网络与信息安全课内实验五——

区块链

1. 实验目的

1. 了解区块链是个什么东西。

2. 搭建一个自己的区块链。

二．实验内容

###### 2.创造一个自己的区块链

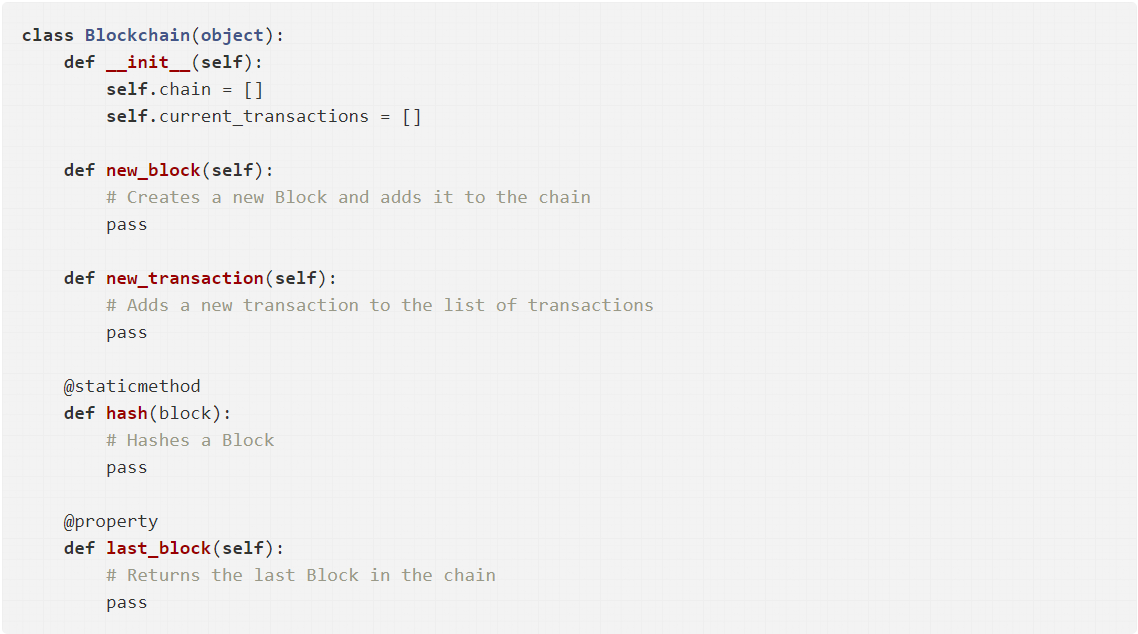
目前搭建区块链的方法有很多，包括用c++、java、python、ruby等等均有开源实现。这次，我们参考<https://segmentfault.com/a/1190000014483104>这篇文档，尝试搭建一个区块链。

先观察一个区块（block）的格式：

一个block中有5个变量：index、timestamp（时间）、transactions（交易）、proof（工作量证明）、previous\_hash。



一个类区块链的实例如下：



按照链接中的说明进行填充之后，再加入工作量证明机制。

工作量证明系统主要特征是客户端需要做一定难度的工作得出一个结果，验证方却很容易通过结果来检查出客户端是不是做了相应的工作。这种方案的一个核心特征是不对称性：工作对于请求方是适中的，对于验证方则是易于验证的。它与验证码不同，验证码的设计出发点是易于被人类解决而不易被计算机解决。

\*\*\*这里查看实验报告要求2.

Blockchain 作为 API 接口

我们将使用 Python Flask 框架，这是一个轻量 Web 应用框架，它方便将网络请求映射到 Python 函数，现在我们来让 Blockchain 运行在基于 Flask web 上。

我们将创建三个接口：

/transactions/new 创建一个交易并添加到区块

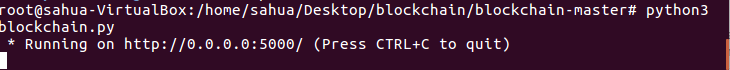
/mine 告诉服务器去挖掘新的区块

/chain 返回整个区块链

一般而言，这一步的目的是让所有人可以访问这个区块链，并不需要将new和mine公开。不过现在为了操作方便，就这样了吧。

运行区块链

运行python3 blockchain.py



错误提示：print(f’’)错误，则需要升级python3到3.6版本以上

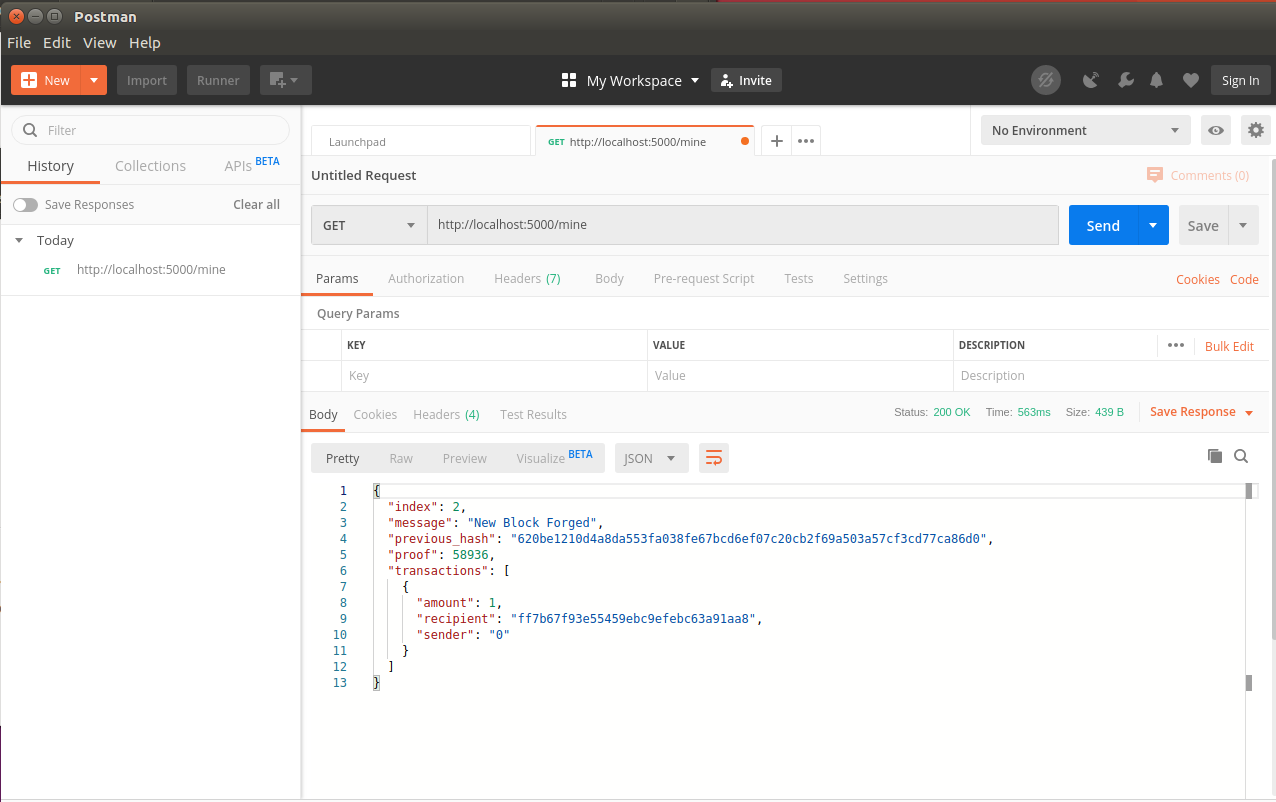
错误提示：import错误，安装所需的库。

现在我们用postman来进行操作。

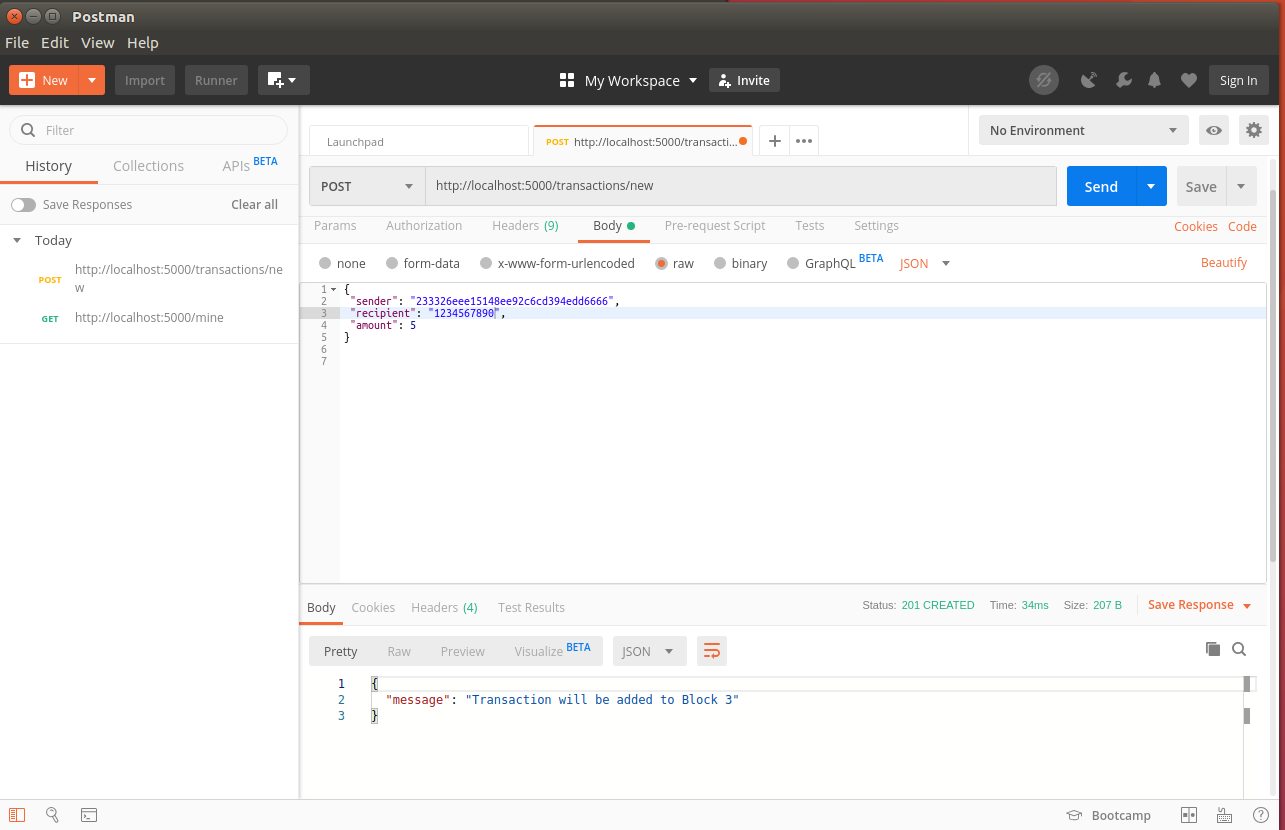
附：postman Ubuntu 安装方法<https://blog.csdn.net/xiamoyanyulrq/article/details/80334900>

附2：不需要注册，弹出界面直接叉掉。

再postman中输入新建一个请求，并访问http://localhost:5000/mine，如图所示。



同理，post请求也能正常运行。



节点一致性（共识）

我们已经有了一个基本的区块链可以接受交易和挖矿。但是区块链系统应该是分布式的。既然是分布式的，那么我们究竟拿什么保证所有节点有同样的链呢？这就是一致性问题，我们要想在网络上有多个节点，就必须实现一个一致性的算法。

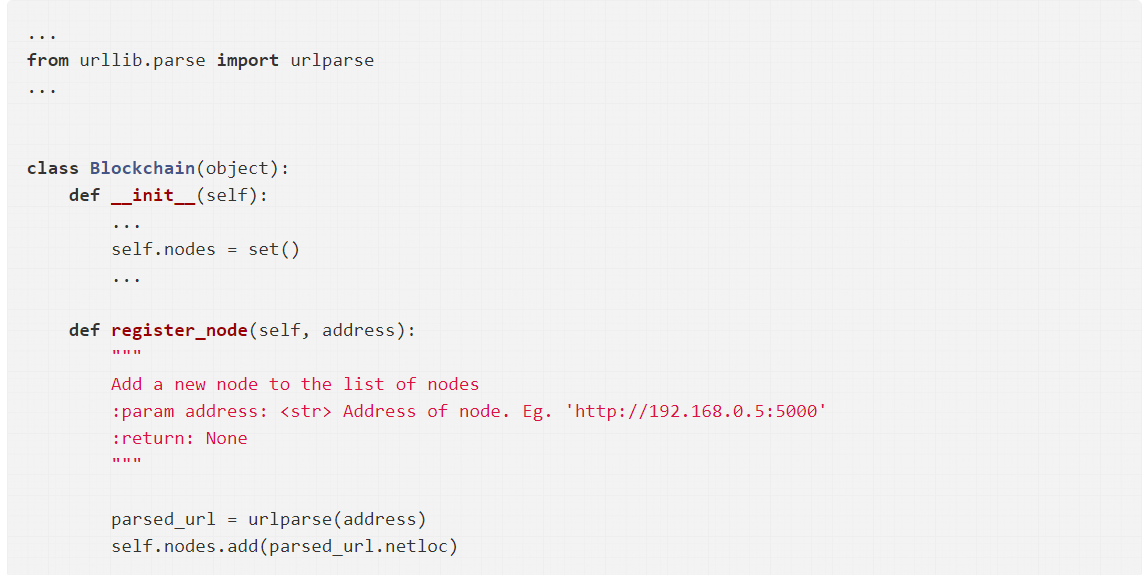
当一个节点与另一个节点有不同的链时，就会产生冲突。 为了解决这个问题，我们将制定最长的有效链条是最权威的规则。换句话说就是：在这个网络里最长的链就是最权威的。 我们将使用这个算法，在网络中的节点之间达成共识。

注册节点

在实现一致性算法之前，我们需要找到一种方式让一个节点知道它相邻的节点。每个节点都需要保存一份包含网络中其它节点的记录。因此让我们新增几个接口：

/nodes/register 接收 URL 形式的新节点列表.

/nodes/resolve 执行一致性算法，解决任何冲突，确保节点拥有正确的链.



这篇文章中的proof\_of\_work写的有一点问题，请说明它有什么问题，可能导致什么结构，并进行修正，附在实验报告中。

def proof\_of\_work(self, last\_proof):

"""

Simple Proof of Work Algorithm:

- Find a number p' such that hash(pp') contains leading 4 zeroes, where p is the previous p'

- p is the previous proof, and p' is the new proof

:param last\_proof: <int>

:return: <int>

"""

proof = 0

while self.valid\_proof(last\_proof, proof) is False:

proof += 1

return proof

@staticmethod

def valid\_proof(last\_proof, proof):

"""

Validates the Proof: Does hash(last\_proof, proof) contain 4 leading zeroes?

:param last\_proof: <int> Previous Proof

:param proof: <int> Current Proof

:return: <bool> True if correct, False if not.

"""

guess = f'{last\_proof}{proof}'.encode()

guess\_hash = hashlib.sha256(guess).hexdigest()

return guess\_hash[:4] == "0000"

1. 实验总结

在此次实验中，由于代码等都是由教程给出的，所以理解起来也不复杂，本次实验让我最难解决的地方反而是环境的安装，遇到了许多的问题，比如pip安装flask等库时遇到的一系列报错，包括了python文件夹权限不够，源文件中有些需要修改的地方，以及下载需要更换到国内源等问题。在使用flask时jinja2中已经没有‘Markup’的模块，又需要重新安装jinjia2等。教程中的flask的版本过低，安装起来又出现了许多的问题，【ImportError: cannot import name ‘json‘ from ‘itsdangerous‘】等，所以使用的是2.0.2的版本。

区块链，我用自己的认识来描述一下，首先提到区块链，大家应该印象最深刻的无非是比特币，以太坊，这些代币。其实区块链应用最值得去开发就是他的公平性，不可复制性，以及不可篡改性。公平性，其实也就是去中心化。不可复制性，也是他的唯一性。而不可篡改也是我觉得的最重要的地方。