**Workshop**

Problems for exercises and homework for the [“Blockchain Basics” mini course](https://github.com/mdamyanova/Blockchain-Basics-Mini-Course).

**Blockchain Network**

In our workshop we will **create a Blockchain network**. Keep in mind that this exercise is just to see how “**the code goes through your hands**”. It is not necessary to understand **any part** of the code. Just **follow the steps** and see if you like coding and Blockchain!

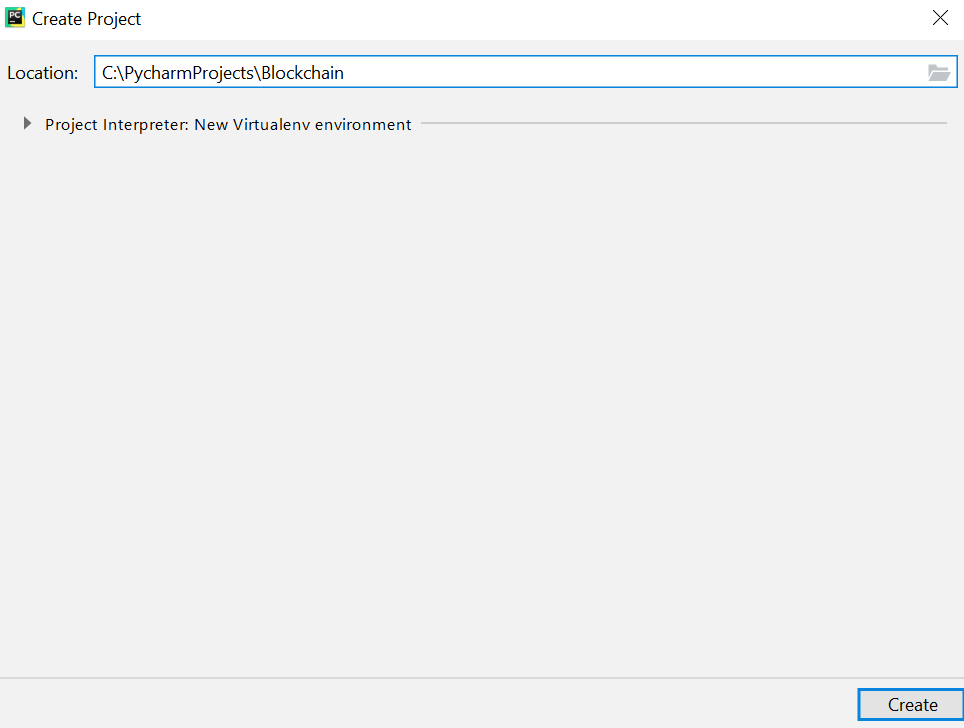
1. **Python IDE**

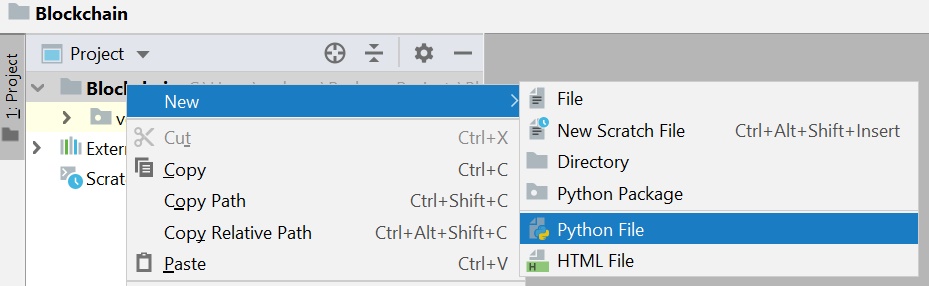
We will use **the programming language Python**. The first step is to choose where to write the code. **IDE** is an **I**ntegrated **D**evelopment **E**nvironment. This is a special program for writing code. For **Python** we can use [PyCharm](https://www.jetbrains.com/pycharm/) but notice that we have to **install** it to our computer. A good **online** alternative is [Repl](https://repl.it/languages/python3) where you can write code online without installing anything. It’s your choice!

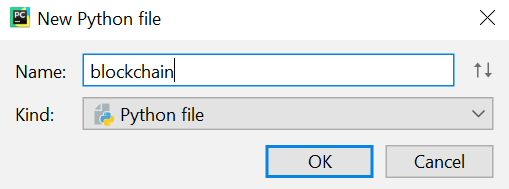
**NOTE:** If you choose **PyCharm**, you should **download Python** too (below I use PyCharm).

1. **Creating blockchain.py File**

Once you start the chosen **IDE**, create a new **Python File** called **blockchain.py**.

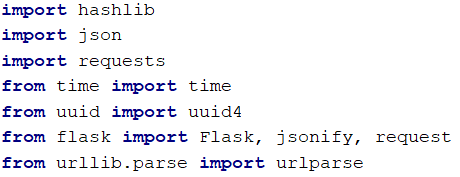






1. **Imports**

We should import some **packages** that will help us to write the code for the **Blockchain**. On **the first lines** add this code:



**NOTE:** You have to click on the **red lamp** and **install the packages**.

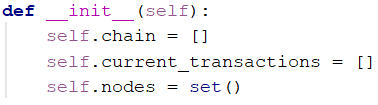
1. **Blockchain Class**

It’s time to create our **Blockchain class**! Skip one row and write **class Blockchain**. Our class should have functions. They will be called **\_\_init\_\_, register\_node, new\_block, new\_transaction, hash, last\_block, proof\_of\_work, valid\_proof, valid\_chain, resolve\_conflicts**. It’s not a problem if you don’t know how to write all of this! At this moment, our class should looks like this:



1. **Function \_\_init\_\_(self)**

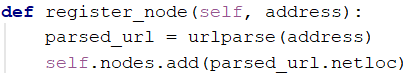
This is our **main function**. It shows what we have in the class and what we can use in the other functions. For this purpose we should have a **chain**, **current\_transactions** and **nodes**.



The **chain** and the **transactions** are simply **empty arrays** (this means sequence of elements). And the nodes you can imagine that they are **each computer in the network**.

1. **Function register\_node(self, address)**

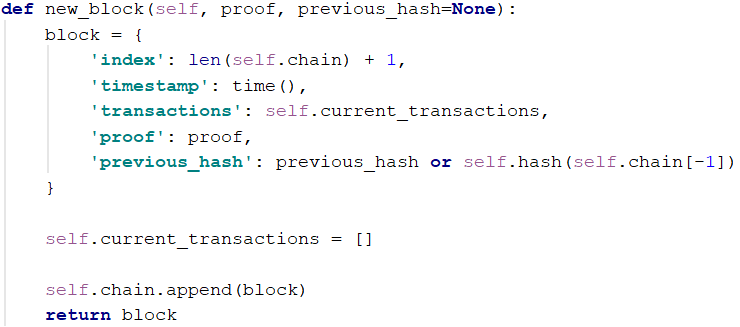
This function is responsible for **registering a new node to the network**. Add this code to the function.



**NOTE:** Notice the **space** in each row. Spaces are **really important** in **Python**. You should follow them **exactly as it is shown**.

1. **Function new\_block(self, proof, previous\_hash=None)**

In this function is the magic for **creating a new block**. We have to **create a new block and add it to the chain**. We will also **return the new block**. Write the following code:

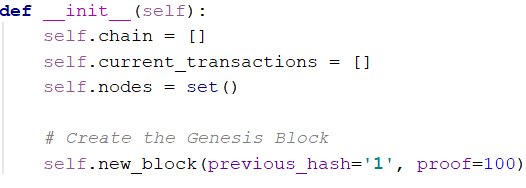


Every block in our network should have **index**, **timestamp**, **transactions**, **proof** and **previous hash**. After that we **set the current transactions to be empty** (because they are now in the new block) and we **append the block to our chain**. Finally, we simply **return the block**.

1. **Creating the Genesis Block**

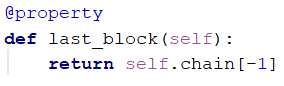
In **Blockchain**, the **Genesis Block** is **the first block which is in the network**. Let’s create it!

We have to go back to **\_\_init\_\_** function and add the following code.



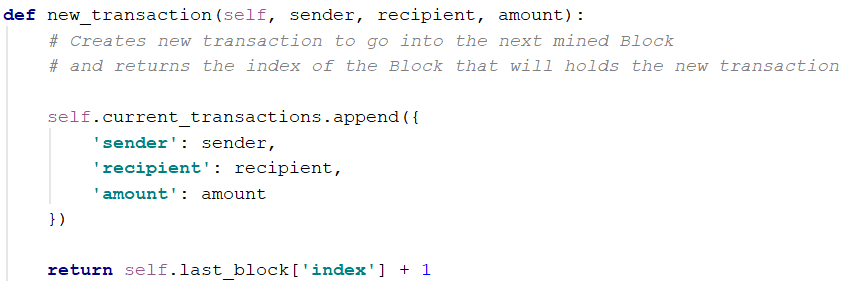
1. **Function new\_transaction(self, sender, recipient, amount)**

Here we can **create new transactions** and **add them to the current transactions**. After that, our function will **return the index of the last block in the chain**. First, let’s create the so called **property**. It has the same structure as a function.



This property will **return the last block**.

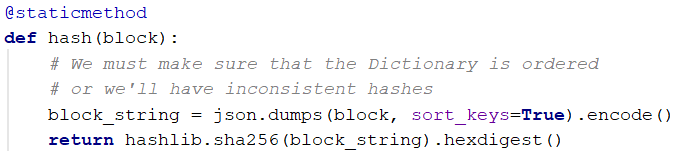
After we have **the last block**, we can continue writing the logic for **new transaction**. It’s quite simple – we **append** the information which is given to **the current transactions**.



1. **Function hash(block)**

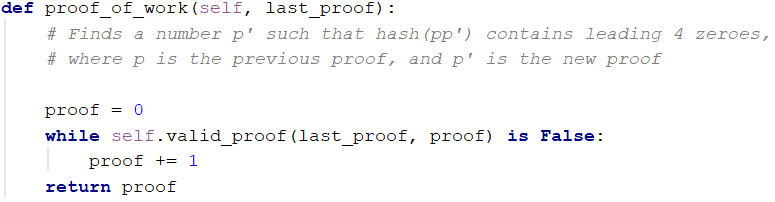
**Hashing** is an **important** part of the **Blockchain**. The blocks are linked as every block gets a **unique** (digital) **signature** that corresponds to exactly **the string of data** in that block. If anything inside a block changes, even just a single digit change, the block will get a new signature. This happens through **hashing**.

In this function we will hash the given block. First, we will sort all signatures and we will **return the hashed block’s signature**.

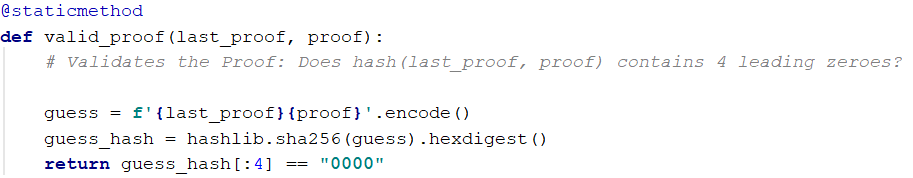


1. **Function proof\_of\_work(self, last\_proof)**

This is the function with our **Proof of Work**. The code should looks like this:

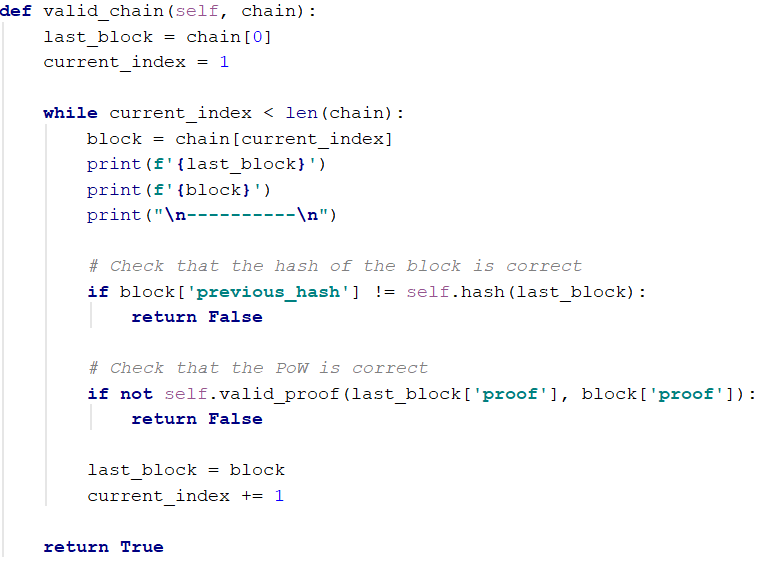


You can see we are using a **valid\_proof** function but we don’t have it yet. Let’s change that and write it. This function will be **static** and it will simply check **if the hash contains 4 leading zeroes**.



1. **Function valid\_chain(self, chain)**

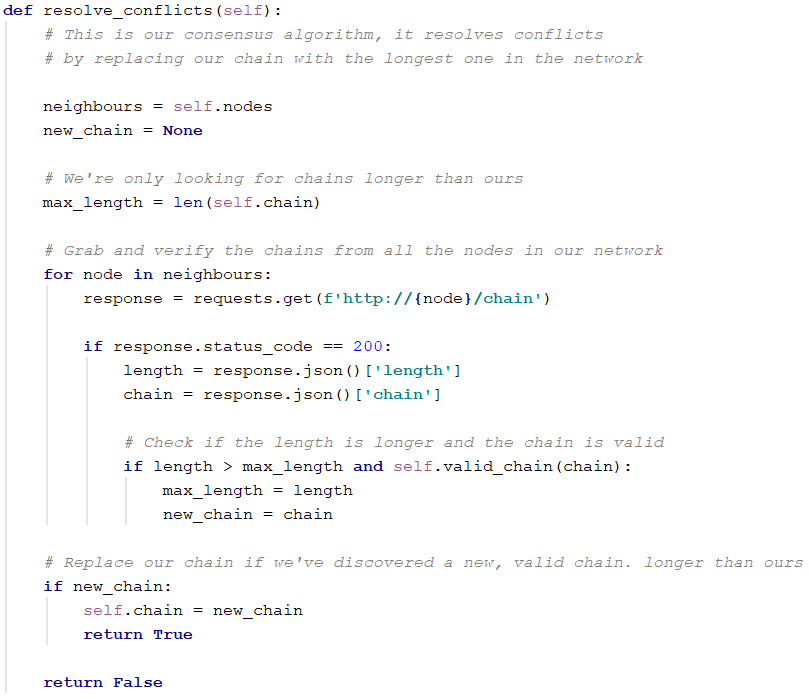
The **valid\_chain** function will **validate our chain**. The code should looks like this:



If everything is okay, we will **return True**.

1. **Function resolve\_conflicts(self)**

This is our **consensus algorithm** for resolving conflicts by **replacing our chain with the longest one in the network**. Here we will use our **valid\_chain** function to **verify the chains from all the nodes in our network**. Write this code:

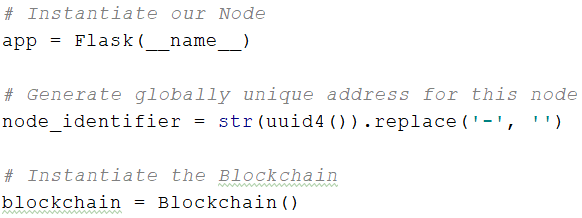


With this our **Blockchain** class is done!

1. **Main Class**

For writing outside the Blockchain class (our main class where we will run the Blockchain) we should write **without space**. We will use **Flask**. **Flask** is a micro web **framework** written in **Python** for developing web applications**.**

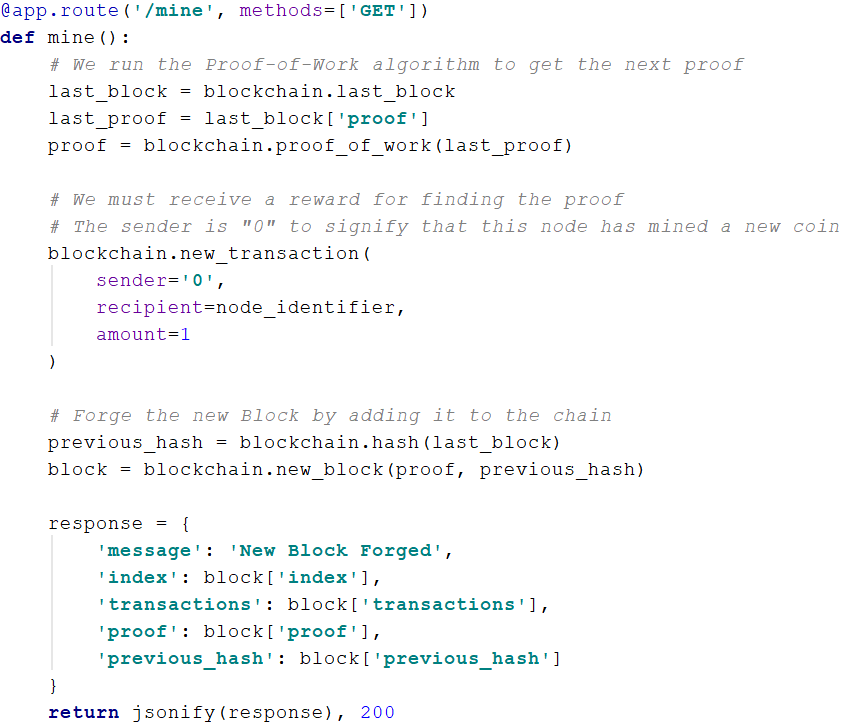
First**,** we will create an **app** using **Flask.** Next, we will create a **node\_identifier**. This will generate unique addresses for each node in the network. And finally, we will **instantiate our Blockchain**.



It’s time to create our **routes** (the URL addresses)!

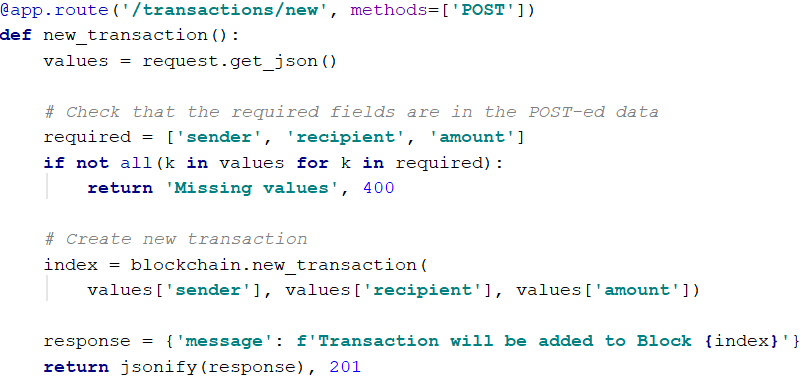
1. **Route /mine**

This is the **/mine** route where we will **run the** **PoW** algorithm and we will **receive** **a reward**. We will **add the** **new block** to the chain and we will **return an information about it**.



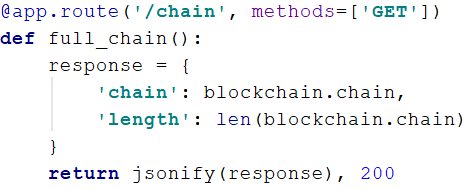
1. **Route /transactions/new**

In the **/transactions/new** route we will simply **create a new transaction**. We will check if all the **required fields** are sent and if they are, we will create the transaction and we will **return an informative message** that the transaction was added to the block.



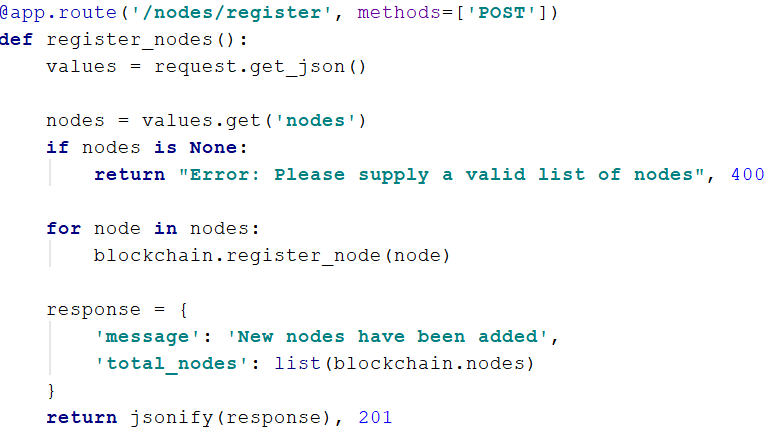
1. **Route /chain**

The **/chain** route will simply return to us **the chain**.



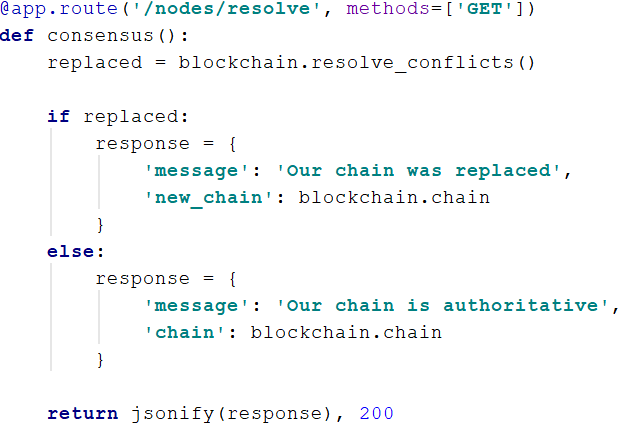
1. **Route /nodes/register**

In **/nodes/register** route we will add new nodes to the network. Here we will also **return an informative message.**



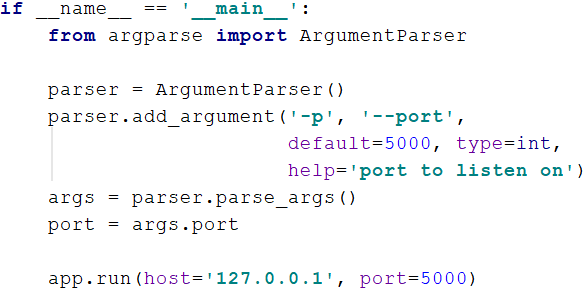
1. **Route /nodes/resolve**

The **/nodes/resolve** route will be responsible for **resolving conflicts**. We will **return an informative message together with our Blockchain**.



1. **Run the Blockchain**

This is the part of the code where **we can run the application**. After you run it (either you click on the green arrow or with right button – “run blockchain”) you can go to your browser and type <http://127.0.0.1:5000/chain> and see the result. You can test the other routes too.



Congratulations! You are now part of the people in the world who know Blockchain! 😊