract hoặc bên ngoài
- **Private:** hàm chỉ được gọi ở Contract mà nó được định nghĩa
- **Internal:** hàm được gọi từ Contract chứa nó và các Contract kế thừa
- **External:** tương tự public, hàm được gọi từ bên ngoài Contract



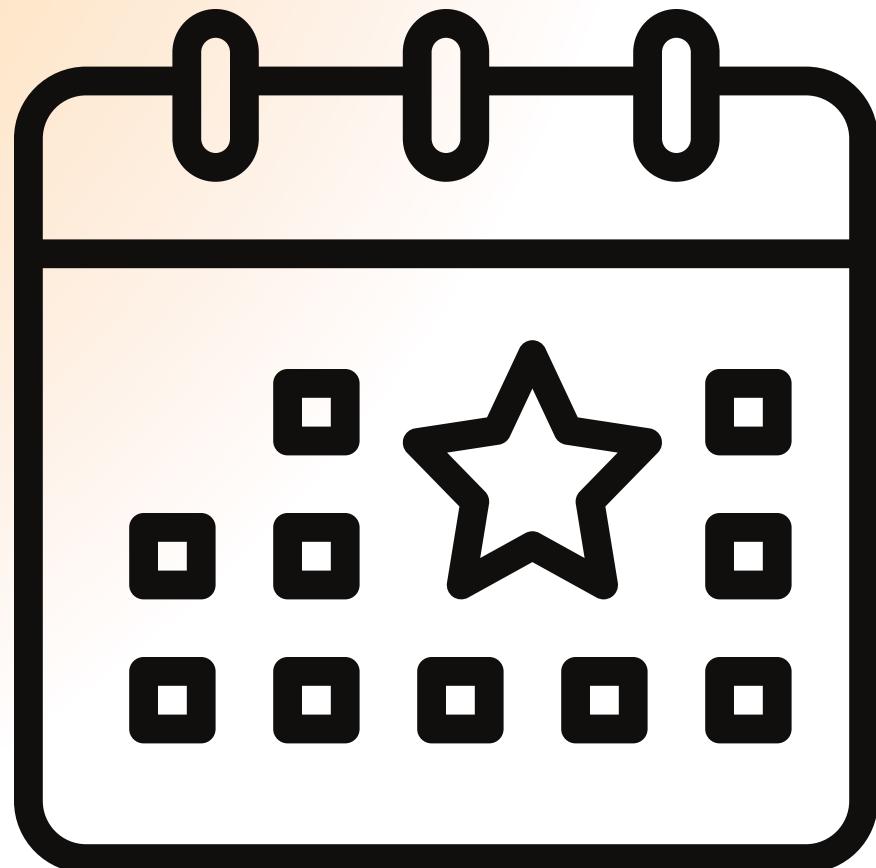
## HÀNH VI HÀM

- **View:** hàm chỉ được đọc các biến trong contract, các biến trạng thái không thể bị sửa đổi sau khi gọi chúng.
- **Pure:** Các hàm thuần túy không đọc hoặc sửa đổi các biến trạng thái, mà chỉ trả về các giá trị bằng cách sử dụng các tham số được truyền cho hàm hoặc các biến cục bộ có trong nó.
- **Payable:** hàm khi gọi có thể truyền vào giá trị wei



## EVENT

- 1 cách Smart contract giao tiếp với bên ngoài
- Event thường được sử dụng để broadcast ra cho phía client biết rằng hàm nào đó đã được gọi và thực thi.



## KẾ THỪA

Solidity hỗ trợ cả kế thừa đơn lẻ cũng như đa kế thừa.

- Một hợp đồng kế thừa có thể truy cập tất cả các public, internal function và các biến trạng thái công khai
- Cho phép ghi đè chức năng



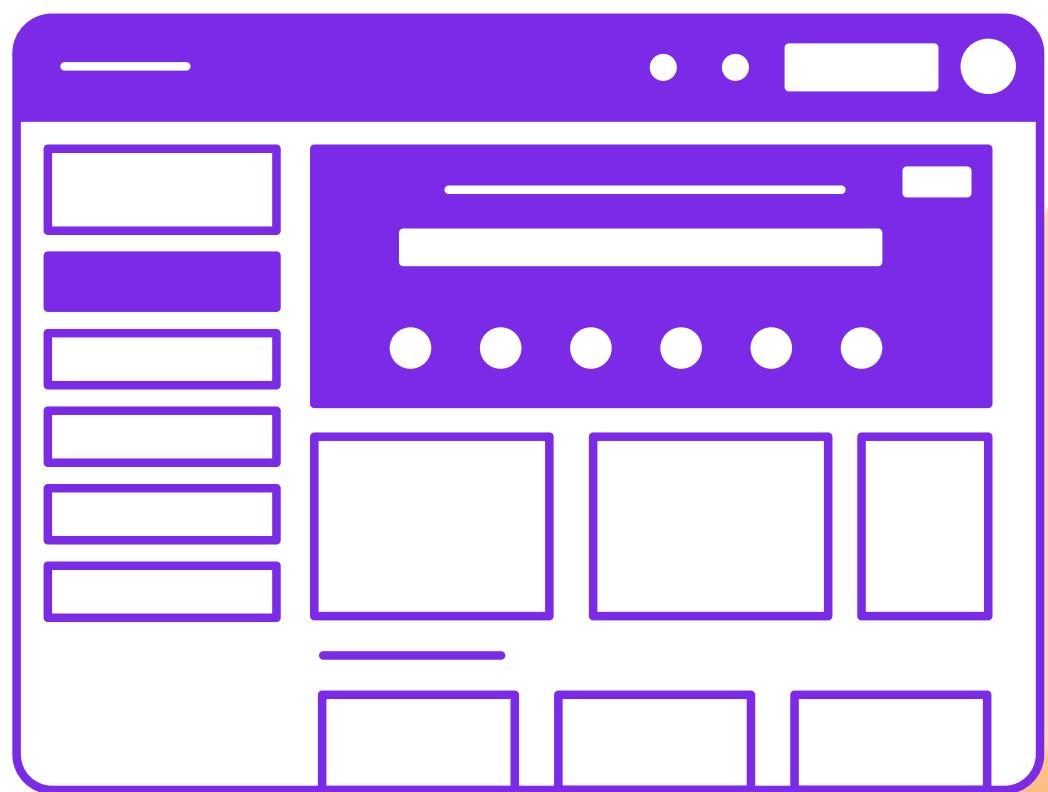
## ABSTRACT

- Khá giống Interface nhưng nó có thể làm nhiều việc hơn.
- Có 2 loại method là abstract method và method thường:
- abstract method là method trống không có thực thi.
- method thường là method có thực thi.
- Các lớp chỉ có thể kế thừa một Abstract class
- Hướng đến tính năng và những tính năng có thực thi được sử dụng làm hàm chung cho các class extend.



## INTERFACE

- Không phải là contract.
- Chỉ chứa những method/properties trống không có thực thi.
- Các contract có thể implements nhiều interface.
- Contract implement phải triển khai các method theo như interface đã định nghĩa.



# **BIÊN DỊCH VÀ GỠ LỖI**



TIẾP THEO

**Solidity cho người mới: Cơ  
bản về hợp đồng thông minh**