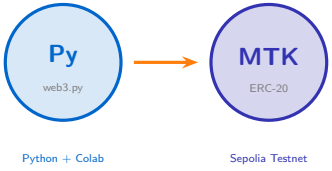
# Deploy a Token from Python

## Colab Notebook Walkthrough

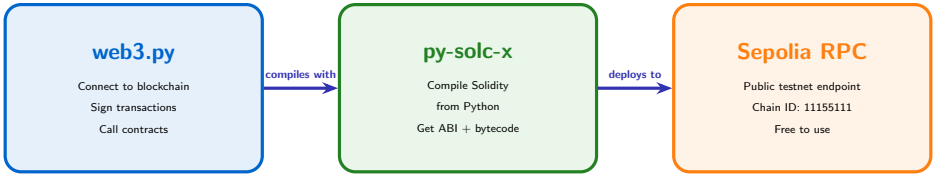



Prof. Dr. Joerg Osterrieder

University Lecture Series

**No Remix, no Hardhat — just Python in a browser.**

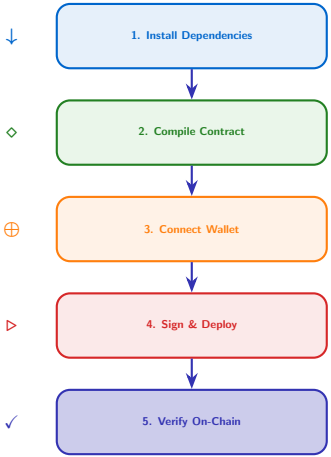
# The Python Toolchain




```
pip install web3 py-solc-x
```

**Two pip packages and a public RPC URL — that is the entire toolchain.**

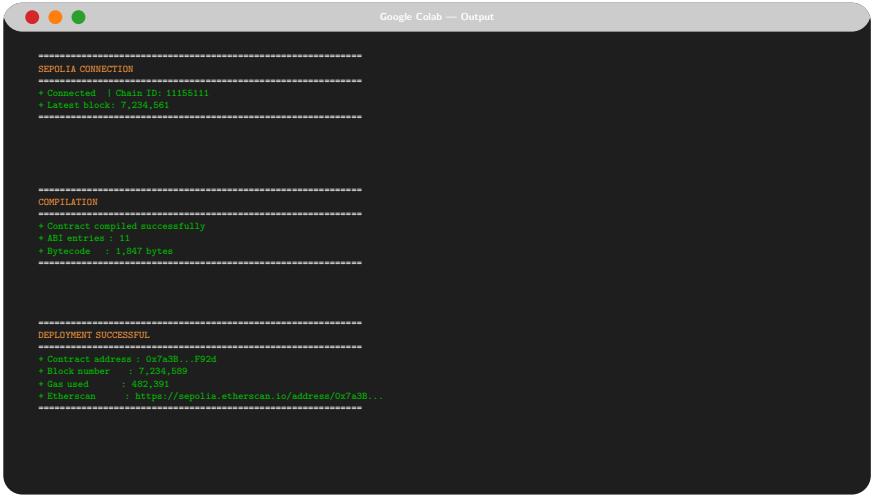
# The Deployment Pipeline




| Step | Python Code | On-Chain Effect |
|---|---|---|
| 1 | `pip install web3 py-solc-x` | None (local setup) |
| 2 | `solcx.compile_source()` | None (local compile) |
| 3 | `Web3(HTTPProvider())` | RPC handshake |
| 4 | `sign_transaction()` | Tx broadcast, contract created |
| 5 | `token.functions.name()` | Read-only call |

**Five cells in Colab, five steps from zero to a live token — the notebook mirrors this pipeline exactly.**

# What the Notebook Outputs




**Three output blocks confirm success — connection, compilation, and deployment with an Etherscan link.**

# Key Takeaways

| # | Takeaway |
|---|---|
| 1 | **Python can deploy smart contracts** — no Remix needed. A Colab notebook with `web3.py` replaces the entire browser-based IDE workflow. |
| 2 | **py-solc-x compiles Solidity locally** — no npm, no Hardhat, no Node.js. One `pip install` gives you a full Solidity compiler in Python. |
| 3 | **web3.py signs and sends transactions** — your private key never leaves your machine. The key is read with `getpass` and used only to sign locally. |
| 4 | **Sepolia testnet is free** — experiment without risk. Test ETH comes from public faucets and the RPC endpoint is open to everyone. |
| 5 | **The same code works for mainnet** — just change the RPC URL and chain ID. Everything else (compile, sign, deploy) stays identical. |

**Five cells, two pip packages, zero cost — Python is the fastest path from idea to deployed token.**