Smart Contract Programming

- Solidity (javascript based), most popular
  - Not yet as functional as other, more mature, programming languages
- Serpent (python based)
- LLL (lisp based)

Smart Contract Programming

Solidity

Solidity is a language similar to JavaScript which allows you to develop contracts and compile to EVM bytecode. It is currently the flagship language of Ethereum and the most popular.

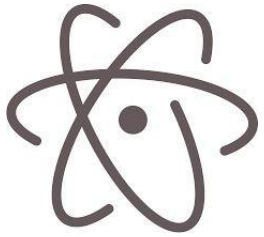
- Solidity Documentation - Solidity is the flagship Ethereum high level language that is used to write contracts.
- Solidity online realtime compiler

Serpent

Serpent is a language similar to Python which can be used to develop contracts and compile to EVM bytecode. It is intended to be maximally clean and simple, combining many of the efficiency benefits of a low-level language with ease-of-use in programming style, and at the same time adding special domain-specific features for contract programming. Serpent is compiled using LLL.

- Serpent on the ethereum wiki
- Serpent EVM compiler

Smart Contract Programming



Atom Ethereum interface - Plugin for the Atom editor that features syntax highlighting, compilation and a runtime environment (requires backend node).

Atom Solidity Linter - Plugin for the Atom editor that provides Solidity linting.

Vim Solidity - Plugin for the Vim editor providing syntax highlighting.

Vim Syntastic - Plugin for the Vim editor providing compile checking.

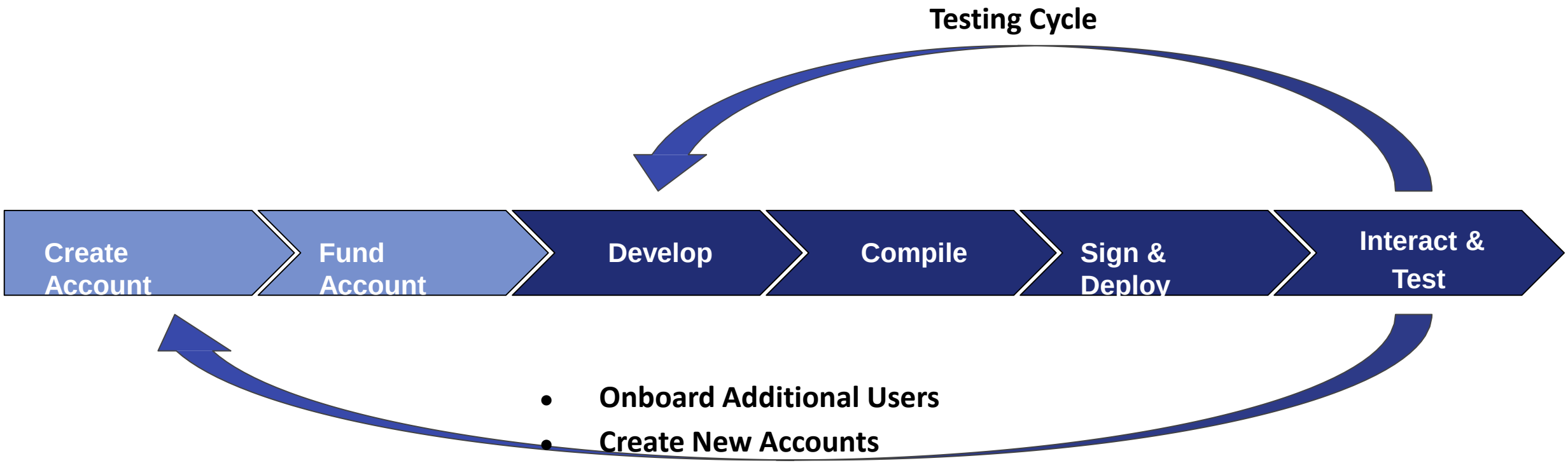
Smart Contract Programming: Solidity

```
contract Example {

   uint value;

   function setValue(uint pValue)
      {  value = pValue;
   }

   function getValue() returns (uint)
      {  return value;
   }

}
```

Smart Contract Programming: Solidity

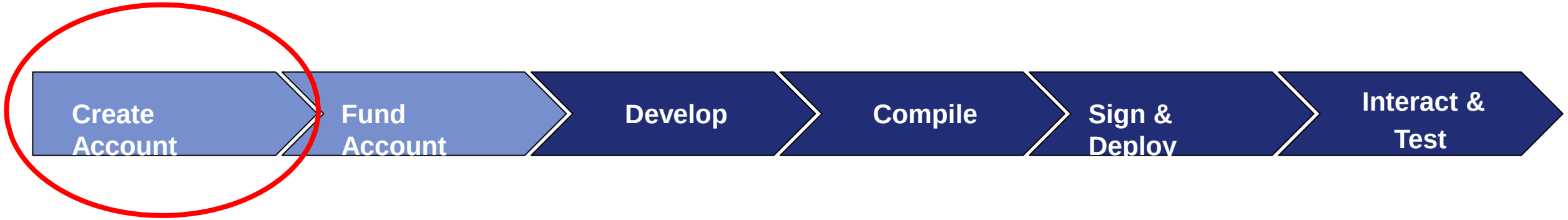
```
var logIncrement =
    OtherExample.LogIncrement({sender: userAddress,
uint value});

logIncrement.watch(function(err, result) {
  // do something with result
})
```

Development Workflow



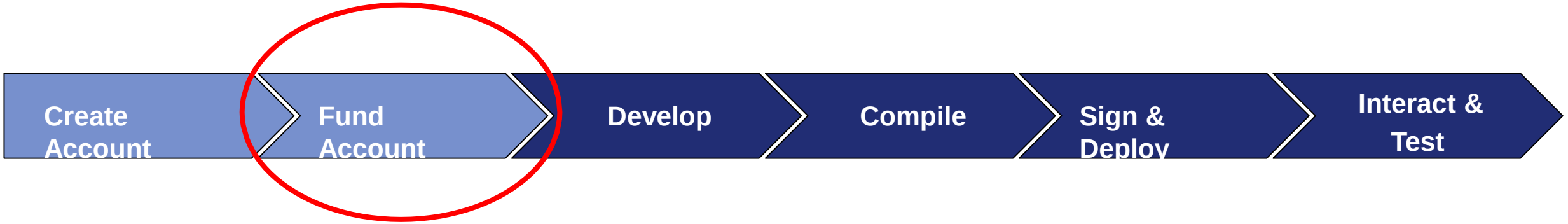
- **Onboard Additional Users**
- **Create New Accounts**
- **Develop New Applications**

Development Workflow: Create Account



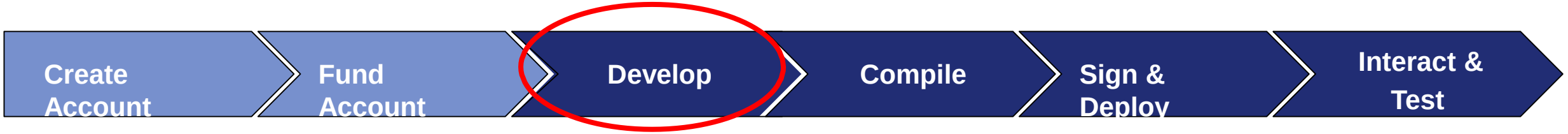
- Programmatically: Go, Python, C++, JavaScript, Haskell
- Tools
  - MyEtherWallet.com
  - MetaMask
  - TestRPC
  - Many other websites

Development Workflow: Fund Account



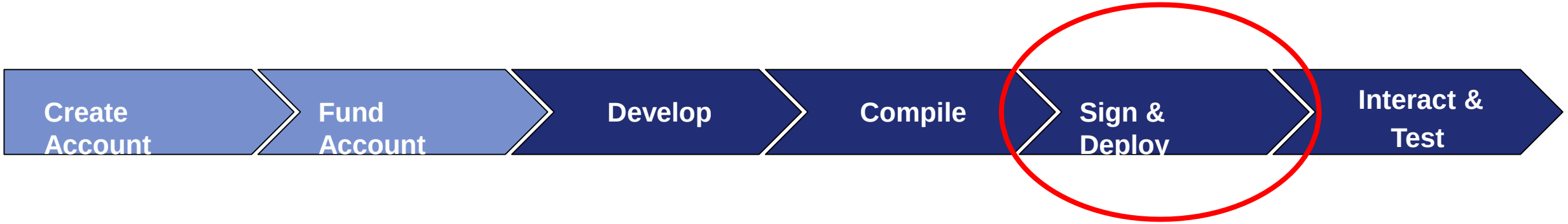
- From friends
- Faucet
- Exchanges (for public blockchain)

Development Workflow: Develop



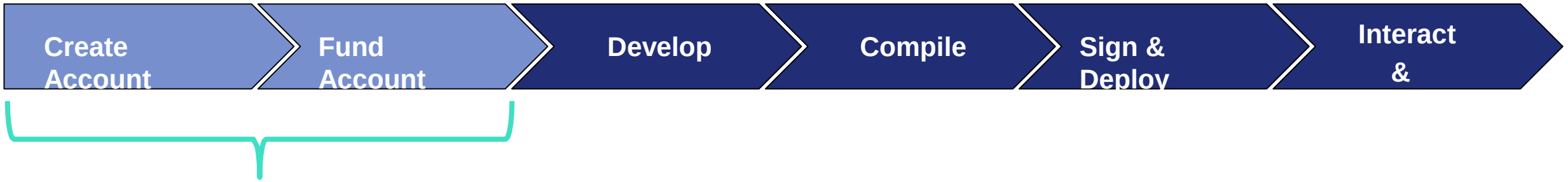
- **Ethereum Application Components:**
  - **Base application**: can be developed in **any** language
  - **Smart contract**: developed in Solidity or one of the other contract compatible languages
  - **Connector library**: facilitates communication between base application and smart contracts (Metamask)

Development Workflow: Sign and Deploy



*Library that facilitates communication and connection with Blockchain; Connects your code to a running node.

Development Workflow: TestRPC



TestRPC/TestChain

- Local development or Test Blockchain
- https://github.com/ethereumjs/testrpc

Development Workflow: TestRPC

- EthereumJS TestRPC: https://github.com/ethereumjs/testrpc is suited for development and testing
- It's a complete blockchain-in-memory that runs only on your development machine
- It processes transactions instantly instead of waiting for the default block time – so you can test that your code works quickly – and it tells you immediately when your smart contracts run into errors
- It also makes a great client for automated testing
- Truffle knows how to use its special features to speed up test runtime by almost 90%.