Session 2 Implementing a simple Blockchain using Python on Flask

Implementation on AWS Ubuntu and on Windows 8/10

Joshi

Feb 2018

Agenda

- Revision of Blockchain functions that we did in last session
- Looking at another Blockchain in Jupyter Notebook
- Proof of work by division / other than adding to the blockchain. Miners Getting Coins
- Appending the blockchain using Flask
- Gentle Introduction to Flask and CURL
- Running the .py file on your system and opening 127.0.0.1:5000
- Understanding Four functions and Three nodes used

- Reference: https://dev.to/aunyks/lets-make-the-tiniest-blockchain-bigger

Recap

- Looking at the old functions
- Final .py is partly based on the what we did last time
- Generating and accessing the block chain
- Trying two more small codes

Prerequisite

- Flask Libarires: pip install flask # Install our web server framework first
- Flask is a micro web framework written in Python, is a micro framework because it does not require particular tools or libraries It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions.
- curl is a tool to transfer data from or to a server, using one of the supported protocols (HTTP, HTTPS, FTP, FTPS, SCP, SFTP, TFTP, DICT, TELNET, LDAP or FILE). The command is designed to work without user interaction.

Three URLs / Nodes

- @node.route('/txion', methods=['POST'])

Gets initialized when we pass on the transaction details

- @node.route('/mine', methods = ['GET'])

Mines the transaction, proof of work and then adds it to the block chain

- @node.route('/blocks', methods=['GET'])

Display the final blockchain

GET vs POST

- GET typically has relevant information in the body of the request. (A GET should not have a body, so aside from cookies, the only place to pass info is in the URL.)
- Beside keeping the URL relatively cleaner, POST also lets you send much more information (as URLs are limited in length, for all practical purposes), and lets you send just about any type of data (file upload forms, for example, can't use GET -- they have to use POST plus a special content type/encoding).
- Aside from that, a POST connotes that the request will change something, and shouldn't be redone willy-nilly. That's why you sometimes see your browser asking you if you want to resubmit form data when you hit the "back" button.
- GET, on the other hand, should be idempotent -- meaning you could do it a million times and the server will do the same thing (and show basically the same result) each and every time.

Four Functions

- def transaction()
- def mine()
- def proof_of_work(last_proof)
- def get_blocks()

Mining

Mining happens after someone submits a request.

1. Proof of work: Divisible by 9 and divisible by last proof of work
2. Adding to new block chain
3. Miner get rewarded

```
mined_block = Block( new_block_index,
    new_block_timestamp,
    new_block_data,
    last_block_hash )
blockchain.append(mined_block)
```

Two Changes to run on Python3 for Windows 8/10

- sha.update(str(self.index).encode('utf-8') + str(self.timestamp).encode('utf-8') + str(self.data).encode('utf-8') + str(self.previous_hash).encode('utf-8'))
- C:\Users\xxx.XXX> curl.exe "http://localhost:5000/txion" -H "Content-Type: application/json" -d '{\"from\":\"akjflw\" , \"to\":\"fjlakdj\", \"amount\": 33333333}' Transaction submission successful PS C:\Users\xxx.XXX>

```
curl "http://localhost:5000/txion" -d "{\"from\": \"akjflw\", \"to\":\"fjlakdj\", \"amount\": 3}" -H "Content-Type: application/json"
```

Viewing Block

- Two transaction every time we mine

- curl localhost:5000/blocks (UBUNTU)
- Curl.exe localhost:5000/blocks (Windows Power Shell)

References

- https://dev.to/aunyks/lets-make-the-tiniest-blockchain-bigger
- https://gist.github.com/aunyks

Possible Errors due to string passing in Windows

<h1>Method Not Allowed</h1>
The method is not allowed for the requested URL.

[1/3]: from: akjflw --> <stdout>
--_curl_--from: akjflw
curl: (3) Port number ended with ' '

[2/3]: to:fjlakdj --> <stdout>
--_curl_-- to:fjlakdj
curl: (3) Port number ended with 'f'

[3/3]: amount: 33333333 --> <stdout>
--_curl_-- amount: 33333333