

# Comprendre les smart contracts



# Comment écrire un smart contract 1/2

- ✦ Un smart contract sur EVM peut être écrit dans deux types de langages :
  - ✦ Solidity (que nous allons voir ici) qui est basé sur du javascript
  - ✦ Vyper basé sur du python



# Comment écrire un smart contract 2/2

- ✦ Pour commencer nous allons utiliser un IDE en ligne qui nous permet de travailler avec les fonctions. Disponible a l'adresse [remix.ethereum.org](https://remix.ethereum.org).
- ✦ Puis dans la partie avancée nous utiliserons visual studio code.



# Compiler / publier un smart contract

- ✦ Les langages solidity & vyper sont des languages qui doivent être compilés.
- ✦ Donc il faudra toujours spécifier dans le navigateur la version du langage que nous allons utiliser.
- ✦ Une fois compilé notre smart contract devra être envoyé sur la blockchain par l'intermédiaire du déploiement.

✦



# Structure d'un smart contract

- ✦ Un smart contract comporte toujours en entête sous forme de commentaire la license (se référer aux licenses SPDX)
- ✦ Et la version du compilateur utilisé :

- ✦

```
1 // SPDX-License-Identifier: GPL-3.0
2 pragma solidity 0.8.1;
3
```


- ✦



# Structure d'un smart contract episode 2

- ✦ Un smart contract doit toujours commencer par contract avec le nom du contrat en majuscule et tout son contenu sera entre brackets {}



```
1 // represente le type de licence à appliquer en général MIT
2 // SPDX-License-Identifier: GPL-3.0
3
4 // defini la version du compilateur à utiliser.
5 pragma solidity 0.8.1;
6
7 // nom du contrat
8 contract Hello {
9     // definition d'une variable string
10    string hello = "hello world";
11
12    //la fonction qui va chercher le contenu de la variable hello cette fonction est pu
13    // toutes les fonctions qui n'effectuent pas d'operations sont des fonctions de typ
14    function sayHello() public view returns (string memory) {  infinite gas
15        //la fonction retournera donc le contenu de hello.
16
17        return hello;
18    }
19 }
```



# Structure d'un smart contract s1ep3

- ✦ Il est conseillé de mettre les variables ayant une portée sur tout le contrat en premier. Les fonctions viendront par la suite.

```
1 // SPDX-License-Identifier: GPL-3.0
2 pragma solidity 0.8.1;
3
4 contract Increment {
5     uint256 public toto = 42;
6
7     function incToto() public {  ⚠ infinite gas
8         toto += 1;
9     }
10
11     function decToto() public {  ⚠ infinite gas
12         toto -= 1;
13     }
14 }
15
```