ETHEREUM & SMART CONTRACTS:

Enabling a Decentralized Future.

JANICE NG
SANEEL SREENI

Introductions

Meet Your Lecturers

Janice Ng

Head of Education

Saneel Sreeni

Software Developer

Table of Contents

01 Smart Contracts

02 Ethereum

03 EVM

04 Use Cases

Smart Contracts

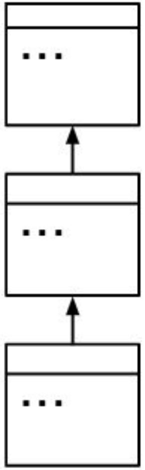
Bitcoin Review

Before we start..a question:

What makes Bitcoin so special?

A Distributed Network: Bitcoin’s Bare Bones

cryptographic identities



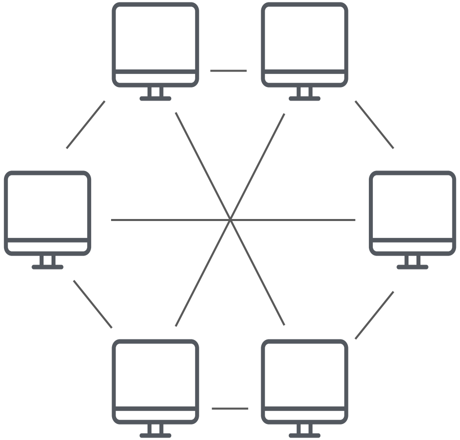
blockchain

trustless consensus

A Distributed Network

Transferrable Benefits of Bitcoin

- Pseudonymous, cryptographic identities allow for accountability
- Democratic decisions made through consensus protocol that **doesn't require trust**
- Immutable ledger of truth
- Uncensorable, cannot be controlled by any one party
- Distributed: no central point of failure



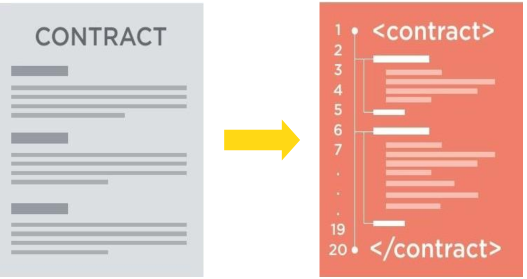
Traditional Contracts

con·tract

(noun) /ˈkäntrakt/

1. a written or spoken agreement … that is intended to be enforceable by law.

Smart Contracts



smart con·tract

(noun) /smärt ˈkäntrakt/

1. code that **facilitates**, **verifies**, or **enforces** the negotiation or execution of a digital contract.
   a. **Trusted entity** must run this code

Ethereum

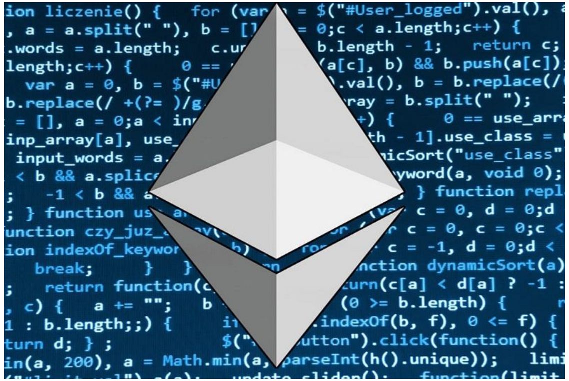
What is Ethereum?

Ethereum is a **decentralized** platform designed to run **smart contracts**

- Distributed computer to execute code
- Account-based blockchain
- Transactions == state transaction function

Ethereum has a native asset called **ether**

- Basis of value in the Ethereum ecosystem

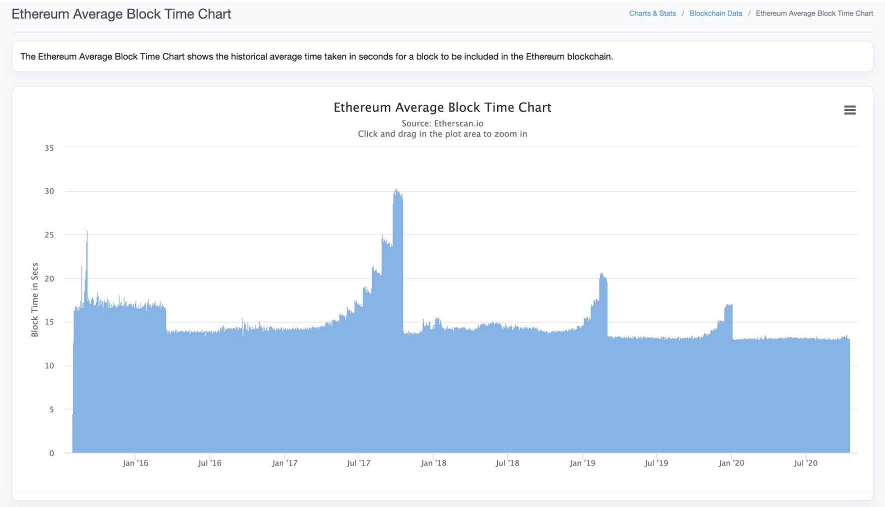


Ethereum

What is Ethereum?

Misc. Implementation Details

- Block creation time: ~13 sec (Ethereum) vs ~10 min (Bitcoin)
- Exchange Rate: $367.75 (2020-10-16)



Ethereum Average Block Time Chart

Charts & Stats / Blockchain Data / Ethereum Average Block Time Chart

The Ethereum Average Block Time Chart shows the historical average time taken in seconds for a block to be included in the Ethereum blockchain.

Ethereum Average Block Time Chart

Source: Etherscan.io

Click and drag in the plot area to zoom in

Ethereum vs Bitcoin...What is the difference?

WHO WOULD WIN?

Bitcoin

- First successful cryptocurrency
- Trustless
- Immutable
- Uncensorable
- Pseudonymous
- No central point of failure
- One-CPU-One-Vote

Ethereum

?

Ethereum vs Bitcoin...What is the difference?

Bitcoin

- The "Gold Standard" of blockchains
- Asset: Bitcoins
  - Primary purpose of the Bitcoin blockchain
- Simple and robust
- Stack-based, primitive scripting language, not Turing-complete
- UTXO-based

Ethereum

- Smart Contract Blockchain Platform
- Asset: Ether
  1. Fund computation
  2. Align incentives
- Complex and feature-rich
- Turing-complete scripting language
- Account-based

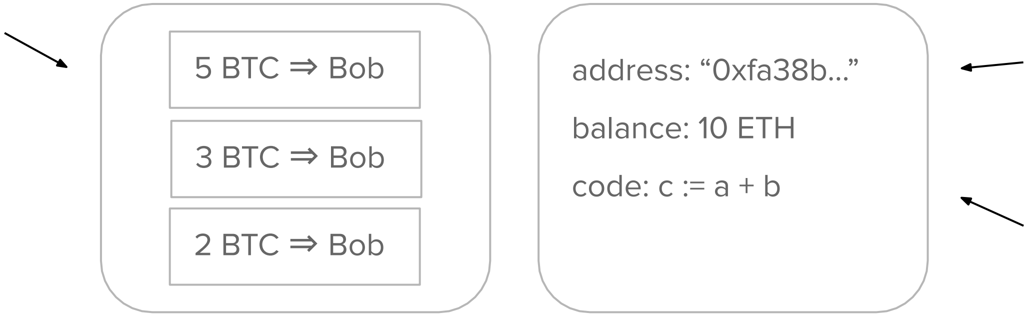
Ethereum Accounts

Bitcoin:

Bob owns private keys to set of UTXOs

Ethereum:

Alice owns private keys to an account



Easy to make transactions and prevent double spending

5 BTC ⇒ Bob

3 BTC ⇒ Bob

2 BTC ⇒ Bob

address: “0xfa38b…”

balance: 10 ETH

code: c := a + b

Space-efficient to update balances instead of storing UTXOs

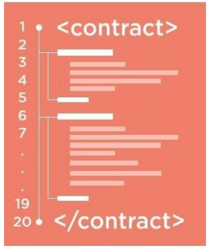
Easier to look up balance and transfer between accounts when programming

AUTHOR: PHILIP HAYES & GLORIA ZHAO

Ethereum Account Types

Externally Owned Accounts

- Owned by some external entity (person, corporation, etc.)
- Can send transactions to transfer ether or trigger contract code
- Contains:
    - Address
    - Ether Balance



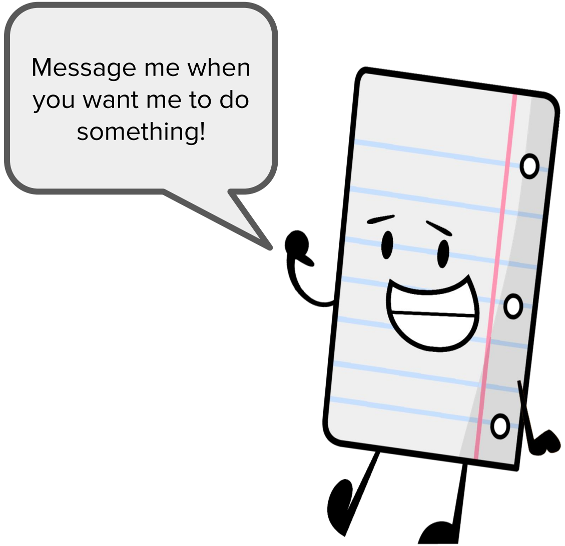
Contract Accounts

- “Owned” by contract
- Code execution triggered by transactions or function calls (msg)
- Contains:
    - Address
    - Associated contract code
    - Persistent storage

Ethereum Smart Contracts: Control

Smart Contracts in Ethereum are like autonomous agents that live inside of the Ethereum network

- React to external world when "poked" by transactions (which call specific functions)
- Have direct control over:
  - **internal ether balance**
  - **internal contract state**



Ethereum Smart Contracts Purposes

Ethereum Contracts generally serve four purposes:

- **Store and maintain data**
  - Data represents something useful to users or other contracts
  - Ex: a token currency or organization’s membership
- **Manage contract or relationship between untrusting users**
  - Ex: financial contracts, insurance
- **Provide functions to other contracts**
  - Serving as a software library
- **Complex Authentication**

Recipe for Mining: Ethereum

A full-fledged Ethereum miner must:

0. **Download** the entire Ethereum blockchain
1. **Verify** incoming transactions and **Run** Smart Contract code invoked by transactions
2. **Create** a block
3. **Find** a valid nonce
4. **Broadcast** your block
5. **Profit**!

Image source:
http://www.coindesk.com/information/how-to-set-up-a-miner/

The Distributed Computer

- Ethereum is a “distributed computer”
- Ethereum’s distributed consensus protocol is **Proof-of-Work**
- Network consensus removes the need for Trusted Third Party
- Secure Peer-to-Peer agreements that live on the blockchain forever

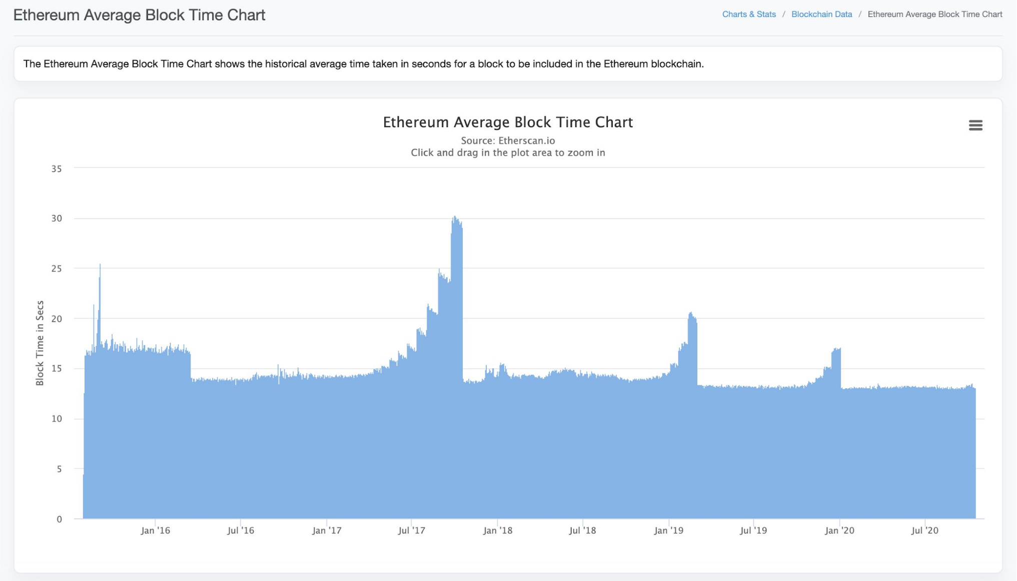
WHAT IS ETHEREUM?

COMPARISON WITH BITCOIN

Misc. Implementation Details

- Block creation time: ~13 sec (Ethereum) vs ~10 min (Bitcoin)
- Proof-of-Work: Ethash (currently ASIC resistant) vs SHA-256
- Exchange Rate: $367.75 (2020-10-16)

THANK YOU



Ethereum Average Block Time Chart

Charts & Stats / Blockchain Data / Ethereum Average Block Time Chart

The Ethereum Average Block Time Chart shows the historical average time taken in seconds for a block to be included in the Ethereum blockchain.

Ethereum Average Block Time Chart

Source: Etherscan.io

Click and drag in the plot area to zoom in

AUTHOR: PHILIP HAYES

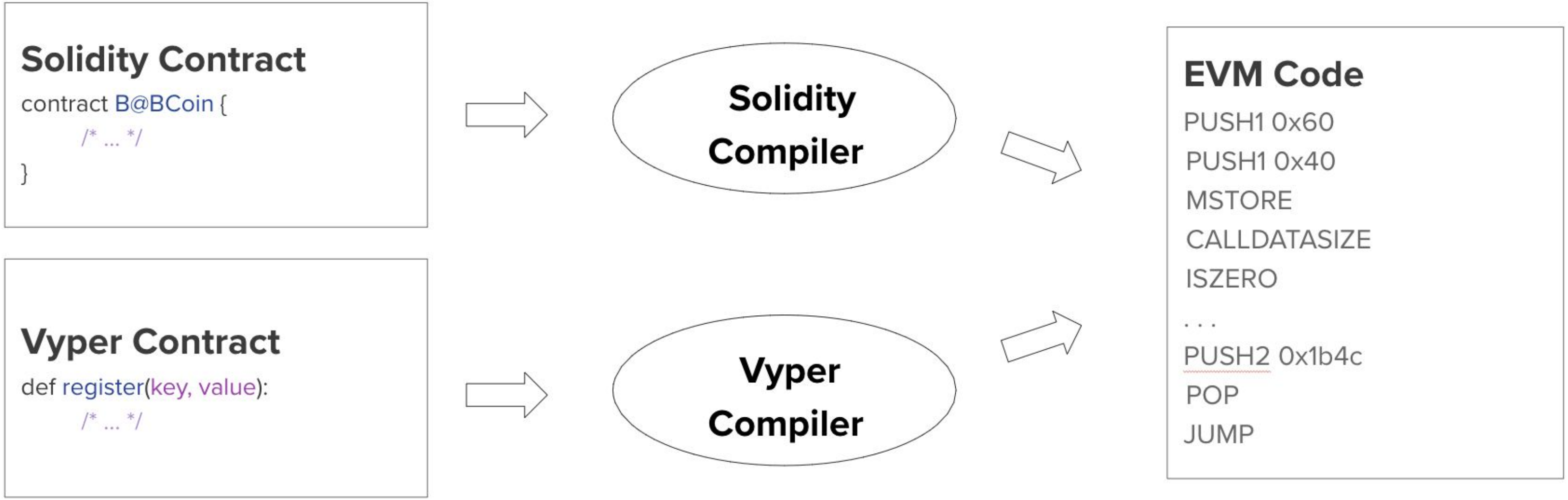
Ethereum Smart Contracts Purposes

```
contract Betting {
    address public owner;
    address public gamblerA, gamblerB, oracle;
    uint[] outcomes;
    struct Bet {
        uint outcome;                       /* Defines a Bet */
        uint amount;
        bool initialized;
    }

    mapping (address => Bet) bets;      /* Keep track of every gambler's bet */
    mapping (address => uint) winnings; /* Keep track of every player's winnings */
    ...
    function makeBet(uint _outcome) payable returns (bool) { … }
    function makeDecision(uint _outcome) oracleOnly() { … }
    function withdraw(uint withdrawAmount) returns (uint remainingBal) { … }
```

Ethereum Virtual Machine

EVM: Compilation & Process



Solidity Contract

```
contract B@BCoin {
        /* ... */
}
```

Vyper Contract

```
def register(key, value):
        /* ... */
```

Solidity Compiler

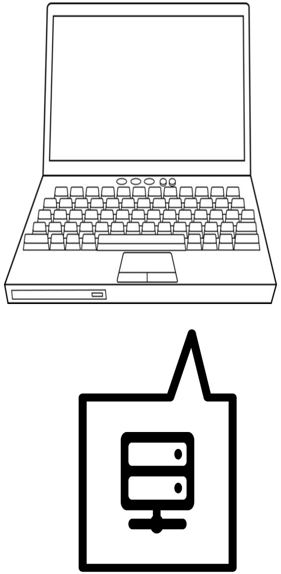
Vyper Compiler

EVM Code

```
PUSH1 0x60
PUSH1 0x40
MSTORE
CALLDATASIZE
ISZERO
. . .
PUSH2 0x1b4c
POP
JUMP
```

EVM: High Level Overview

- The **EVM (Ethereum Virtual Machine)** is a “mini computer” on your computer that runs contract code
- Contract code that actually gets executed on every node is EVM code
  - **EVM code**: low-level, stack based bytecode language (i.e. JVM bytecode)
- Every Ethereum node runs EVM



Question...

What if our contract has an infinite loop?

What if our contract has an infinite loop?

- Every node on the network will get stuck executing the loop forever!
- By the *halting problem,* it is **impossible** to determine ahead of time whether the contract will ever **terminate**
  - Lead to: **Denial of Service (DoS) Attack**

…is there a solution?

EVM: Gas & Fees

Ethereum's solution:

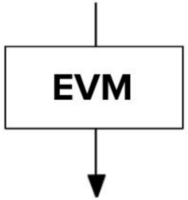
- Every contract requires **"gas"**, which "fuels" contract execution
- Every EVM operation-code requires some gas in order to execute
- Every transaction specifies:
  - **startgas**: Max quantity of gas it is willing to consume
  - **gasprice**: Fee in ether it is willing to pay per unit gas

EVM: Gas & Fees

- At the start of the transaction
  - startgas * gasprice (units = ether) are subtracted from the sender’s account (the one “poking” the contract)
- If the contract **successfully executes…**
  - the remaining gas is refunded to the sender
- If the contract execution **runs out of gas** before it finishes…
  - execution reverts
  - startgas * gasprice are not refunded
- Purchasing gas == purchasing distributed, trustless computational power
- An attacker looking to launch a DoS attack will need to supply enough ether to fund the attack

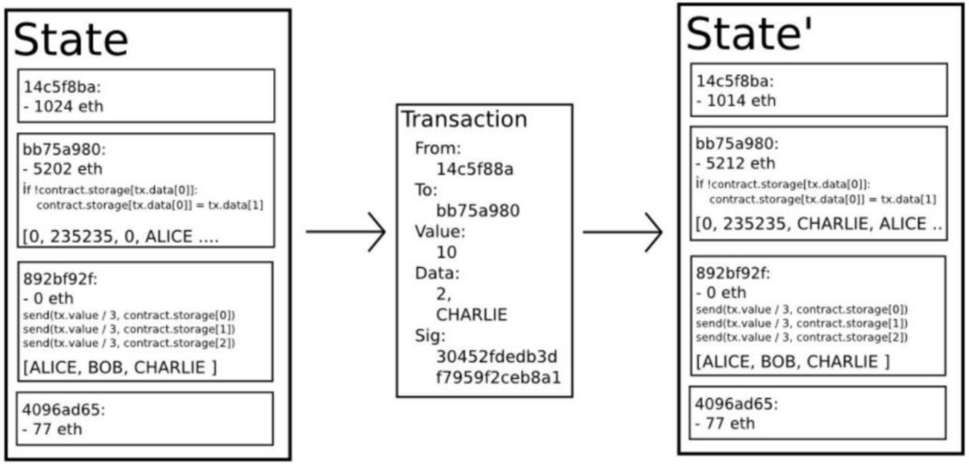
Ethereum Network State: State Transition Function

(block_state, gas, memory, transaction, message, code, stack, pc)



EVM

(block_state’, gas’)



State

14c5f8ba:
- 1024 eth

bb75a980:
- 5202 eth
if !contract.storage[tx.data[0]]:
    contract.storage[tx.data[0]] = tx.data[1]
[0, 235235, 0, ALICE ....

892bf92f:
- 0 eth
send(tx.value / 3, contract.storage[0])
send(tx.value / 3, contract.storage[1])
send(tx.value / 3, contract.storage[2])
[ALICE, BOB, CHARLIE]

4096ad65:
- 77 eth

Transaction
From:
14c5f88a
To:
bb75a980
Value:
10
Data:
2,
CHARLIE
Sig:
30452fdedb3d
f7959f2ceb8a1

State'

14c5f8ba:
- 1014 eth

bb75a980:
- 5212 eth
if !contract.storage[tx.data[0]]:
    contract.storage[tx.data[0]] = tx.data[1]
[0, 235235, CHARLIE, ALICE ..

892bf92f:
- 0 eth
send(tx.value / 3, contract.storage[0])
send(tx.value / 3, contract.storage[1])
send(tx.value / 3, contract.storage[2])
[ALICE, BOB, CHARLIE]

4096ad65:
- 77 eth

Ethereum Whitepaper

Ethereum: Conclusions

- **Ethereum is not about optimising efficiency of computation**
- Its parallel processing is **redundantly parallel**
  - **way to reach consensus** on the system state without needing trusted third parties
- Contract executions are redundantly replicated across nodes
  - ⇒ expensive, slow, memory-intensive
  - creates an **incentive not to use the blockchain** for computation that can be done off chain

Ethereum: Conclusions

Use Blockchain:

- Need for a **shared database** with multiple writers
- Parties **cannot trust** one another, and no trusted third party or authority is available
- Interested in fault-tolerance, data immutability or censorship resistance

Use Centralised Database:

- Database does not need to be shared, or is shared by parties who trust one another
- Must keep data **confidential**
- Must handle **complex** and/or large amounts of data
- Need to be able to edit data
- Interested in cost-effectiveness, speed or **efficiency**

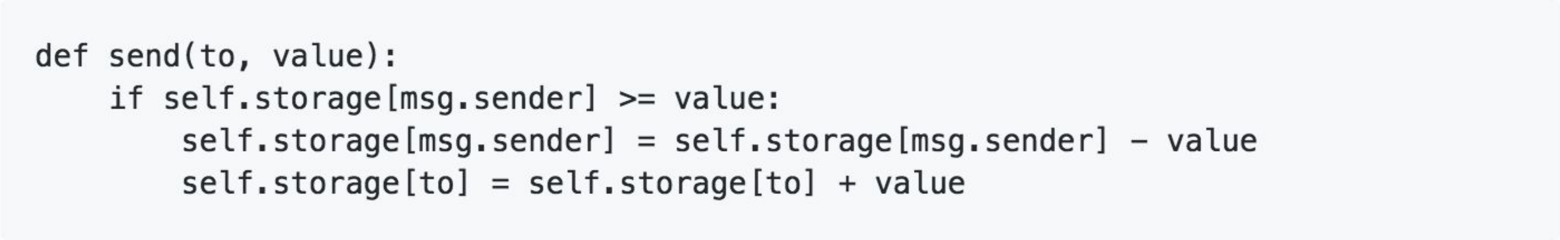
Use Cases

Basic Use Cases

Basic Use Cases

Ethereum Tokens (ERC Tokens)

- Token System Implementation
  - Recreating Bitcoin in 4 lines of code
- Database with one operation
  - Ensure Alice has enough $$ and that she initiated the transaction
  - Subtract X from Alice, give X to Bob

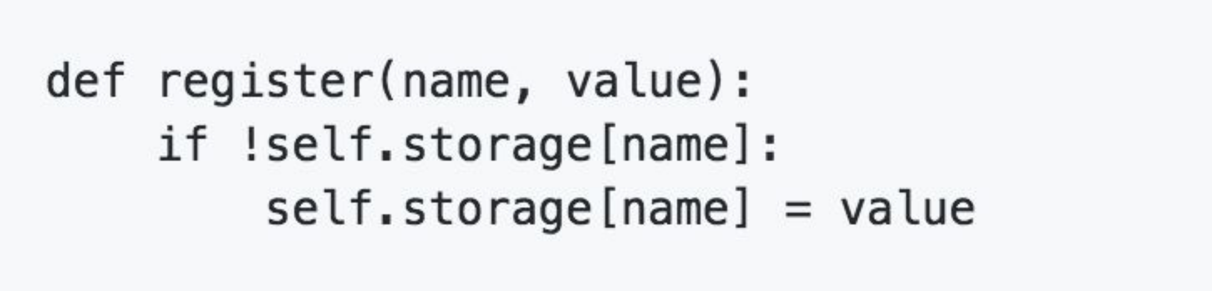


```
def send(to, value):
    if self.storage[msg.sender] >= value:
        self.storage[msg.sender] = self.storage[msg.sender] - value
        self.storage[to] = self.storage[to] + value
```

Basic Use Cases

Domain Naming System (DNS)

- DNS System
  - Maps domain name to IP address
  - “gillian.chu” → “12.34.56.78”
- Easy to implement in Ethereum



```
def register(name, value):
    if !self.storage[name]:
        self.storage[name] = value
```

Advanced Use Cases

Advanced Use Cases

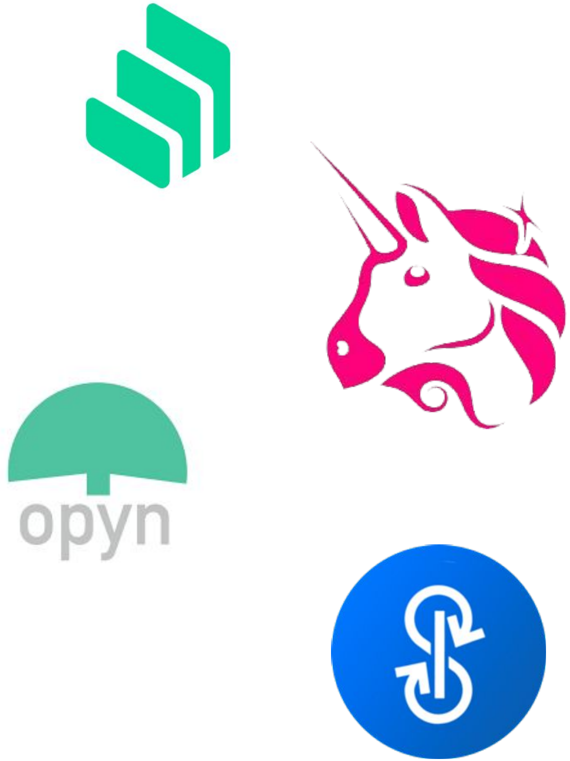
Decentralized Finance (DeFi)

Problem:

- Most of traditional finance is **closed** to regular users
- Most derivatives or other assets built on top of cryptocurrencies are in **centralized** platforms

Pitfalls:

- Financial institutions can **manipulate** markets
- Retail users are **priced out**
- People distrustful of **big banks**



Advanced Use Cases

Decentralized Finance (DeFi)

The Ethereum Solution:

- Build out dApps (decentralized applications) using smart contracts to handle logic
- Use game theory and economics to find ways to cut on trust
- Rely on Ethereum blockchain to verify transactions

Advanced Use Cases

Decentralized Finance (DeFi)

Drawbacks:

- Capital **inefficient**
- **Risk** from composability, smart contracts, etc.
- **Frontrunning** in the mempool and adversarial miner behavior

Can you think of anything else?

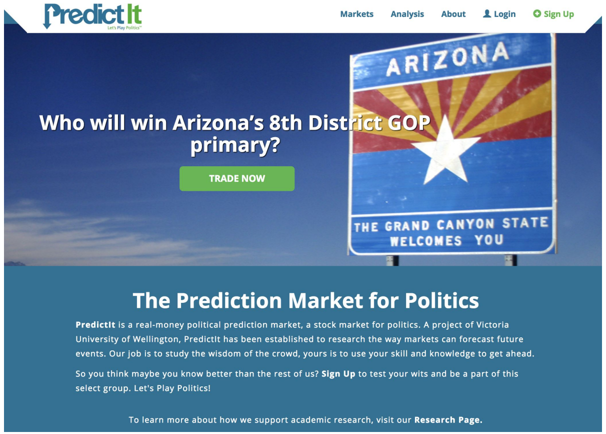
Advanced Use Cases

Prediction Markets

Draws on the wisdom of the crowd

Ex: *Who will win the 2020 Presidential Election? Trump or Biden?*

1. Bets are replaced with shares of outcomes
2. Oracles report the outcome



Who will win Arizona's 8th District GOP primary?

The Prediction Market for Politics

PredictIt is a real-money political prediction market, a stock market for politics. A project of Victoria University of Wellington, PredictIt has been established to research the way markets can forecast future events. Our job is to study the wisdom of the crowd, yours is to use your skill and knowledge to get ahead.

So you think maybe you know better than the rest of us? **Sign Up** to test your wits and be a part of this select group. Let's Play Politics!

To learn more about how we support academic research, visit our **Research Page.**

Advanced Use Cases

Prediction Markets

"A service that never crashes, a service that's completely transparent…"

Benefits:

- No restrictions on market types
- Shared liquidity
- Censorship-resistant
- Automated and trustless

augur

GNOSIS

Advanced Use Cases

Prediction Markets

augur

GNOSIS

Example Use Cases and Markets

- “Buying” information → *Will Movie X flop?*
- Hedging and Insurance → *Will my house burn down?*
- Security Bug Bounty → *Is there a bug in my smart contract?*

Advanced Use Cases

Prediction Markets

Also…

“I bet $1 million that Bob will be alive on October 4.” → **Assassination Market**