# ChainScan Documentation

*Release 0.1*

fungibit

Sep 04, 2017

# Contents:

*Feel the blockchain, one transaction at a time.*

# CHAPTER 1

# What is ChainScan?

ChainScan is a python library implementing simple tools for **iterating over the Bitcoin blockchain**, block by block, transaction by transaction.

# CHAPTER 2

# Getting Started

Install using *pip*:

```
pip install chainscan
```

For an easy start, see the examples.

# CHAPTER 3

# Features

Some of the notable features supported:

- Iterate over blocks in the longest chain
- Iterate over all blocks from all forks (in topological order)
- "Tracked spending": For each tx input, resolve the tx output spent by it
- Resumability: All iterators are resumable. You can pickle them, and later reload them, picking up from where you left off.
- Tailability: You can keep waiting for the next blocks to arrive. The iterator will return the next blocks/txs as they arrive (think *tail -f*, or MongoDB's tailable cursor)
- A *BlockChain* data structure, supporting block lookup by hash or height

# CHAPTER 4

# Design and Goals

ChainScan is focused on the (surprisingly complicated) task of iterating over the Bitcoin blockchain.

ChainScan *does not* aim at being “everything bitcoin” or a one-stop-shop solution. Various other python libraries already implement many of the tools you’d need for your bitcoin development tasks (python-bitcoinlib, python-libbitcoin, and pybitcointools, to name the main ones).

ChainScan aims at being simple-yet-powerful. This package has been carefully designed for simplicity, flexibility, extensibility, and customizability, so that it can be useful for a wide variaty of usages.

The basic entities (e.g., the *Block* and *Tx* classes) are deliberately simple and minimalistic. You’d often want to use ChainScan along with another library – ChainScan will take care of the looping, the other library with what you want to do *foreach block* or *foreach tx*. (See the examples.)

# CHAPTER 5

# Speed

ChainScan is fast because it reads the blockchain data directly from the block data files (*blk*.dat*), rather than using bitcoind's RPC, which is slow, and also subject to communication errors.

ChainScan is not super-fast, however, because it is written in python. Alternative implementations in C (for example) will likely be faster.

The reason python was chosen is for its ease-of-use. See the *design section* of the docs to get a better feeling of why python is good here.

That said, efforts have been made for making this library as fast as possible, without compromising its *design principles*. Some parts of the implementation are using Cython. Many more speed improvements are coming soon.

# CHAPTER 6

# Development Status

ChainScan is at an early stage of its development, yet the current release can already be useful to many.

The next releases may not be backward compatible.

*Bug reports, suggestions and contributions are appreciated.*

Issues are tracked on github.

# CHAPTER 7

# More

- genindex
- modindex
- search