

Summer School of Solana

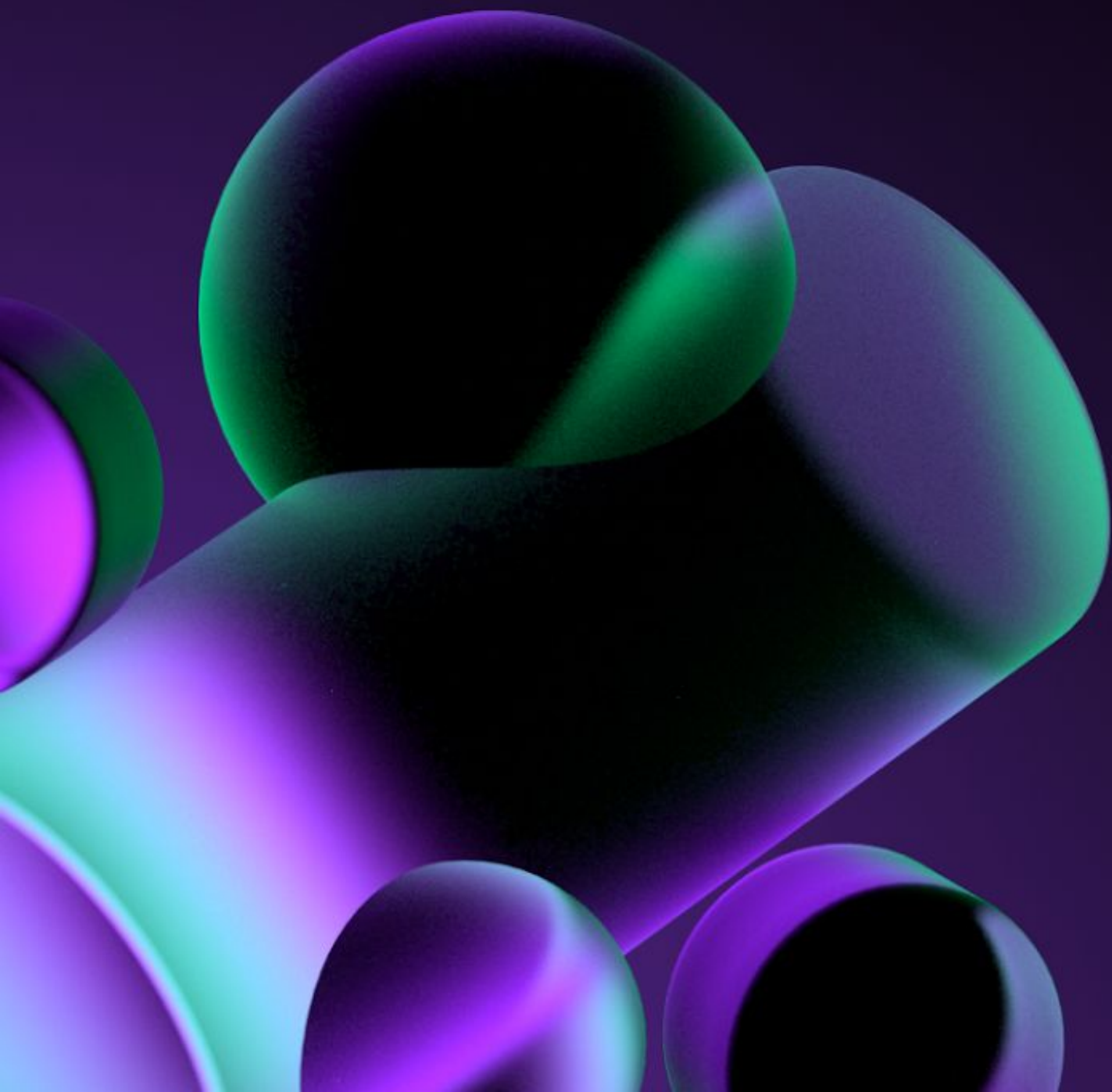
LECTURE 5

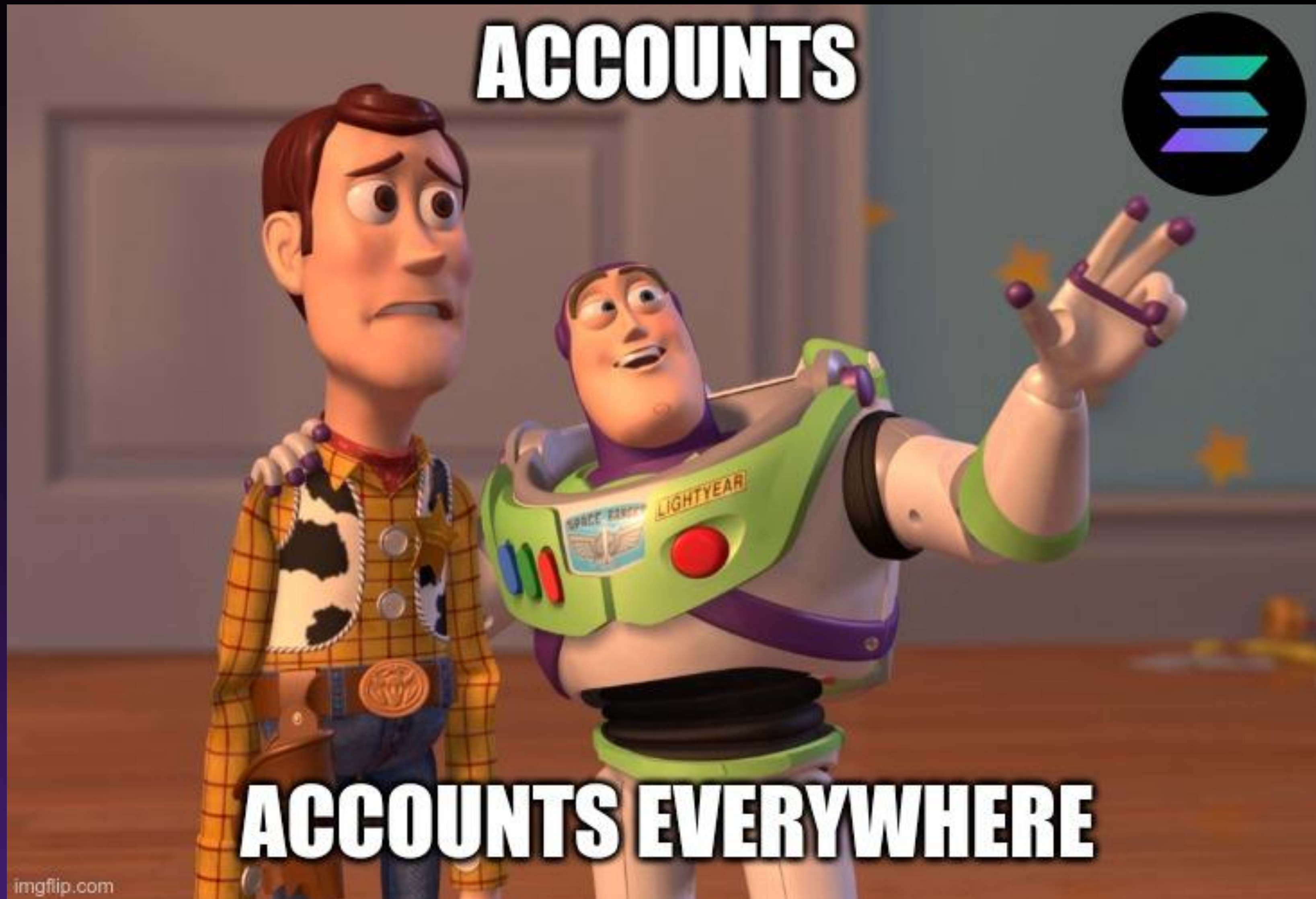
Solana Programming model 3/3

About this lecture

- Solana Programming model - recap
 - Recap
 - Program Derived Addresses (PDAs)
 - Tokens
- Hands on Example
 - Turnstile program (fixes)

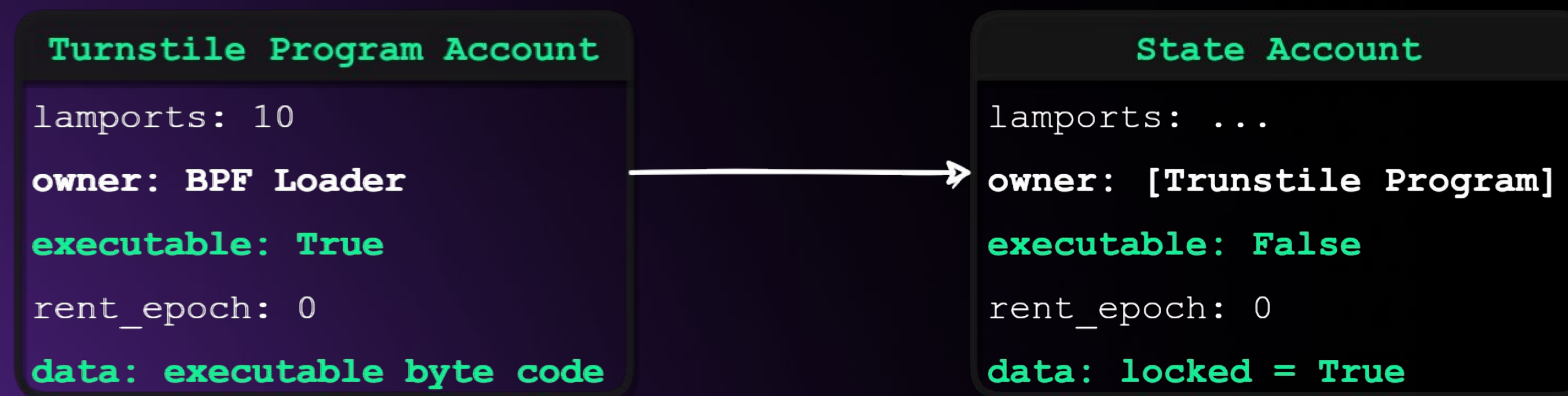
Recap





Accounts

- Solana **accounts**:
 - Program accounts
 - Data accounts
- Program accounts **do not store state**!
- Only a data account's **owner** can **modify its data** and **subtract lamports**.
- To prevent an account from being deleted, **you must pay rent**.



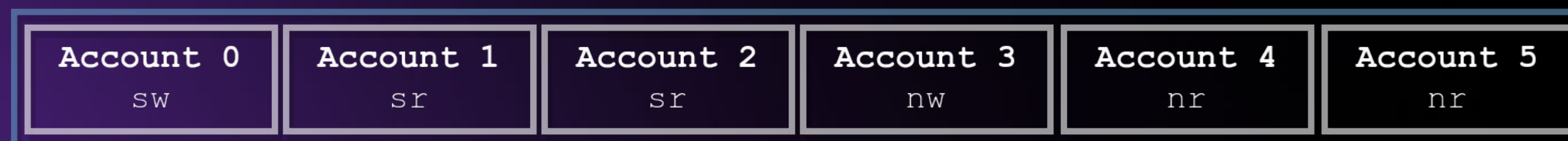
Transactions

- The basic operational unit on Solana is an **instruction**.
- One or more instructions can be bundled into a **transaction**.
- Instructions in one transaction are processed **in order** and **atomically**.
- You must **forward-declare every account** you intend to read from or write to.
 - be aware of potential **SECURITY RISKS**

Instruction Format



s - signed
n - not signed
w - writable
r - read only



PDA's



pencilflip.sol 🍄
@pencilflip

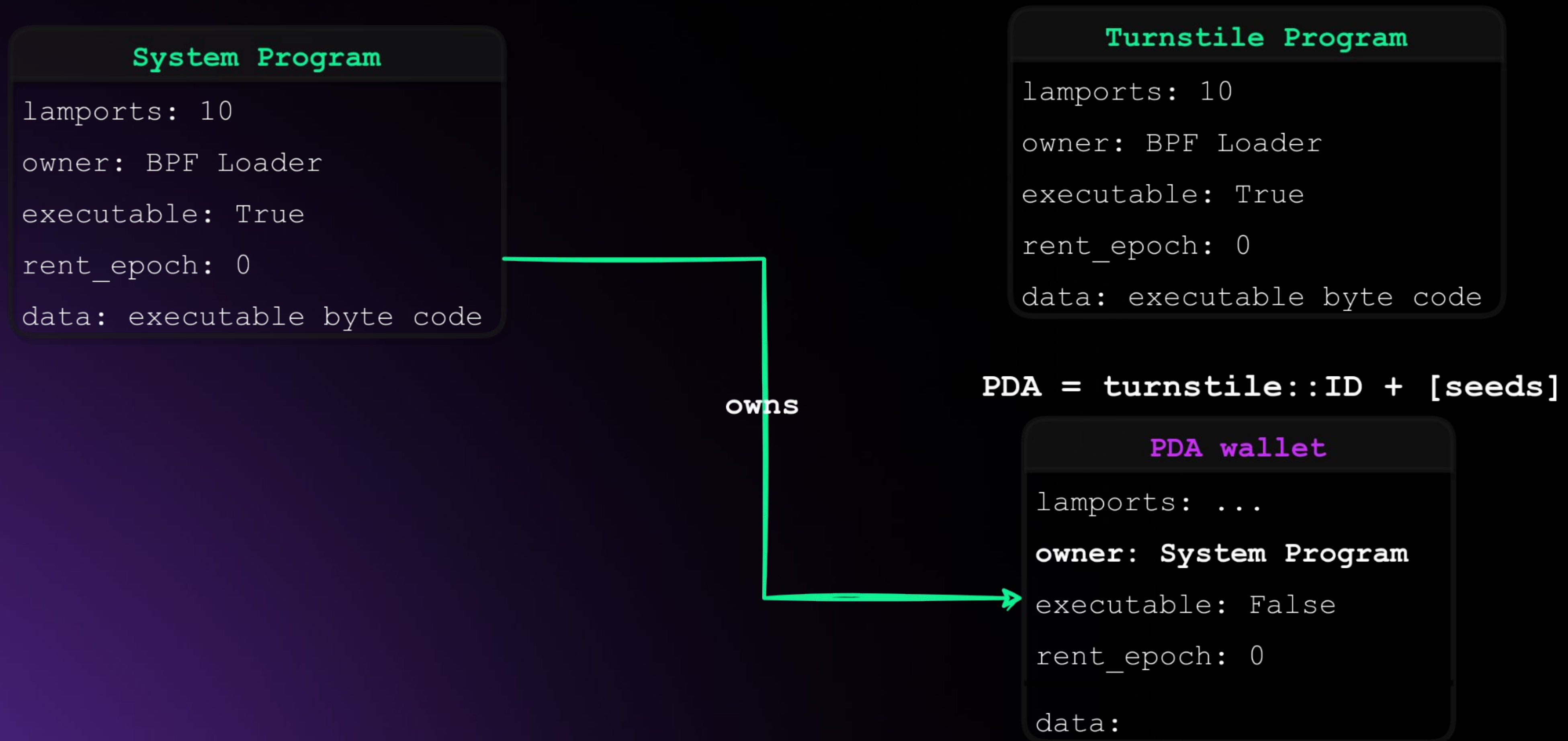
...

PDA's, or program derived addresses, are one of the trickier [#Solana](#) 🍷 concepts 🤔. They're also something that every Solana dev should understand.

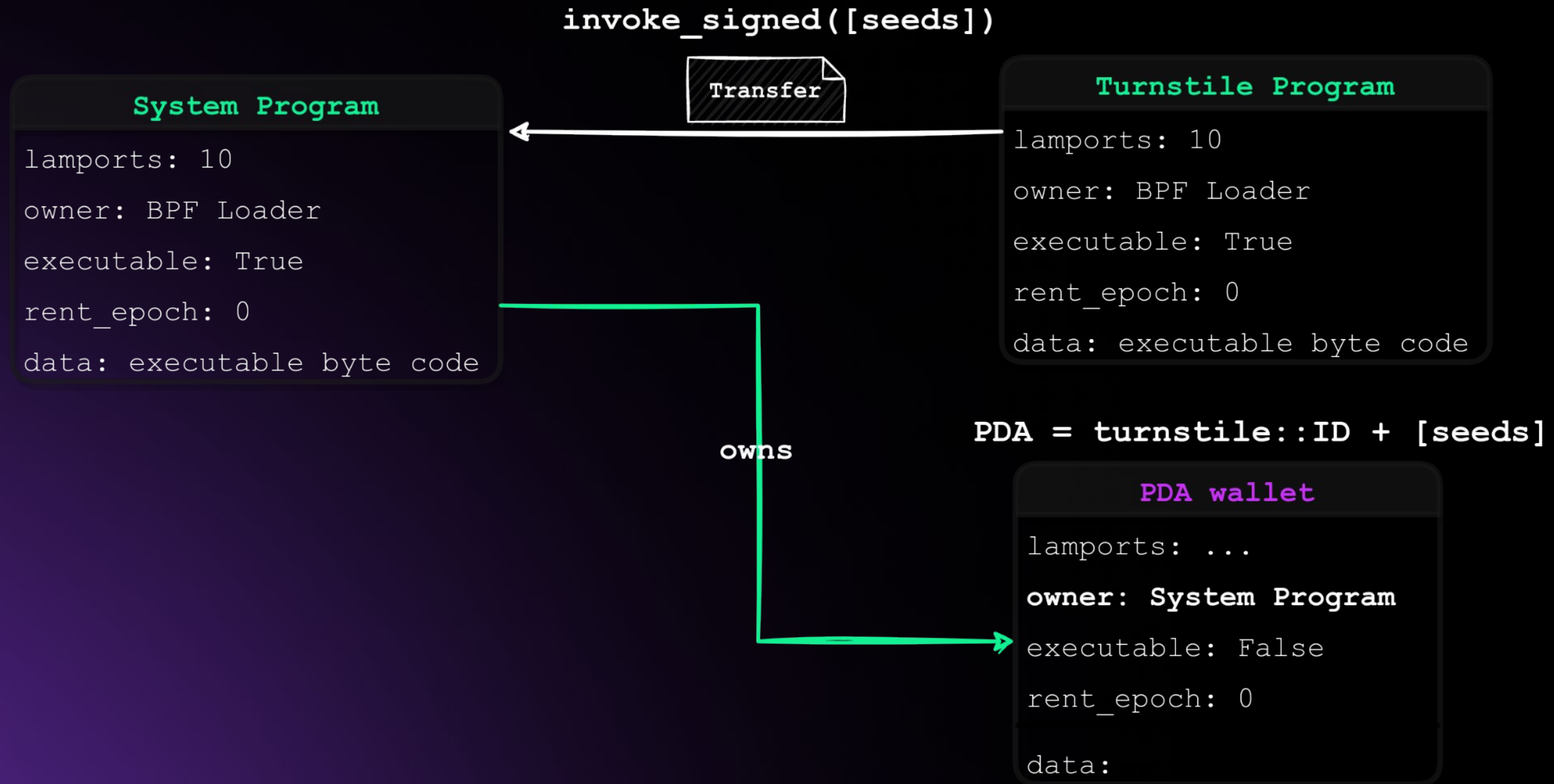
PDAs motivation



PDA (invoke_signed)



PDAs (invoke_signed)



PDA

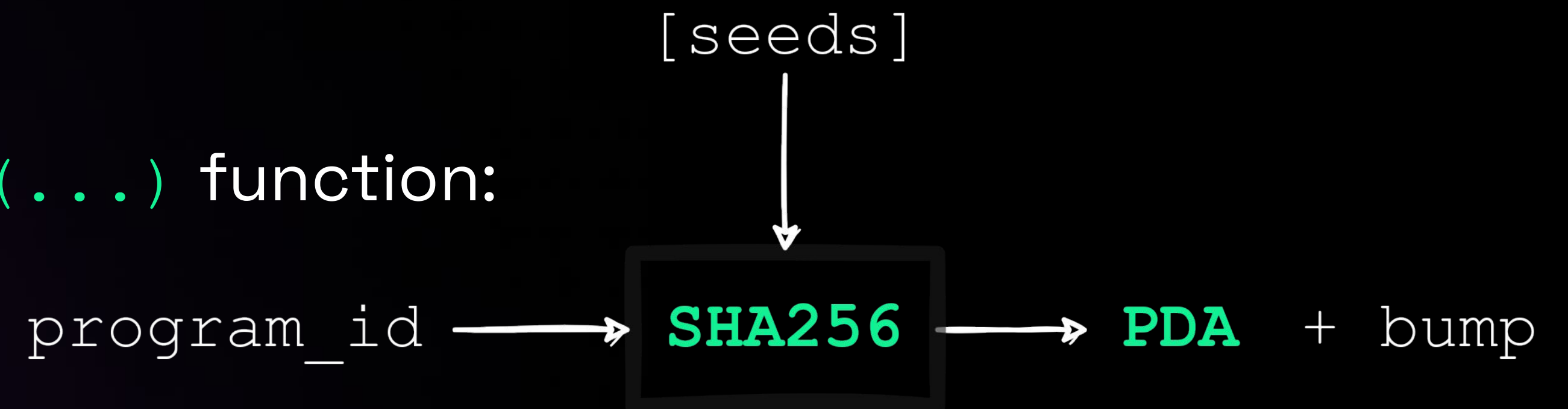
- Programs want to be able to sign accounts for cross-program invocation.

- PDA = program derived address

- PDA account = an account whose address is a PDA.

- How does a PDA get derived?

- `Pubkey::find_program_address(...)` function:



- PDA is bumped off the Ed25519 elliptic curve. Hence, there is no private key.

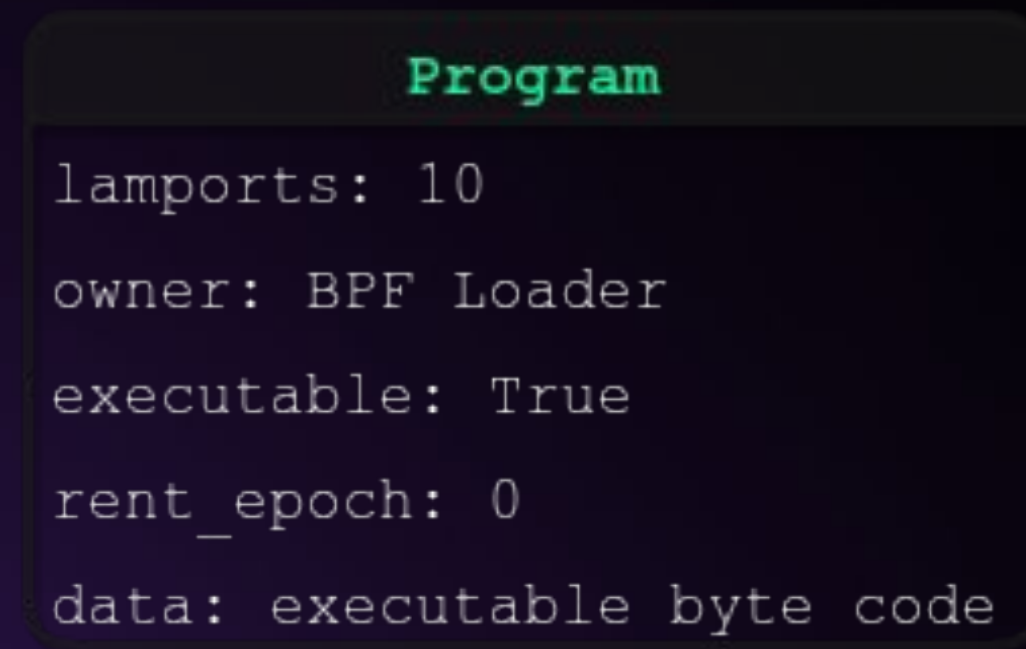
PDAs (deterministic addresses)

```
pub fn push(  
    turnstile_program: Pubkey,  
    state: Pubkey  
) -> Instruction {  
    Instruction {  
        program_id: turnstile_program,  
        accounts: vec![AccountMeta::new(state, false)],  
        data: TurnstileInstruction::Push.try_to_vec().unwrap()  
    }  
}
```

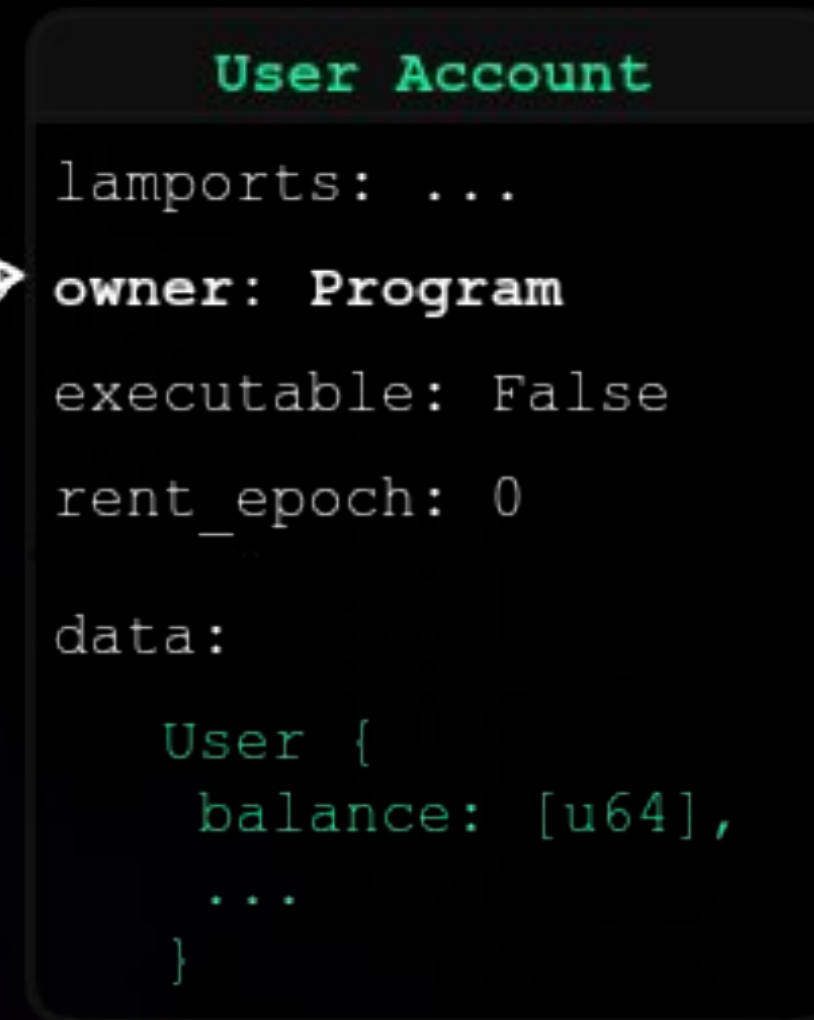
PDAAs (deterministic addresses)

```
...  
  
// function which builds the Push instruction  
pub fn push(  
    turnstile_program: Pubkey,  
) -> Instruction {  
    let (state, bump) =  
        Pubkey::find_program_address(  
            &[b"STATE"],  
            turnstile_program  
        );  
    Instruction {  
        program_id: turnstile_program,  
        accounts: vec![AccountMeta::new(state, false)],  
        data: TurnstileInstruction::Push  
            .try_to_vec()  
            .unwrap(),  
    }  
}
```

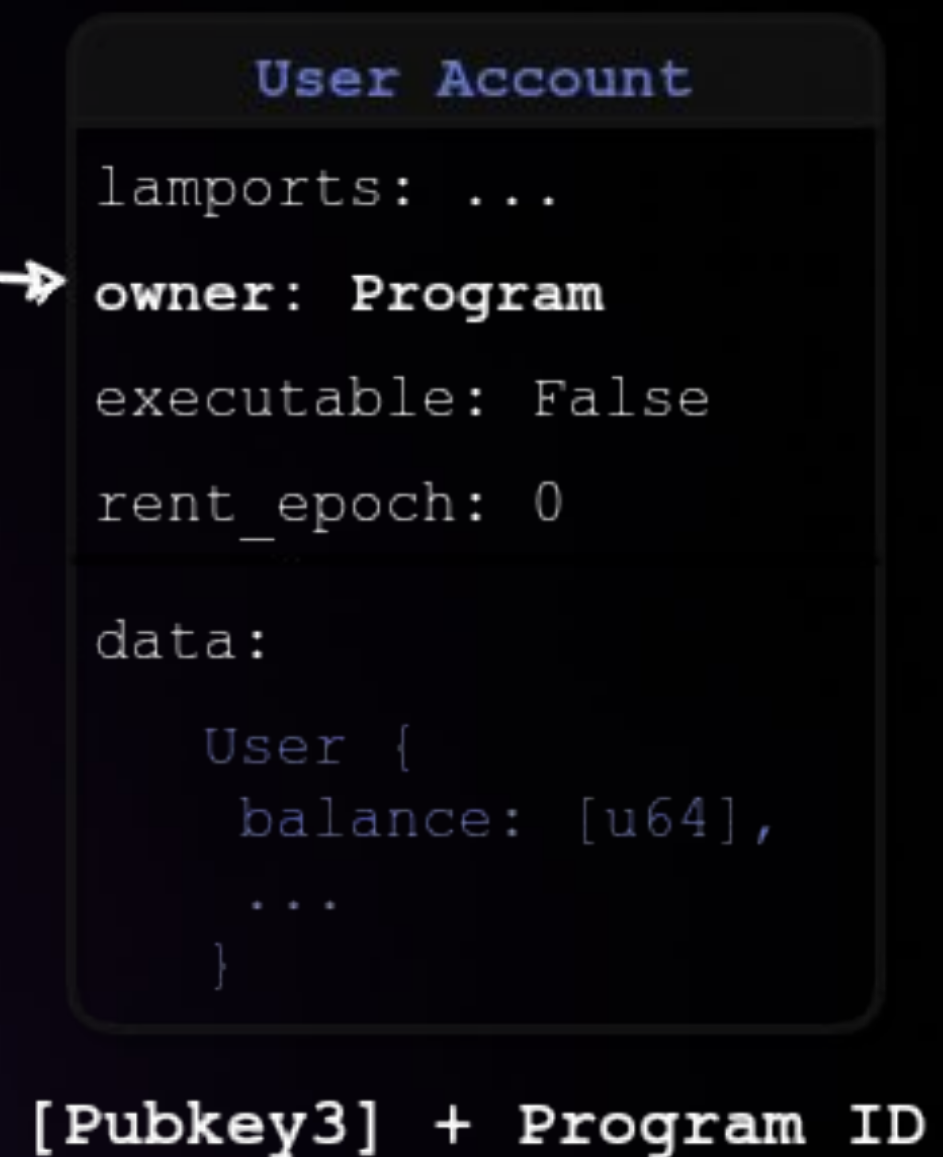
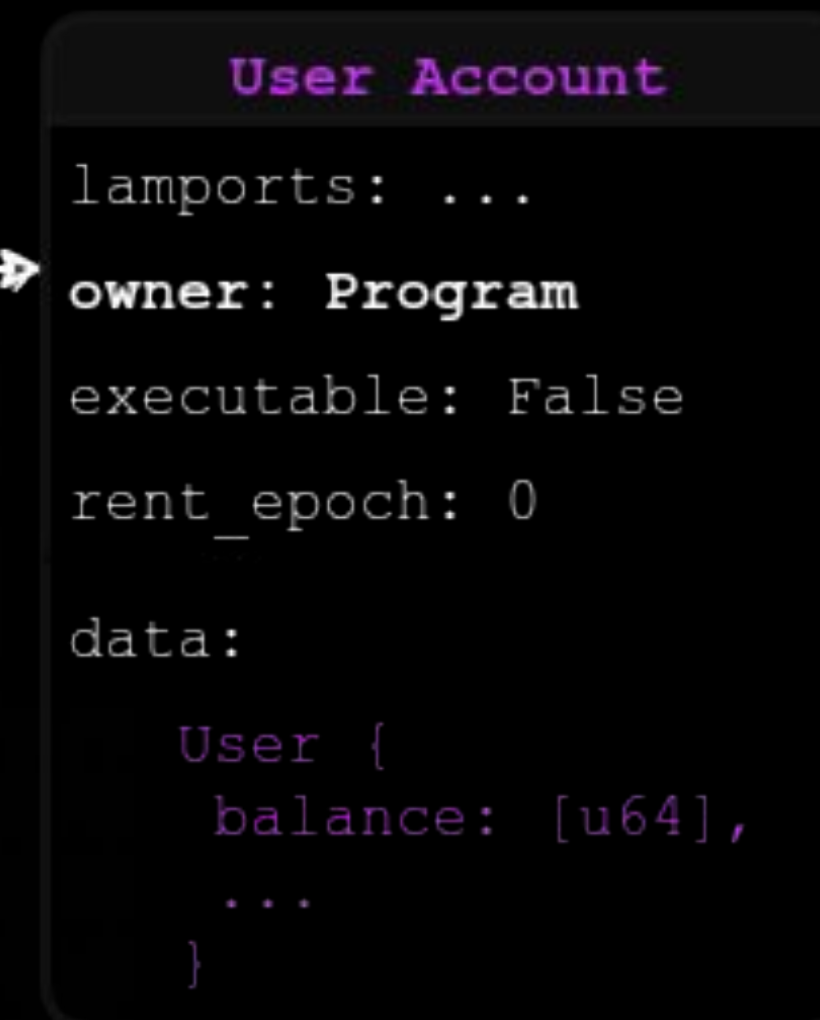
PDA's (hash map)



[Pubkey1] + Program ID

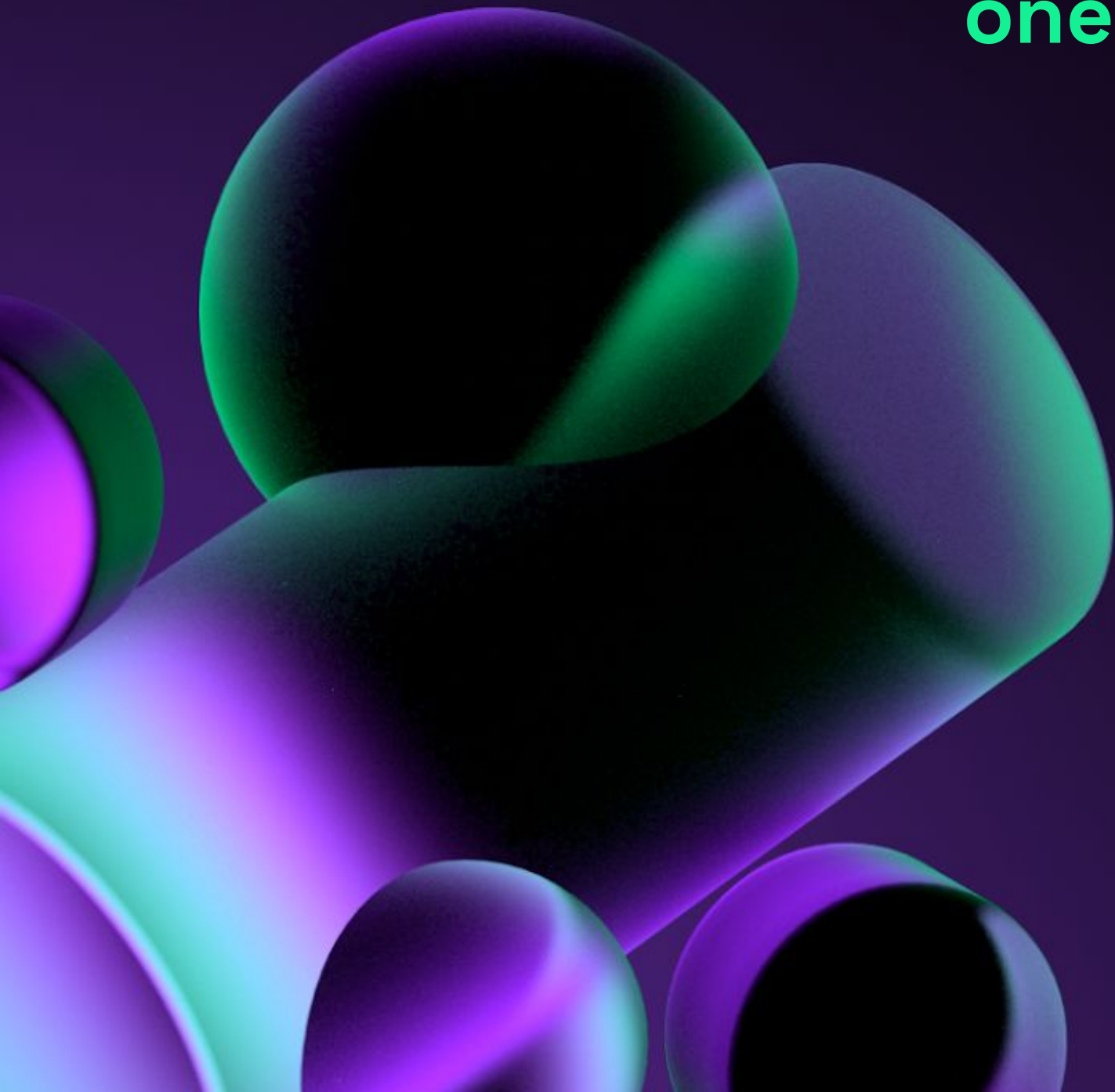


[Pubkey2] + Program ID



Solana Program Library

The advantage of the program/account model is that you can have **one generic program** that operates on **various data**.



Solana Program Library (SPL)

master 36 branches 64 tags

Go to file Add file Code

About

A collection of Solana-maintained on-chain programs

[solana.com](#)

Readme

View license

1.5k stars

72 watching

863 forks

Releases 40

SPL Stake Pool and CLI - v0.6.4 Latest 19 days ago

+ 39 releases

Packages

No packages published

Used by 908

Contributors 94

+ 83 contributors

joncinque	anchor: Updates for token-2022 (#2984)	d51b582 3 days ago	2,452 commits
.github	Run CI on version updates (#2981)	4 days ago	
.travis	Switch to Github Actions for CI (#768)	16 months ago	
associated-token-account/program	Bump solana to v1.9.9 (#2902)	11 days ago	
binary-option	Bump solana to v1.9.9 (#2902)	11 days ago	
binary-oracle-pair	Bump solana to v1.9.9 (#2902)	11 days ago	
ci	fuzz: Fix fuzz build (#2980)	4 days ago	
docs	Update Token TS Docs (#2938)	13 days ago	
examples	Bump solana to v1.9.9 (#2902)	11 days ago	
farms	Bump solana to v1.9.9 (#2902)	11 days ago	
feature-proposal	Bump solana to v1.9.9 (#2902)	11 days ago	
governance	Bump solana to v1.9.9 (#2902)	11 days ago	
libraries/math	Bump solana to v1.9.9 (#2902)	11 days ago	
memo	Bump solana to v1.9.9 (#2902)	11 days ago	
name-service	Bump rust versions (#2978)	4 days ago	
record/program	Bump solana to v1.9.9 (#2902)	11 days ago	
shared-memory	Bump solana to v1.9.9 (#2902)	11 days ago	
stake-pool	Bump solana to v1.9.9 (#2902)	11 days ago	
stateless-asks	Bump solana to v1.9.9 (#2902)	11 days ago	
token-lending	Bump solana to v1.9.9 (#2902)	11 days ago	
token-swap	Bump solana to v1.9.9 (#2902)	11 days ago	
token	token-2022: minor rename pod types with descriptive types (#2983)	3 days ago	
utils	Bump solana to v1.9.9 (#2902)	11 days ago	

<https://github.com/solana-labs/solana-program-library>

Solana Tokens

ERC20 token = a token whose smart contract uses the ERC20 standard



- Fungible tokens use the **ERC20 standard**.
- **Smart contract** template to create a new token.
- To create a new token, you **deploy** ERC20 smart contract on chain.

SPL token = a token created with the Solana token program



- Every SPL token has a standard set of functionality.
- On Solana, there is a **single token program** (spl-token-program).
- To create a new token, you just call a **create_mint** instruction of the token program.

Solana Tokens

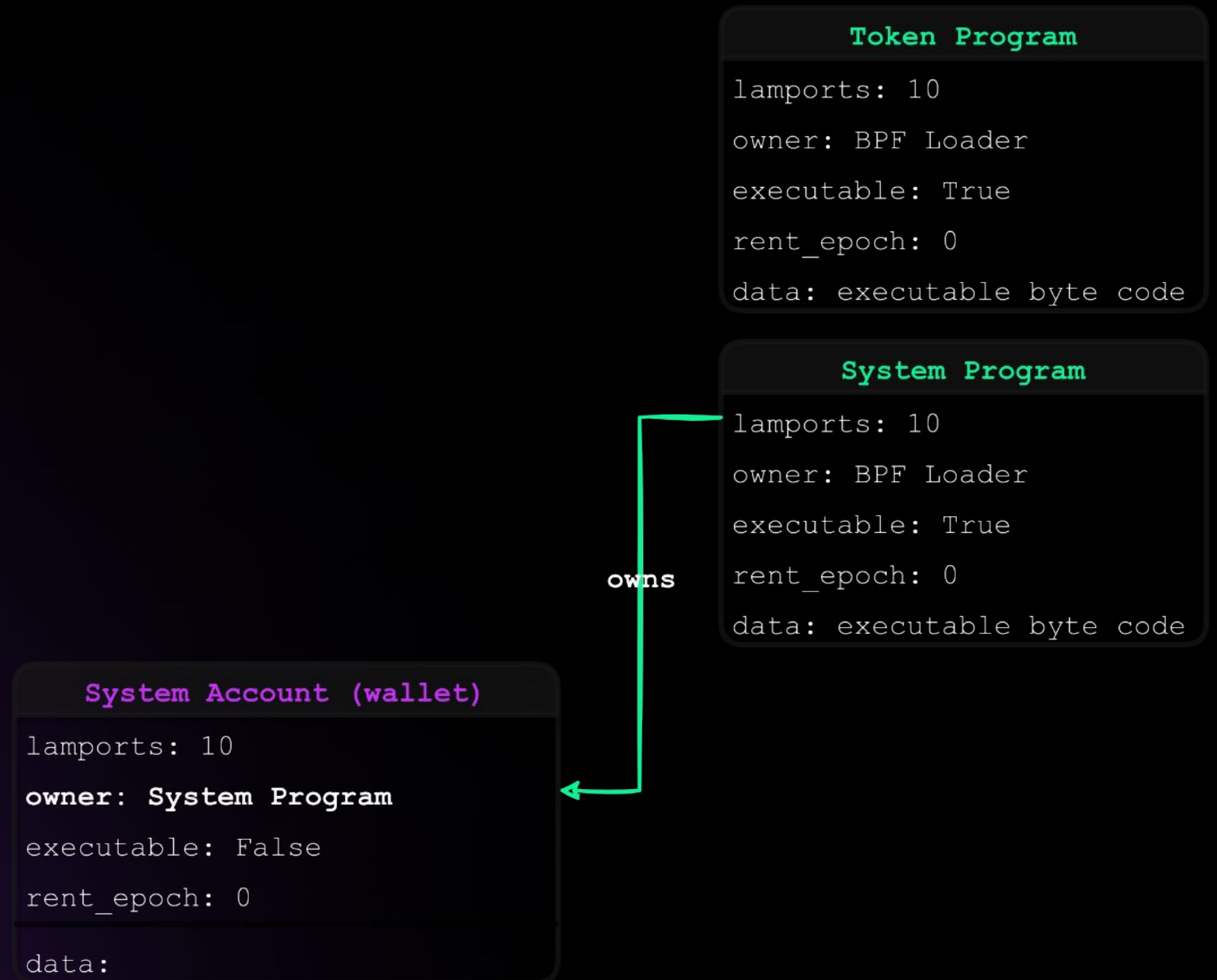
Token Program

```
lamports: 10  
owner: BPF Loader  
executable: True  
rent_epoch: 0  
data: executable byte code
```

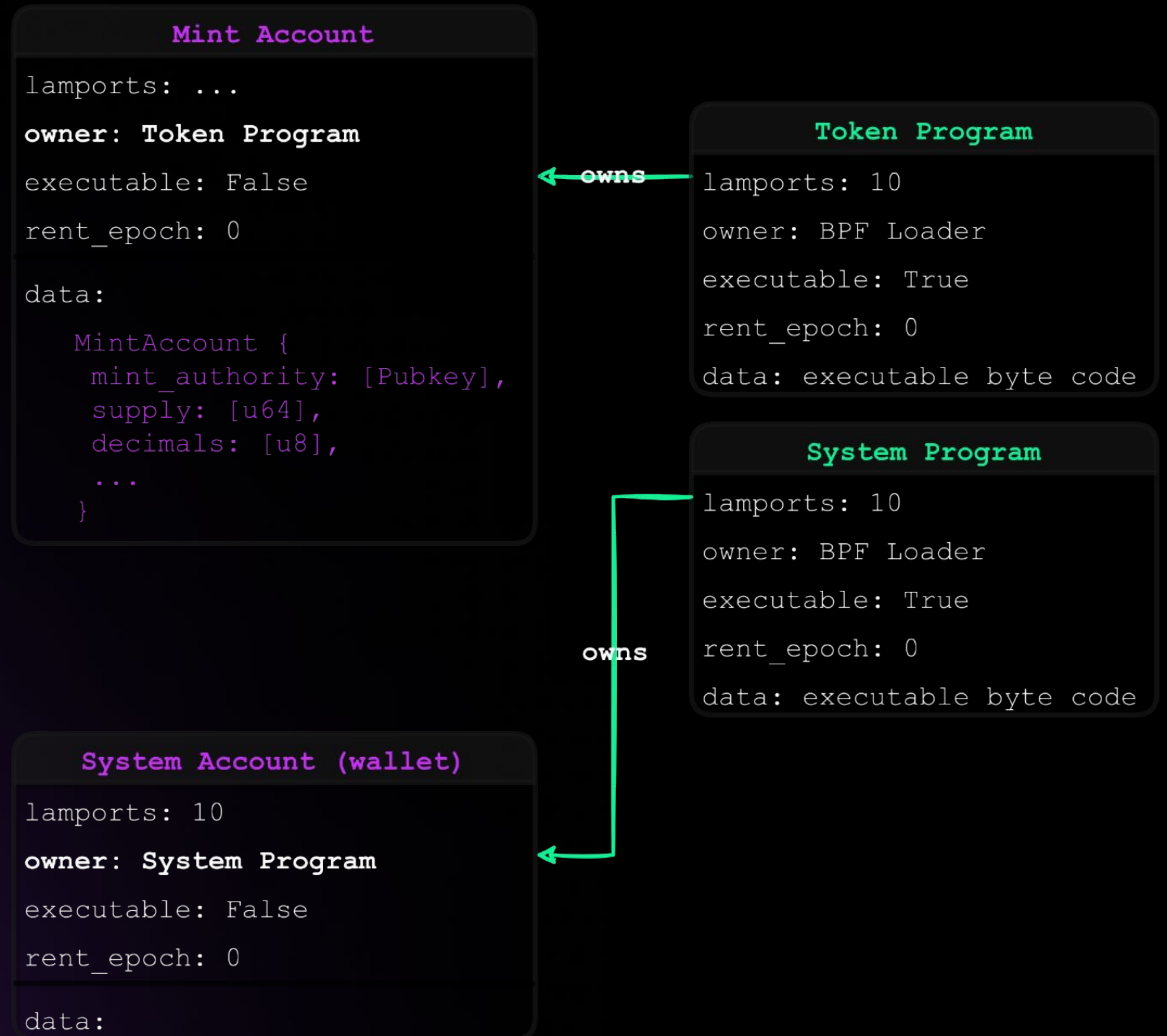
System Program

```
lamports: 10  
owner: BPF Loader  
executable: True  
rent_epoch: 0  
data: executable byte code
```

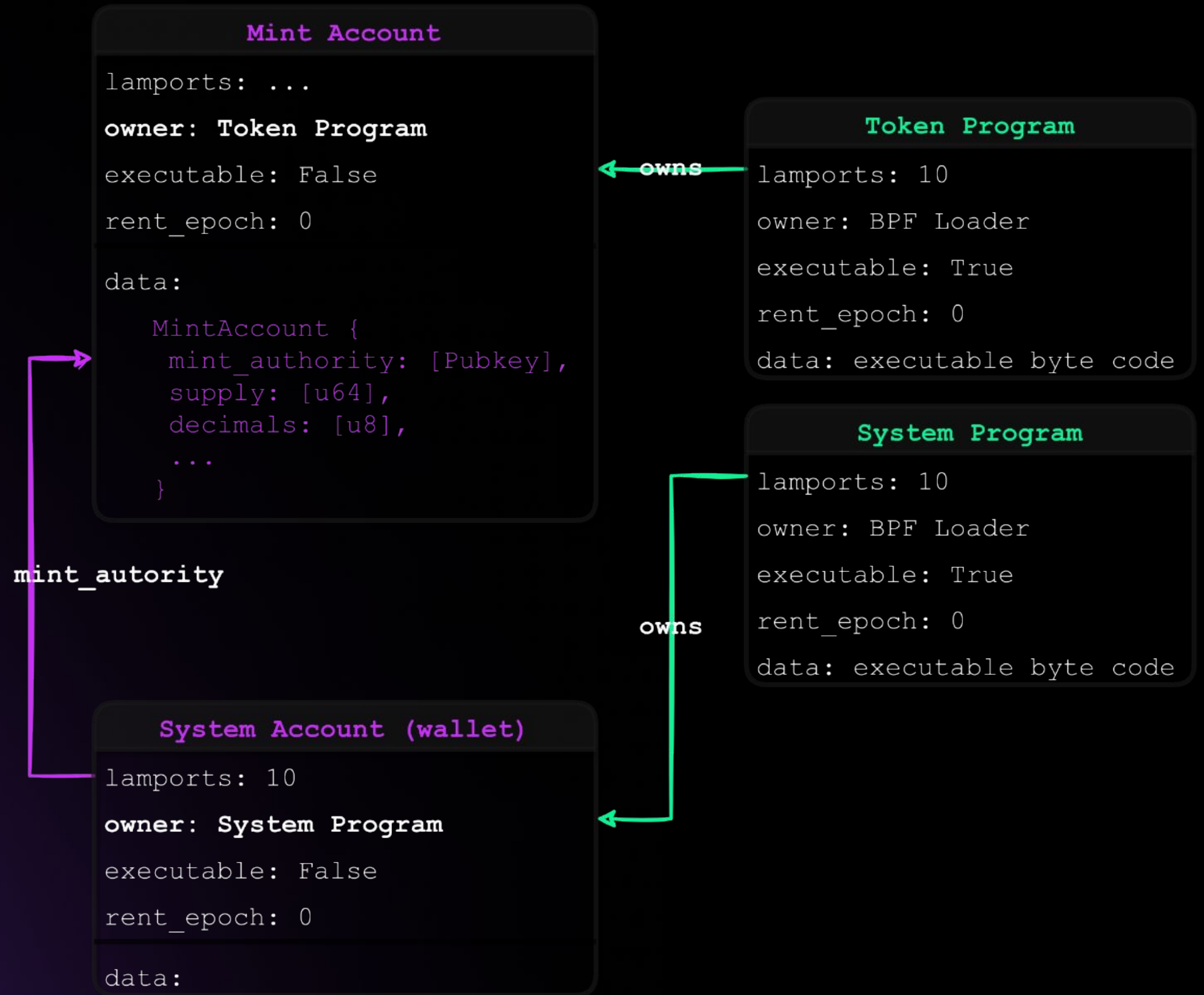
Solana Tokens



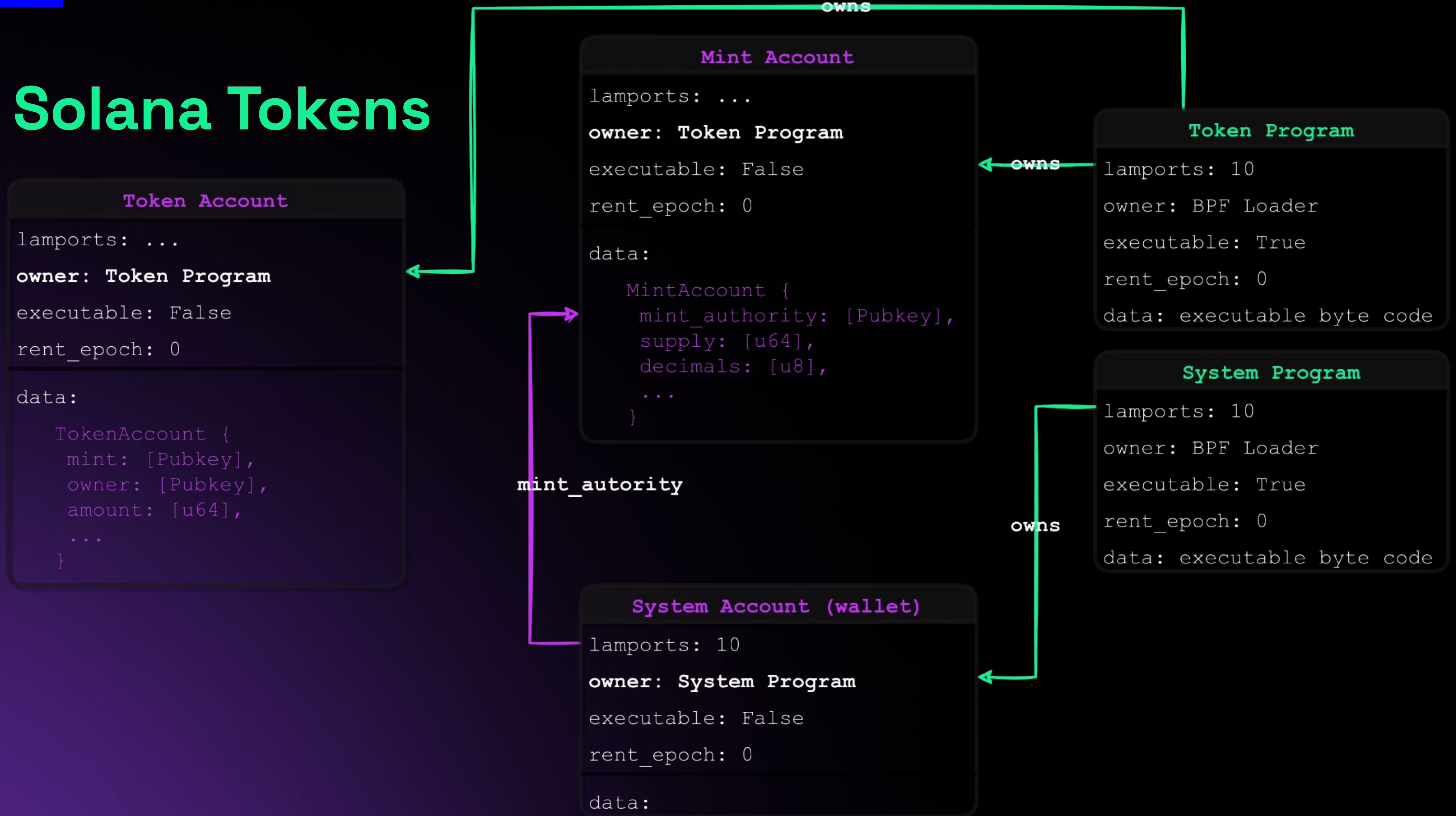
Solana Tokens



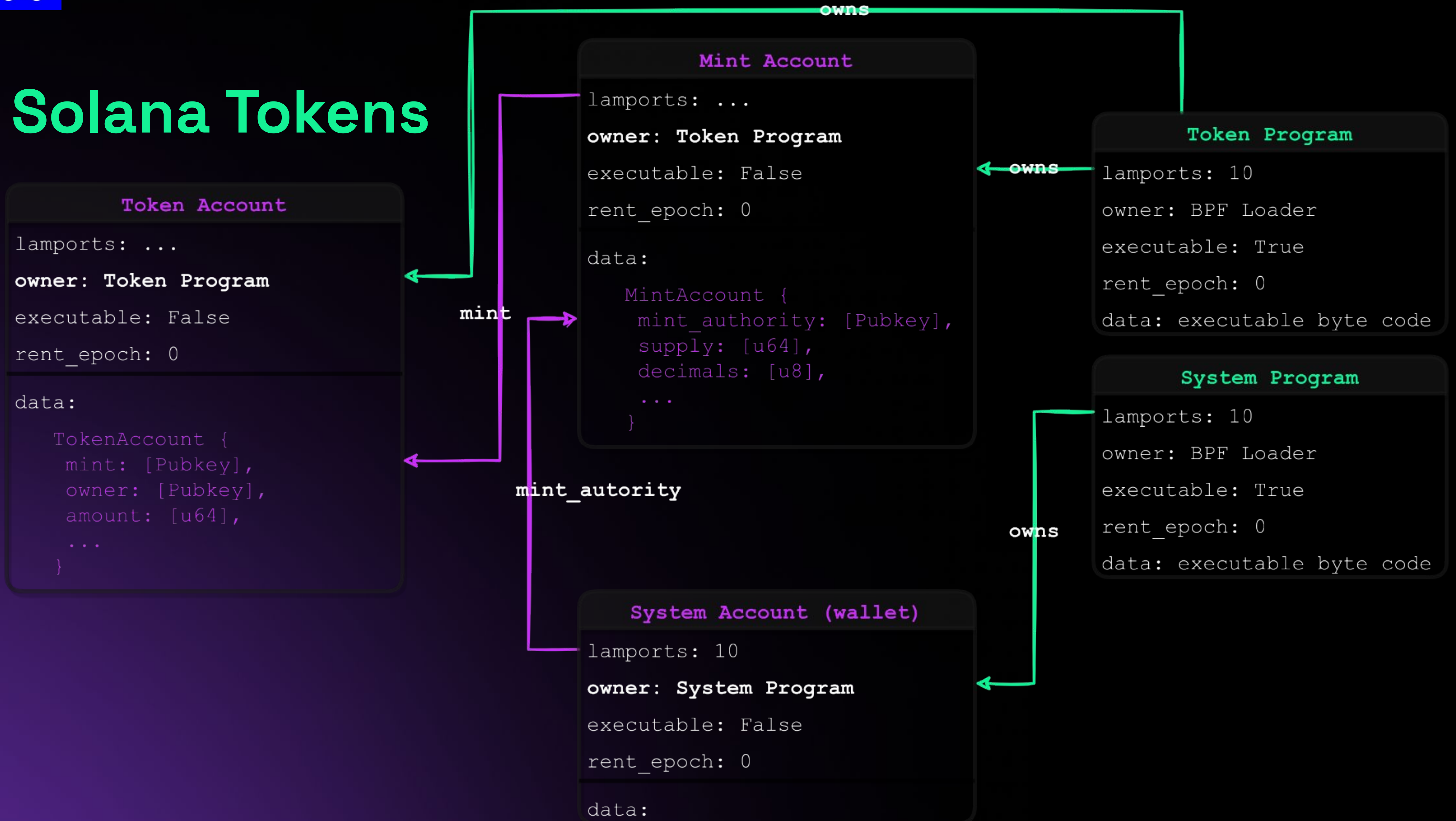
Solana Tokens



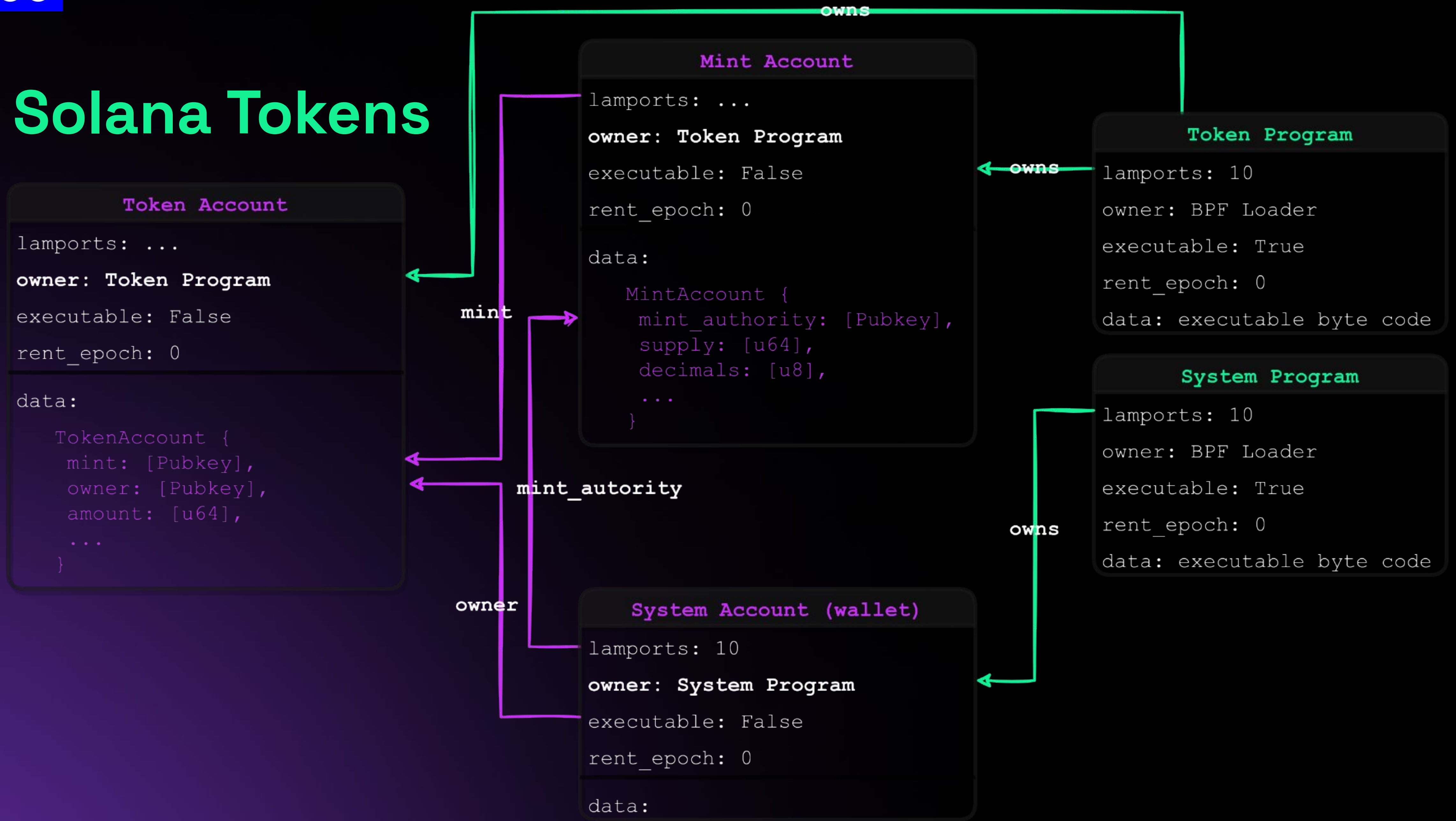
Solana Tokens



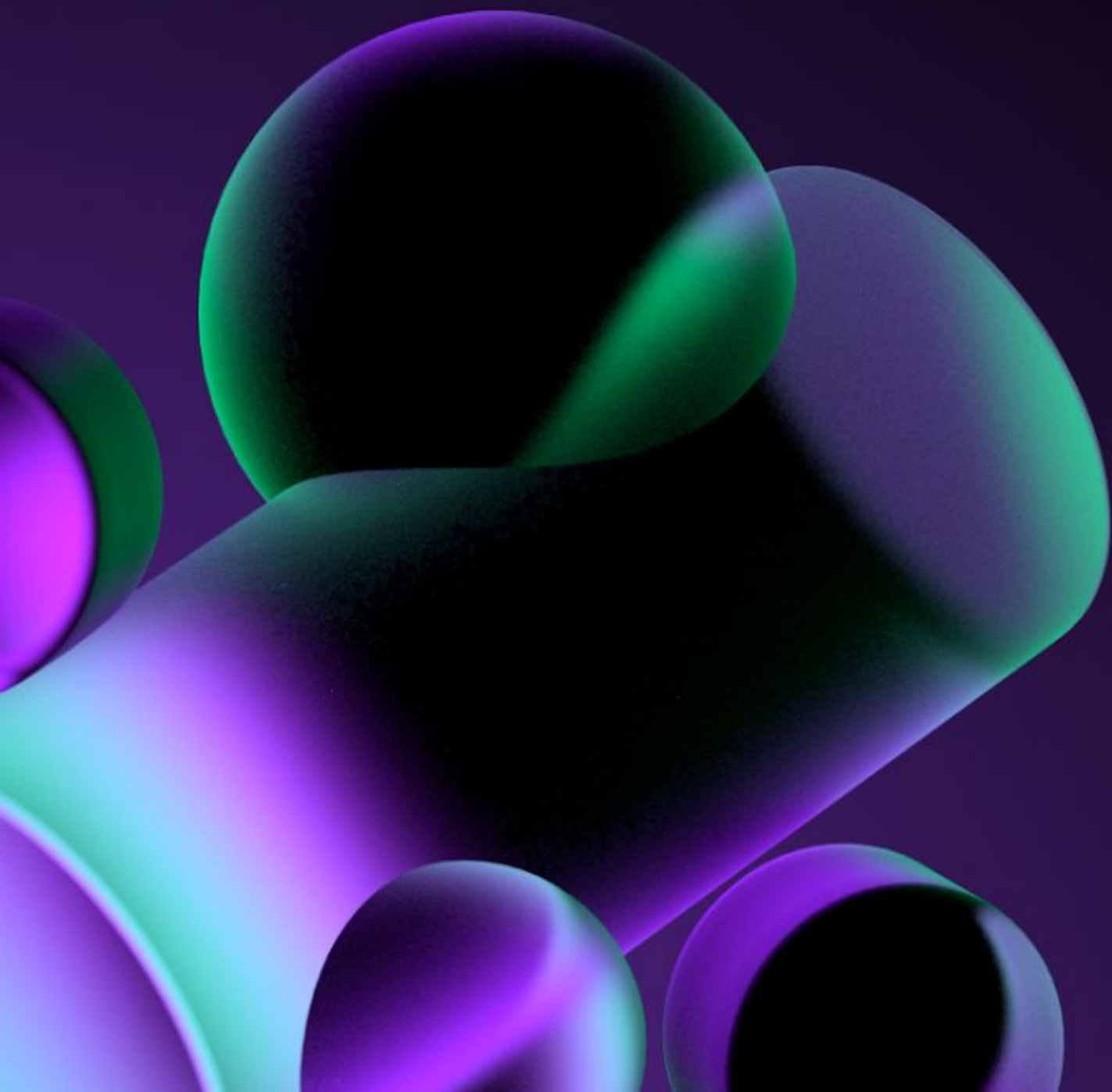
Solana Tokens



Solana Tokens



Hands on example





Thank you

See you next time!