




## TP 02: Bitcoin Block Parser (90 minutes)

5BLOC  TP 02: Bitcoin Block Parser (90 minutes)

Programming Language: Python

Time: 90 minutes

The goal of this workshop is to teach you how to **read and parse a Bitcoin block** manually using Python and cryptographic libraries. You will be given several raw data bytes of Bitcoin blocks, and your task is to extract relevant information by applying your knowledge of Bitcoin's structure, hashing functions, and cryptographic principles.

You have been given 2 files, `simple_btc.txt` and `pizza_btc.txt`. You will do all of the following steps for the two given files.

### Constraints:

- Coding AI generators are strictly forbidden as this is a learning exercise.
- No external libraries for Bitcoin parsing are allowed. You will only use cryptographic libraries.
- **Exception:** If you are stuck on a challenge and have not parsed the first block within an hour, you can request permission to use `bitcoinlib` to avoid falling behind.

## 1. Inspect the Raw Data (15 minutes)

Before beginning to parse the Bitcoin block programmatically, you must first **inspect the raw data** to identify the structure of the block. This will give you a better understanding of how the block is laid out and where key fields such as the **block header**, **Merkle root**, and **transactions** are located.

### Steps:

1. Open the given raw block data in `xxd` or [hexed.it](https://hexed.it) to visualize the hexadecimal representation of the block.
2. Identify the first 8 bytes containing the **magic number** and the **bloc size**, then the following **80 bytes** of the block, which constitute the **block header**. The header will include the version, previous block hash, Merkle root, timestamp, bits, nonce, and transaction count.
3. Identify the start of the **transaction data** (after the block header) and the structure of each transaction.

## 2. Block Header Parsing (40 minutes)

### Resources:

- <https://en.bitcoin.it/wiki/Block>

### Goal:

Parse the block header and extract the following elements:

- Magic Number
- Bloc size
- Version
- Previous block hash
- Merkle root
- Timestamp
- Bits
- Nonce
- Number of transactions

Leave transactions alone for now.

### Guidance:

### 1. Structure of a Bitcoin Block Header:

The block header consists of a fixed-length structure. You'll need to read the first 80 bytes of the block to extract these fields.

- **Version** is a 4-byte field.
- **Previous block hash** is a 32-byte field.
- **Merkle root** is another 32-byte field.
- **Timestamp** is 4 bytes.
- **Bits** (difficulty target) is 4 bytes.
- **Nonce** is 4 bytes.
- **Number of transactions** is variable, depending on the block size.

#### Pedagogical Tip:

Start by reading and interpreting raw binary data, understanding how these fields are encoded.

Use Python's built-in `struct` module for unpacking binary data into readable values.

## 3. Block Header Hash Calculation (15 minutes)

#### Resources:

- [https://github.com/bitcoinbook/bitcoinbook/blob/develop/ch06\\_transactions.adoc](https://github.com/bitcoinbook/bitcoinbook/blob/develop/ch06_transactions.adoc)
- [https://en.bitcoin.it/wiki/Block\\_hashing\\_algorithm](https://en.bitcoin.it/wiki/Block_hashing_algorithm)

#### Goal:

Compute the **Block header hash** of the block by hashing the raw transaction data. Verify your calculation by checking it against the block Hash in the block header.

## 4. Transaction Parsing (60 minutes)

#### Resources:

- [https://github.com/bitcoinbook/bitcoinbook/blob/develop/ch06\\_transactions.adoc](https://github.com/bitcoinbook/bitcoinbook/blob/develop/ch06_transactions.adoc)
- <https://en.bitcoin.it/wiki/Transaction>

#### Goal:

Parse each transaction to extract the following information:

- Lock time
- Transaction ID
- Transaction amount in satoshis
- `vout` (output index)
- `sequence` number

Leave scripts alone for now.

#### Guidance:

##### 1. Transaction Structure:

Bitcoin transactions consist of several fields, starting with an input list and an output list. You will need to first handle the **input list** (previous transactions being spent) and then the **output list** (new addresses receiving funds).

##### 2. Challenges in Parsing:

- Each transaction is preceded by a variable-length **version** field.
- The **inputs** include a previous output reference and a script signature (which can vary in length).
- The **outputs** contain amounts and scriptPubKeys.

#### Pedagogical Tip:

Break down the structure of each transaction. Use the Python `bitstring` library or `struct` module to handle variable-length fields effectively.

## 5. Merkle Root Calculation (30 minutes)

#### Resources:

- <https://brilliant.org/wiki/merkle-tree/>

- <https://medium.com/coinmonks/merkle-tree-a-simple-explanation-and-implementation-48903442bc08>

**Goal:**

Compute the **Merkle root** of the block by hashing the raw transaction data. Verify your calculation by checking it against the one you parsed previously.

**Guidance:**

**1. What is a Merkle Root?**

The Merkle root is a hash of all transactions in the block, used to verify that the transactions have not been tampered with.

**2. Steps to Compute the Merkle Root:**

- Hash each individual transaction.
- Pair adjacent hashes and hash them together, continuing until you're left with a single hash, the Merkle root.

## 6. Script Decoding and Transaction Fee Extraction (40 minutes)

**Resources:**

- <https://en.bitcoin.it/wiki/Script>

**Goal:**

Decode the **scripts** in the transactions to extract the following:

- Transaction fees
- Destination address
- Sender address
- Script type

Other data or content of the script should not be looked. (Of course you can do it as a bonus.)

**Guidance:**

**Try to find the script type you have on the two given blocks, and handle only the two script types.**

**1. Understanding Scripts:**

Bitcoin scripts can be complex. You'll encounter **scriptSig** (input scripts) and **scriptPubKey** (output scripts). These scripts often contain operations (OP codes) that must be parsed to understand the conditions of the transaction.

**2. Transaction Fees:**

Transaction fees can be derived by subtracting the sum of outputs from the sum of inputs. Focus on understanding how to decode and calculate the fees based on input/output scripts.

## Final Reflection (10 minutes)

- Reflect on how Bitcoin's decentralization and cryptographic principles ensure security and integrity.
- Discuss challenges you faced while manually parsing the blocks and how understanding Bitcoin's internals is essential for anyone working with blockchain technology.

Activité précédente

Chapitre 02 - Structure générale d'un bloc  
Bitcoin

Aller à...



Activité suivante


Chapitre 03: IPFS

Contactez-nous



Suivez-nous



 [Contacter l'assistance du site](#)

Connecté sous le nom « [Ibrar Hamouda](#) »  
([Déconnexion](#))