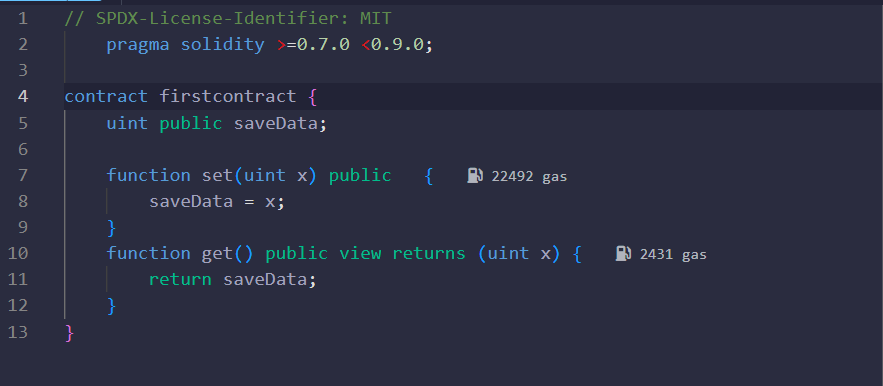
**Học về Solidity**

**Bài 1: Solidity và chương trình cơ bản**

**1.** Solidity là ngôn ngữ lập trình chuyên dụng để phát triển smart contract (hợp đồng thông minh) trên blockchain Ethereum và các mạng tương thích với Ethereum Virtual Machine (EVM). Đây là một ngôn ngữ lập trình bậc cao, hướng đối tượng với cú pháp tương tự JavaScript, C++ và Python, nhưng được thiết kế riêng để xử lý logic blockchain một cách an toàn và hiệu quả.

IDE web : https://remix.ethereum.org/

**2. Chương trình cơ bản (set giá trị và get giá trị):**



**Giải thích code:**

1. **pragma solidity >=0.7.0 <0.9.0;**  
   Dòng này khai báo phiên bản solidity được sử dụng từ phiên bản 0.7.0 đến dưới phiên bản 0.9.0
2. **contract firstcontract {**  
   khai báo, định nghĩa 1 contract (giống như class trông OOP) với tên là “firstcontract”
3. **uint public saveData;**Khai báo 1 biến tên là saveData với kiểu dữ liệu là uint(unsigned interger 256-bit, giá trị nguyên >=0, trong solidity có thể ghi uint8 từ 0 ->255(2^8),… ), public: “**Visibility modifier”** giá trị này được công khai.
4. **function set(uint x) public{  
    saveData = x;  
   }**Tạo hàm set, hàm này cho phép có thể thay đổi giá trị của biến saveData và public thì ai cũng có thể thay đổi giá trị.
5. **function get() public view returns (uint x){  
    return saveData;  
   }**Tạohàm get, hàm này cho phép lấy giá trị của biến saveData hay trả về giá trị của saveData mà không làm thay đổi trạng thái(có thể nói dễ hiểu là giá trị) của contract.

**Giải thích sâu:**

1. Trong Solidity, **function** được định nghĩa bằng từ khóa **function** theo sau là **tên function**, **danh sách tham số, modifier** và **kiểu trả về**. Cú pháp tổng quát là: **function function\_name(parameter\_list) visibility state\_mutability returns(return\_type) {**

**// code logic**

**}**

1. **Return và returns.**

**Vai trò khác nhau hoàn toàn:**

* **returns:** Khai báo function sẽ trả về gì (trong signature)
* **return:** Thực hiện việc trả về giá trị (trong body)

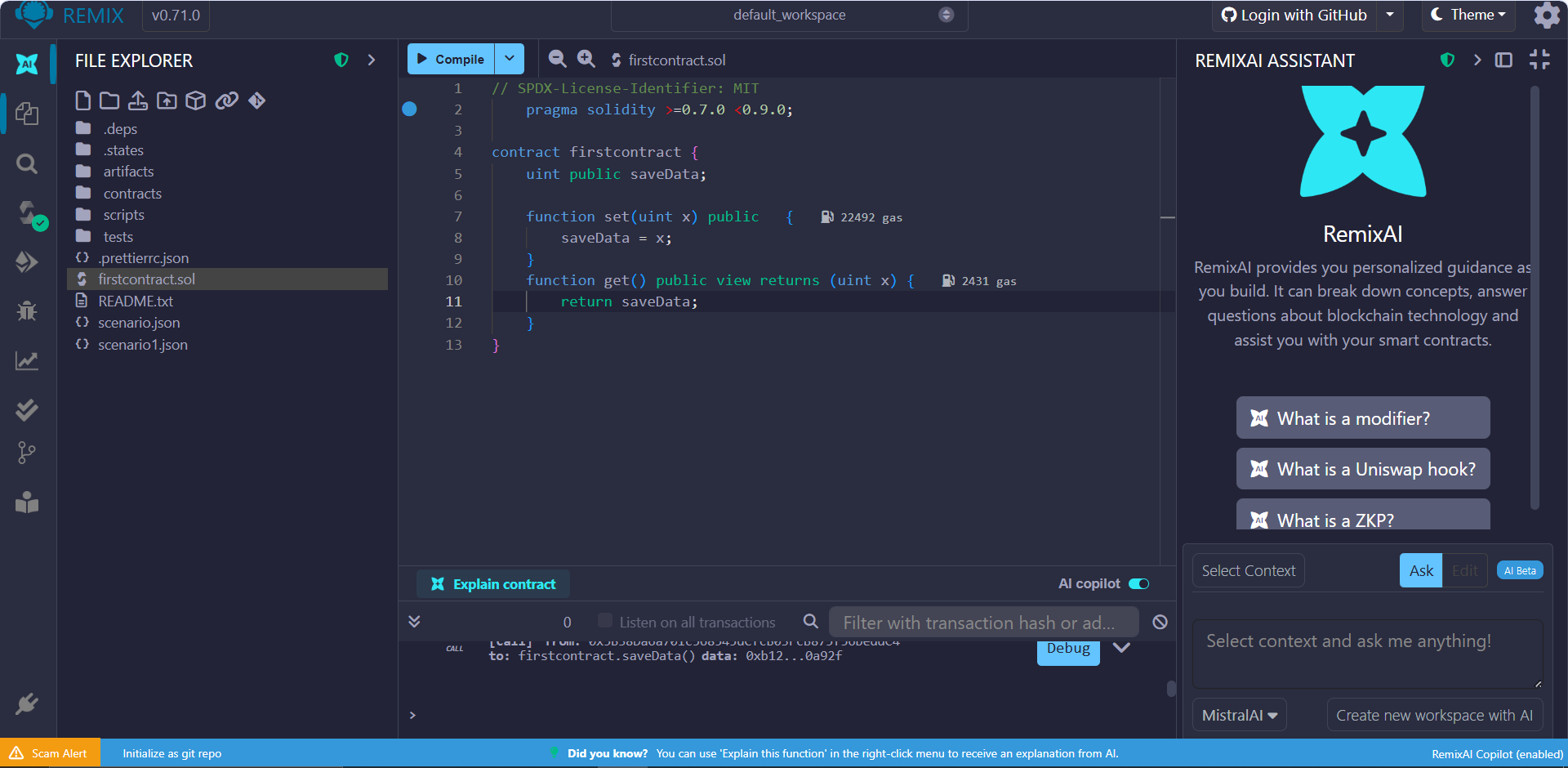
**Vị trí khác nhau**:

* **returns:** Nằm **ngoài** dấu {} của function
* **return:** Nằm **bên trong** dấu {} của function body

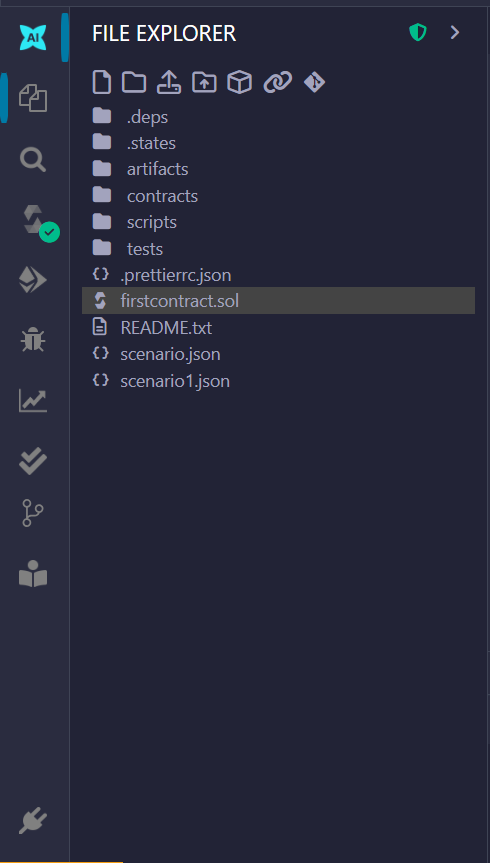
**returns** là **khai báo kiểu** (declaration), còn **return** là **câu lệnh thực thi** (statement). Cả hai đều cần thiết - returns để compiler biết function trả về gì, return để thực sự trả về giá trị đó.

**3. IDE remix.ethereum (Remix IDE)** là 1 **môi trường phát triển tích hợp** và **chạy trực tiếp trên nền tảng trình duyệt web**, chuyên dùng để viết và biên dịch các **smart contract** trên nền tảng **Ethereum**

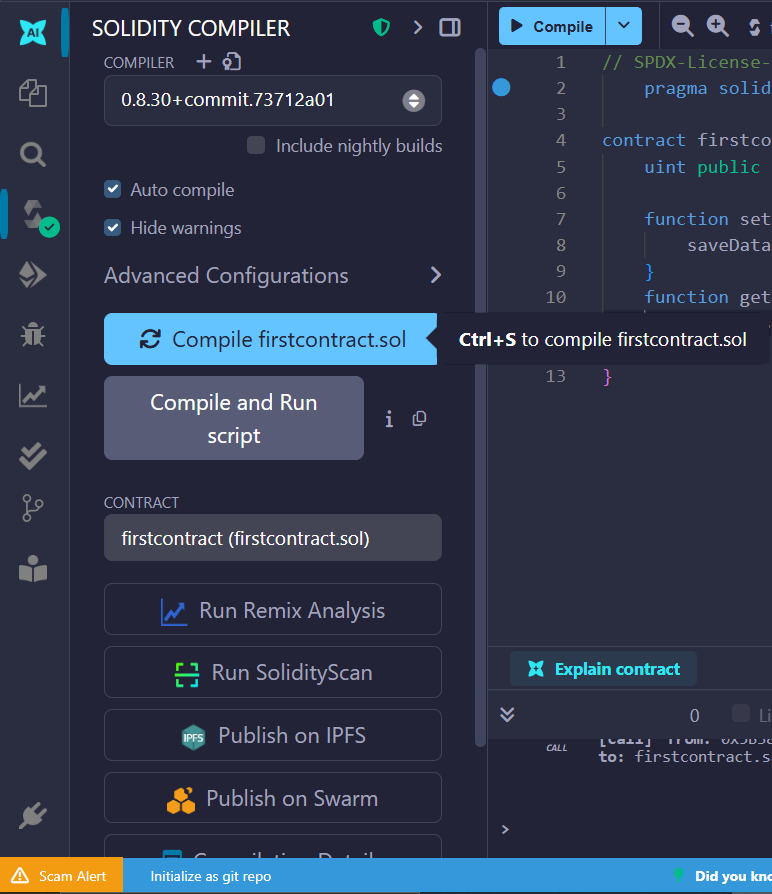
**Giao diện làm việc của IDE:**

****

**Thanh Sidebar:**

****

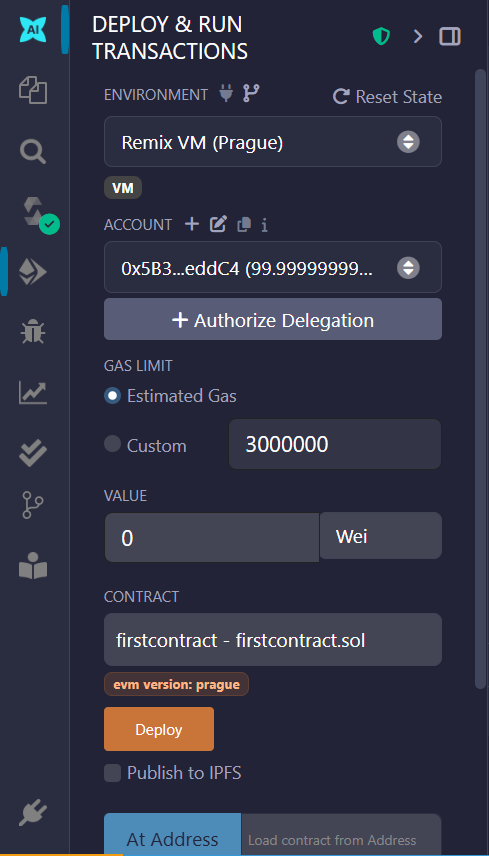
**Compile contract:**

****

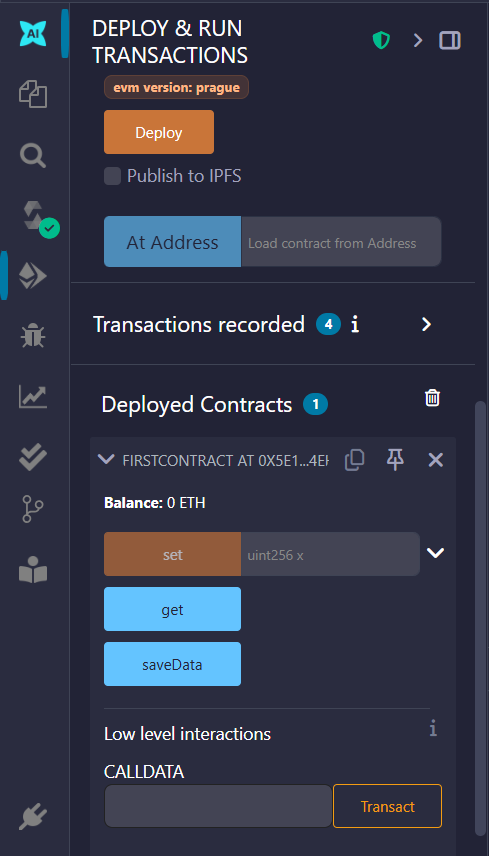
Solidity Compiler có nhiệm vụ **biên dịch** (compile) code Solidity thành bytecode, không liên quan đến việc đẩy lên blockchain. Cụ thể:

* **Input**: Code Solidity (.sol files) mà con người viết
* **Output**: Bytecode (mã máy) mà Ethereum Virtual Machine hiểu được
* **Kèm theo**: ABI (Application Binary Interface) để tương tác với contract

**Deploy & run transactions:**

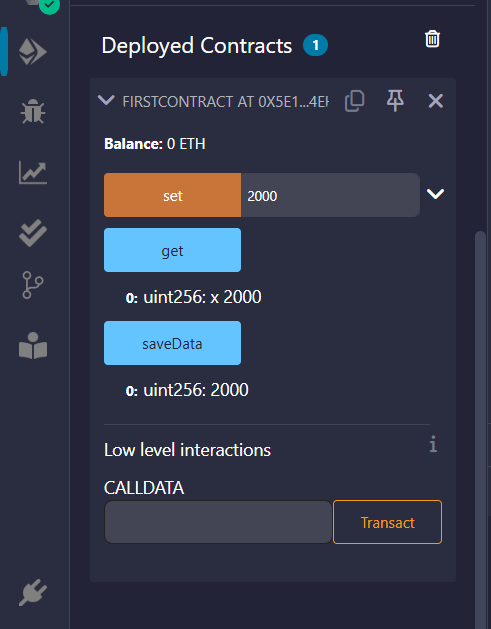
****

Chọn **Environment**(môi trường) và **account**(tài khoản) để triển khai contract lên blockchain, có sẳn hơn **10 tài khoản** với khoản **100ETH cho mỗi tài khoản. Deploy** để gửi **trainsaction** lên blockchain và sẽ nhận **contract address** sau khi deploy thành công**.**



At address(truy cập vào contract đã có). Kết nối với contract đã được deploy lên trước đó bằng address

**Tương tác với Deploy contract:**



**Button xanh(blue):** **View/Pure function** – chỉ đọc, lấy dữ liệu và không tốn **gas**(**tài nguyên** của các smart contract trên Ethereum, giúp vận hành hiệu quả và tránh bị lạm dụng hệ thống).

**Button cam(orange):** **Non-payable functions** – thay đổi **state**(trạng thái), tốn **gas.**

**Button đỏ(red):** **Payable functions** – có thể gửi **ETH** kèm theo.

**Bài 2: Tạo token đầu tiên**

Muốn giao dịch token thì cần phải:

1. **Create a token/Coin**
2. **Create amount token/Coin**

* Minter(người tạo)
* Suply(lượng cung)
* Balance(Ví, số dư)

1. **Sent Token**
   * Receiver(Người nhận)
     1. If Amount > Balance then không đủ tiền để chuyển
     2. Balance sender -= amount;
     3. Balance receiver += amount;
   * Amount(Khối lượng)