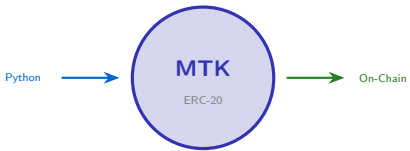
Hands-On: Deploy a Token from Google Colab



Prof. Dr. Joerg Osterrieder

University Lecture Series

colab.research.google.com/github/Digital-AI-Finance/Cryptocurrency/blob/main/notebooks/deploy_token.ipynb

Before You Start — Setup Checklist

✓ **Install MetaMask** browser extension

metamask.io — Chrome or Firefox

✓ **Create a NEW throwaway wallet**

Never use a wallet with real funds!

✓ **Get free Sepolia ETH from faucet**

cloud.google.com/application/web3/faucet/ethereum/sepolia

✓ **Open the Colab notebook**

colab.research.google.com/.../deploy_token.ipynb

SECURITY

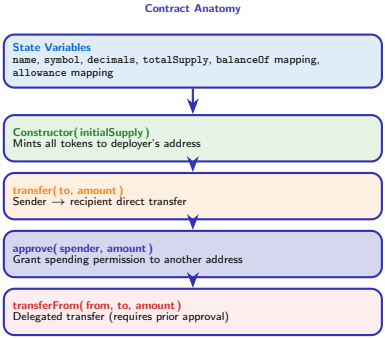
Your **private key** signs transactions on your behalf. It is the master password to your wallet.

Never share it.
Never use a mainnet wallet.
Never paste it in plain text.

The notebook uses getpass — your key is read securely and **never displayed** in the output.

All tools are free. Sepolia ETH is test money with no real value — perfect for learning.

Understanding the Contract Code



Why self-contained?

No npm, no OpenZeppelin imports. The contract is defined inline so `py-solc-x` can compile it directly in Colab without any external dependencies.

Why 18 decimals?

Matches ETH precision. Solidity has no floating point, so 1 token $= 10^{18}$ base units. When you deploy with `initialSupply = 500,000`, the contract stores $500{,}000 \times 10^{18}$ internally.

Why a constructor argument?

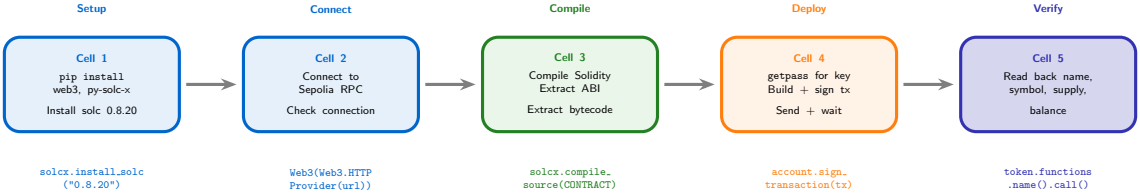
You choose the total supply at deploy time. The constructor runs exactly once when the contract is created, minting all tokens to your address (the deployer).

ERC-20 standard

These five elements (`balanceOf`, `transfer`, `approve`, `allowance`, `transferFrom`) are the minimal ERC-20 interface. Any wallet or DEX that speaks ERC-20 can interact with your token.

The contract is 40 lines of Solidity — fully self-contained, no imports needed, compiles in Python.

Live Coding Walkthrough — What Each Cell Does



Cells 1–2 (Setup): Install Python packages and the Solidity compiler. Connect to Sepolia via a public RPC endpoint. No API key required.

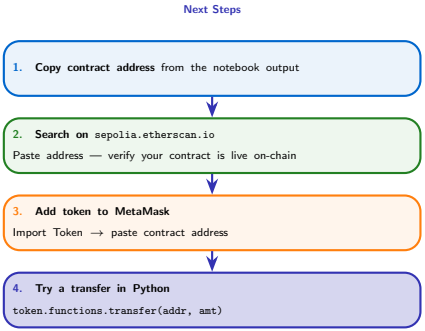
Cell 3 (Compile): Feed the Solidity source to the compiler. Extract ABI (interface) and bytecode (deploy payload).

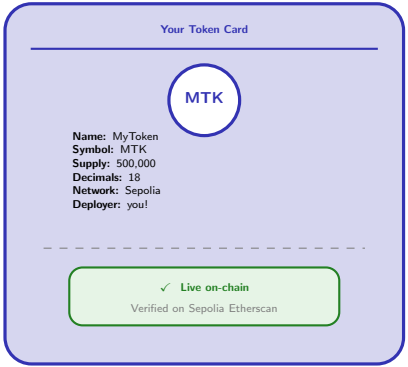
Cell 4 (Deploy): Securely read your private key, build the deploy transaction, sign it locally, broadcast to Sepolia.

Cell 5 (Verify): Call read-only functions to confirm your token is live.

Five cells, five minutes. Each cell builds on the previous one — run them top to bottom in order.

After You Deploy — What to Do Next





Colab notebook: colab.research.google.com/github/Digital-AI-Finance/Cryptocurrency/blob/main/notebooks/deploy_token.ipynb