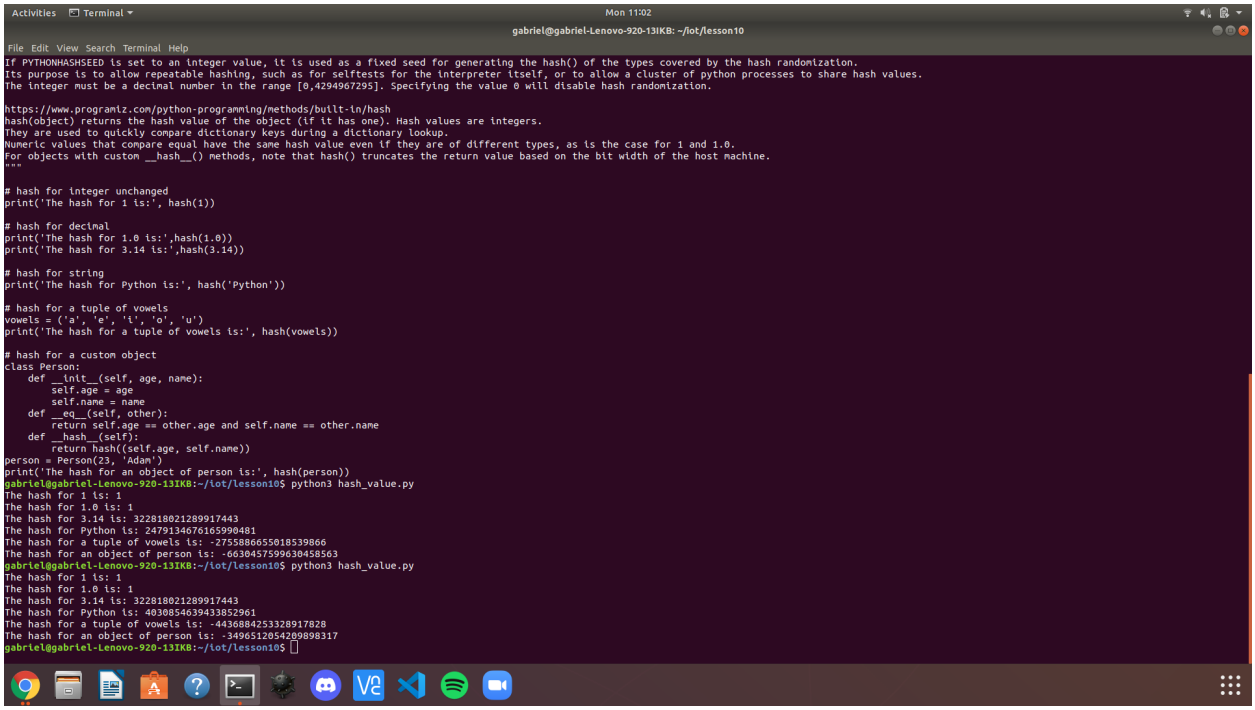
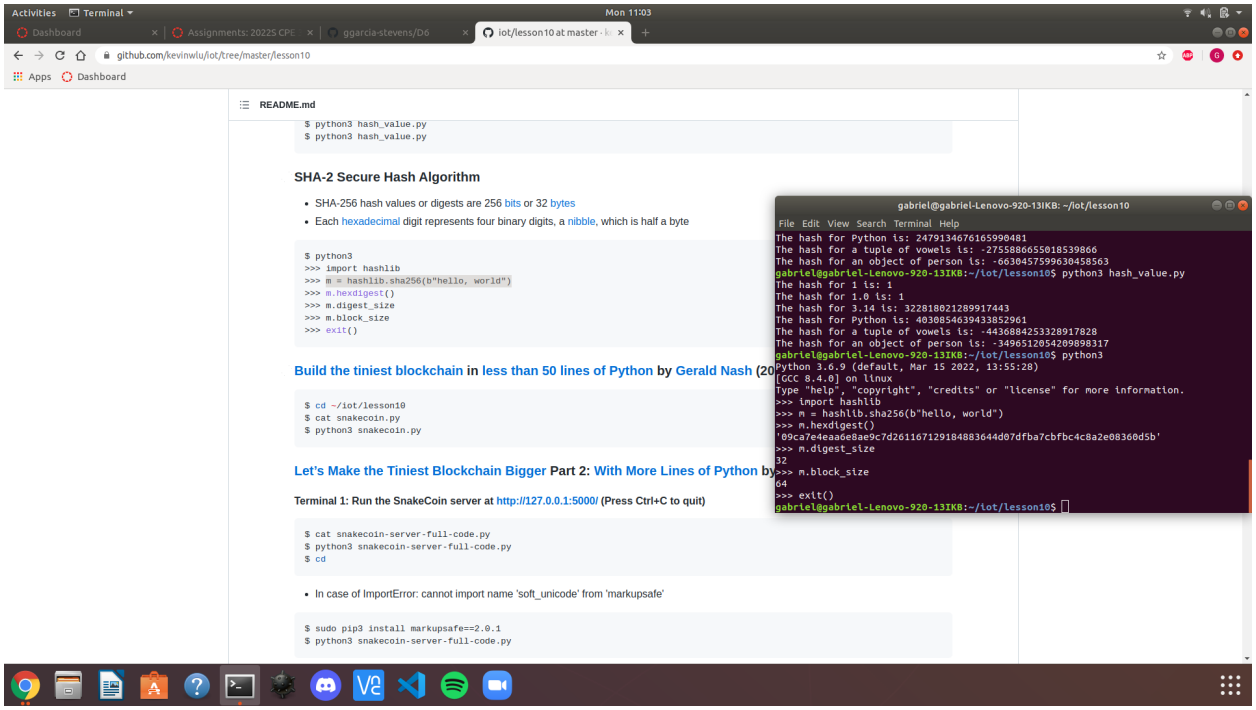
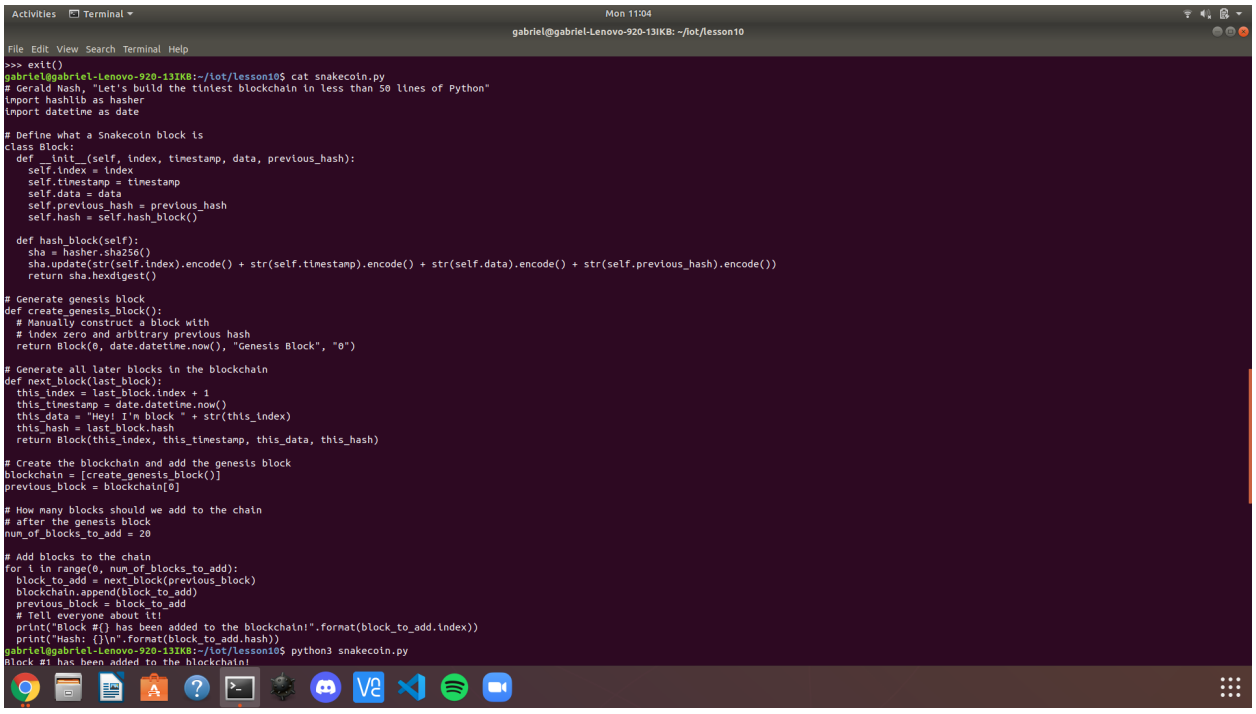
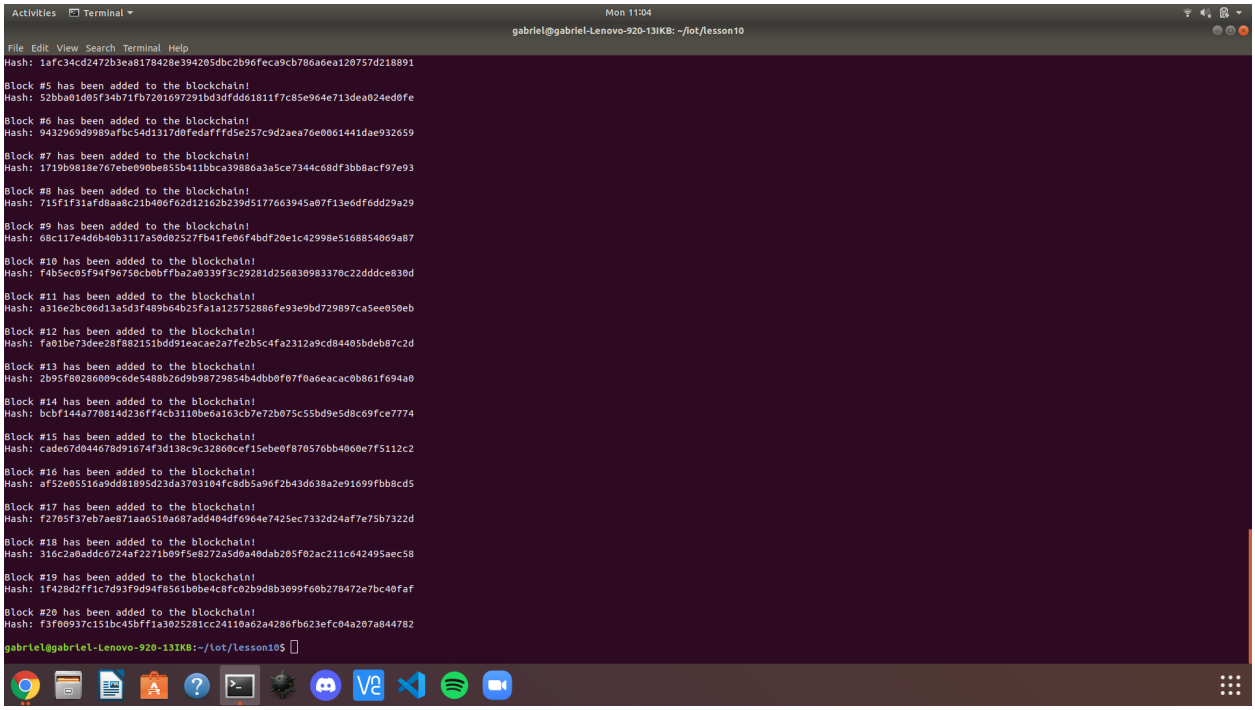
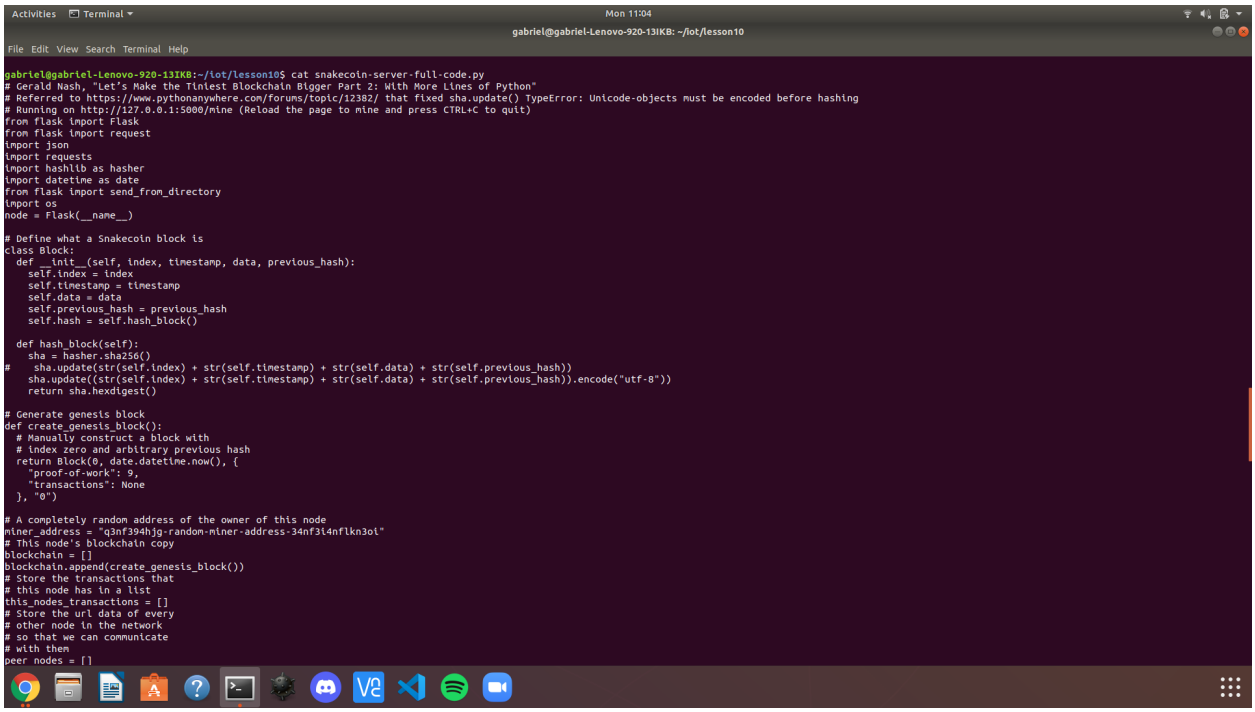
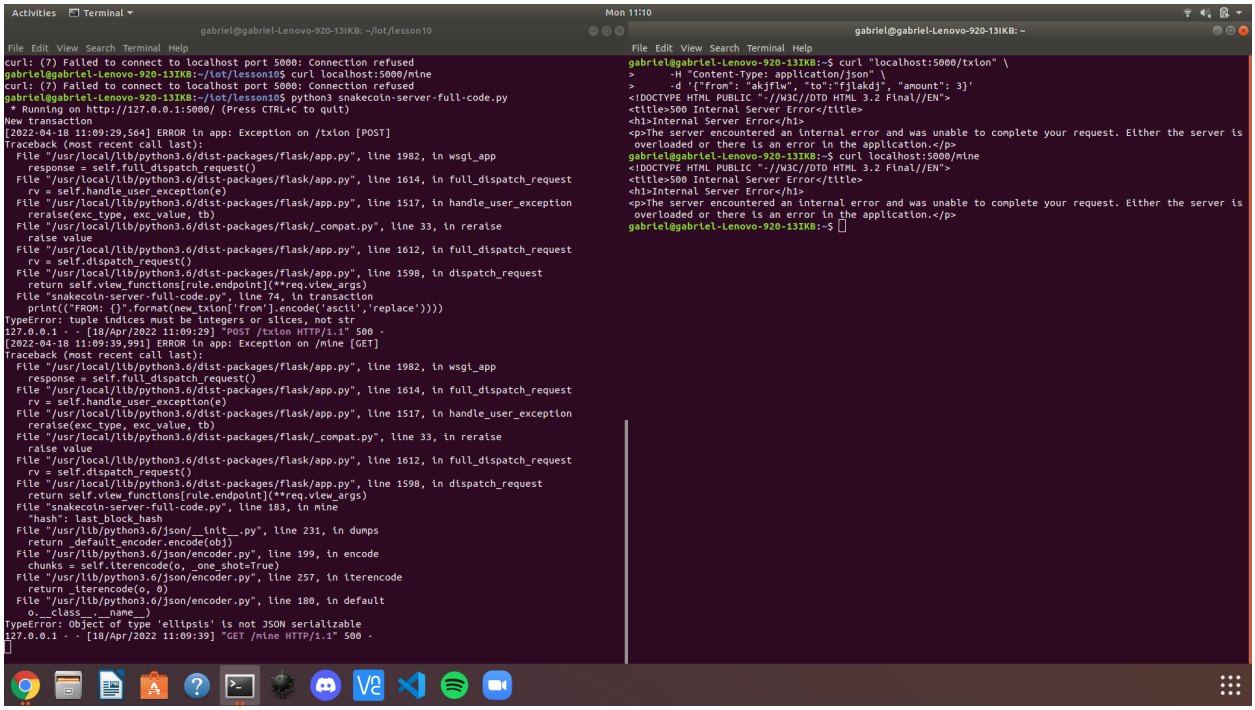

Activities
Terminal
Mon 11:02
gabriel@gabriel-Lenovo-920-13IKB: ~/Iot/lesson10
File Edit View Search Terminal Help
If PYTHONHASHSEED is set to an integer value, it is used as a fixed seed for generating the hash() of the types covered by the hash randomization.
Its purpose is to allow repeatable hashing, such as for selftests for the interpreter itself, or to allow a cluster of python processes to share hash values.
The integer must be a decimal number in the range [0,4294967295]. Specifying the value 0 will disable hash randomization.
https://www.programiz.com/python-programming/methods/built-in/hash
hash(object) returns the hash value of the object (if it has one). Hash values are integers.
They are used to quickly compare dictionary keys during a dictionary lookup.
Numeric values that compare equal have the same hash value even if they are of different types, as is the case for 1 and 1.0.
For objects with custom __hash__() methods, note that hash() truncates the return value based on the bit width of the host machine.
"""
# hash for integer unchanged
print('The hash for 1 is:', hash(1))
# hash for decimal
print('The hash for 1.0 is:',hash(1.0))
print('The hash for 3.14 is:',hash(3.14))
# hash for string
print('The hash for Python is:', hash('Python'))
# hash for a tuple of vowels
vowels = ('a', 'e', 'i', 'o', 'u')
print('The hash for a tuple of vowels is:', hash(vowels))
# hash for a custom object
class Person:
def __init__(self, age, name):
self.age = age
self.name = name
def __eq__(self, other):
return self.age == other.age and self.name == other.name
def __hash__(self):
return hash((self.age, self.name))
person = Person(23, 'Adam')
print('The hash for an object of person is:', hash(person))
gabriel@gabriel-Lenovo-920-13IKB:~/Iot/lesson10$ python3 hash_value.py
The hash for 1 is: 1
The hash for 1.0 is: 1
The hash for 3.14 is: 322818021289917443
The hash for Python is: 2479134676165990481
The hash for a tuple of vowels is: -2755886655018539866
The hash for an object of person is: -6630457599630458563
gabriel@gabriel-Lenovo-920-13IKB:~/Iot/lesson10$ python3 hash_value.py
The hash for 1 is: 1
The hash for 1.0 is: 1
The hash for 3.14 is: 322818021289917443
The hash for Python is: 4030854639433852961
The hash for a tuple of vowels is: -4436884253328917828
The hash for an object of person is: -3496512054209898317
gabriel@gabriel-Lenovo-920-13IKB:~/Iot/lesson10$

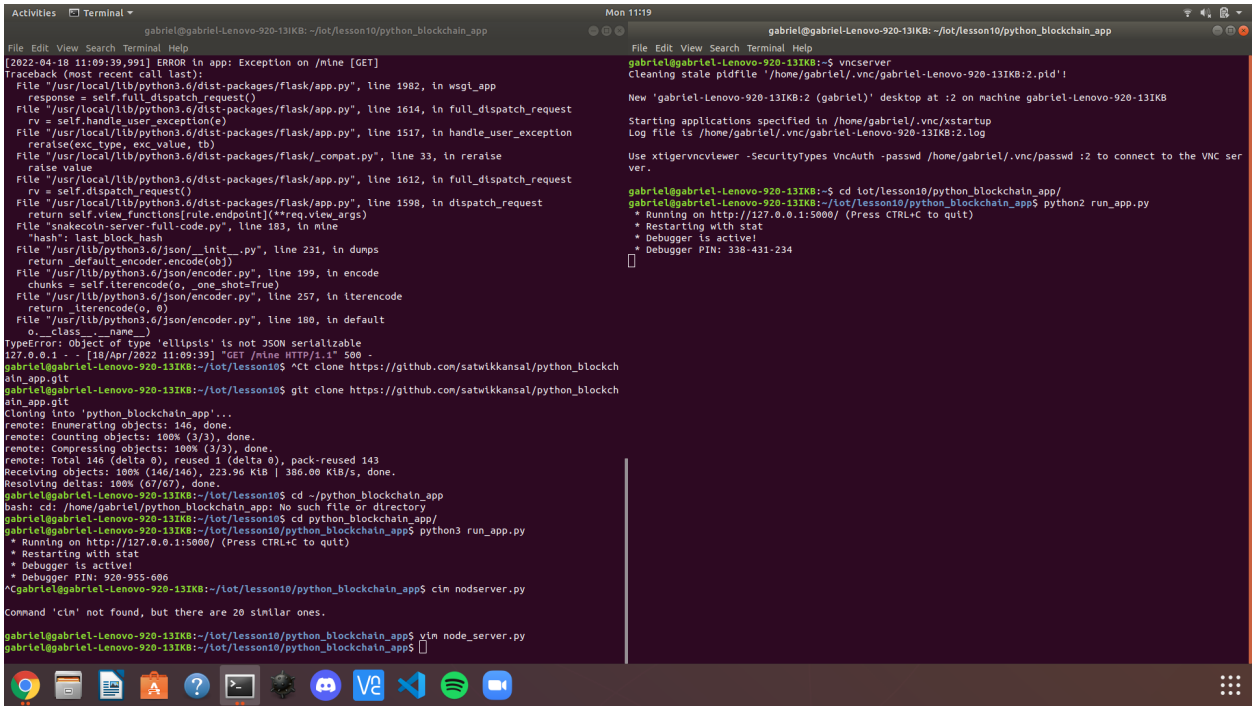

Activities
Terminal
Mon 11:03
Dashboard
Assignments: 20225 CPE
ggarcia-stevens/D6
iot/lesson10 at master · k
github.com/kevinwlu/iot/tree/master/lesson10
Apps
Dashboard
README.md
$ python3 hash_value.py
$ python3 hash_value.py
SHA-2 Secure Hash Algorithm
SHA-256 hash values or digests are 256 bits or 32 bytes
Each hexadecimal digit represents four binary digits, a nibble, which is half a byte
$ python3
>>> import hashlib
>>> m = hashlib.sha256(b"hello, world")
>>> m.hexdigest()
>>> m.digest_size
>>> m.block_size
>>> exit()
Build the tiniest blockchain in less than 50 lines of Python by Gerald Nash (20
$ cd ~/iot/lesson10
$ cat snakecoin.py
$ python3 snakecoin.py
Let's Make the Tiniest Blockchain Bigger Part 2: With More Lines of Python by
Terminal 1: Run the SnakeCoin server at http://127.0.0.1:5000/ (Press Ctrl+C to quit)
$ cat snakecoin-server-full-code.py
$ python3 snakecoin-server-full-code.py
$ cd
In case of ImportError: cannot import name 'soft_unicode' from 'markupsafe'
$ sudo pip3 install markupsafe==2.0.1
$ python3 snakecoin-server-full-code.py
gabriel@gabriel-Lenovo-920-13IKB: ~/iot/lesson10
File Edit View Search Terminal Help
The hash for Python is: 2479134676165990481
The hash for a tuple of vowels is: -2755886655018539866
The hash for an object of person is: -6630457599630458563
gabriel@gabriel-Lenovo-920-13IKB:~/iot/lesson10$ python3 hash_value.py
The hash for 1 is: 1
The hash for 1.0 is: 1
The hash for 3.14 is: 322818021289917443
The hash for Python is: 4030854639433852961
The hash for a tuple of vowels is: -4436884253328917828
The hash for an object of person is: -3496512054209898317
gabriel@gabriel-Lenovo-920-13IKB:~/iot/lesson10$ python3
Python 3.6.9 (default, Mar 15 2022, 13:55:28)
[GCC 8.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import hashlib
>>> m = hashlib.sha256(b"hello, world")
>>> m.hexdigest()
'09ca7e4eaa6e8ae9c7d261167129184883644d07dfba7cbfbc4c8a2e08360d5b'
>>> m.digest_size
32
>>> m.block_size
64
>>> exit()
gabriel@gabriel-Lenovo-920-13IKB:~/iot/lesson10$

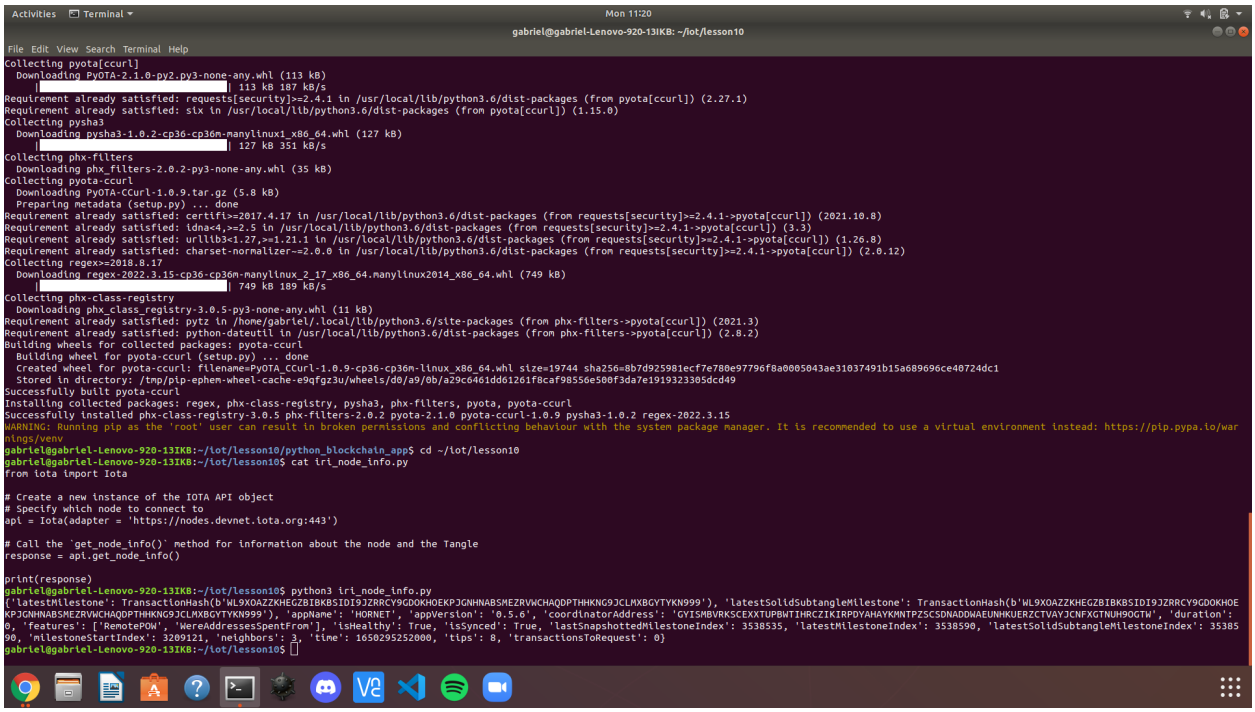

Activities
Terminal
Mon 11:04
gabriel@gabriel-Lenovo-920-13IKB: ~/iot/lesson10
File Edit View Search Terminal Help
>>> exit()
gabriel@gabriel-Lenovo-920-13IKB:~/iot/lesson10$ cat snakecoin.py
# Gerald Nash, "Let's build the tiniest blockchain in less than 50 lines of Python"
import hashlib as hasher
import datetime as date
# Define what a Snakecoin block is
class Block:
def __init__(self, index, timestamp, data, previous_hash):
self.index = index
self.timestamp = timestamp
self.data = data
self.previous_hash = previous_hash
self.hash = self.hash_block()
def hash_block(self):
sha = hasher.sha256()
sha.update(str(self.index).encode() + str(self.timestamp).encode() + str(self.data).encode() + str(self.previous_hash).encode())
return sha.hexdigest()
# Generate genesis block
def create_genesis_block():
# Manually construct a block with
# index zero and arbitrary previous hash
return Block(0, date.datetime.now(), "Genesis Block", "0")
# Generate all later blocks in the blockchain
def next_block(last_block):
this_index = last_block.index + 1
this_timestamp = date.datetime.now()
this_data = "Hey! I'm block " + str(this_index)
this_hash = last_block.hash
return Block(this_index, this_timestamp, this_data, this_hash)
# Create the blockchain and add the genesis block
blockchain = [create_genesis_block()]
previous_block = blockchain[0]
# How many blocks should we add to the chain
# after the genesis block
num_of_blocks_to_add = 20
# Add blocks to the chain
for i in range(0, num_of_blocks_to_add):
block_to_add = next_block(previous_block)
blockchain.append(block_to_add)
previous_block = block_to_add
# Tell everyone about it!
print("Block #{} has been added to the blockchain!".format(block_to_add.index))
print("Hash: {}\n".format(block_to_add.hash))
gabriel@gabriel-Lenovo-920-13IKB:~/iot/lesson10$ python3 snakecoin.py
Block #1 has been added to the blockchain!


Activities
Terminal
Mon 11:04
gabriel@gabriel-Lenovo-920-13IKB: ~/iot/lesson10
File Edit View Search Terminal Help
Hash: 1afc34cd2472b3ea8178428e394205dbc2b96feca9cb786a6ea120757d218891
Block #5 has been added to the blockchain!
Hash: 52bba01d05f34b71fb7201697291bd3dfdd61811f7c85e964e713dea024ed0fe
Block #6 has been added to the blockchain!
Hash: 9432969d9989afbc54d1317d0fedafffd5e257c9d2aea76e0061441dae932659
Block #7 has been added to the blockchain!
Hash: 1719b9818e767ebe090be855b411bbca39886a3a5ce7344c68df3bb8acf97e93
Block #8 has been added to the blockchain!
Hash: 715f1f31afd8aa8c21b406f62d12162b239d5177663945a07f13e6df6dd29a29
Block #9 has been added to the blockchain!
Hash: 68c117e4d6b40b3117a50d02527fb41fe06f4bdf20e1c42998e5168854069a87
Block #10 has been added to the blockchain!
Hash: f4b5ec05f94f96750cb0bffba2a0339f3c29281d256830983370c22dddce830d
Block #11 has been added to the blockchain!
Hash: a316e2bc06d13a5d3f489b64b25fa1a125752886fe93e9bd729897ca5ee050eb
Block #12 has been added to the blockchain!
Hash: fa01be73dee28f882151bdd91eacae2a7fe2b5c4fa2312a9cd84405bdeb87c2d
Block #13 has been added to the blockchain!
Hash: 2b95f80286009c6de5488b26d9b98729854b4dbb0f07f0a6eacac0b861f694a0
Block #14 has been added to the blockchain!
Hash: bcbf144a770814d236ff4cb3110be6a163cb7e72b075c55bd9e5d8c69fce7774
Block #15 has been added to the blockchain!
Hash: cade67d044678d91674f3d138c9c32860cef15ebe0f870576bb4060e7f5112c2
Block #16 has been added to the blockchain!
Hash: af52e05516a9dd81895d23da3703104fc8db5a96f2b43d638a2e91699fbb8cd5
Block #17 has been added to the blockchain!
Hash: f2705f37eb7ae871aa6510a687add404df6964e7425ec7332d24af7e75b7322d
Block #18 has been added to the blockchain!
Hash: 316c2a0addc6724af2271b09f5e8272a5d0a40dab205f02ac211c642495aec58
Block #19 has been added to the blockchain!
Hash: 1f428d2ff1c7d93f9d94f8561b0be4c8fc02b9d8b3099f60b27B472e7bc40faf
Block #20 has been added to the blockchain!
Hash: f3f00937c151bc45bff1a3025281cc24110a62a4286fb623efc04a207a844782
gabriel@gabriel-Lenovo-920-13IKB:~/iot/lesson10$


Activities
Terminal
Mon 11:04
gabriel@gabriel-Lenovo-920-13IKB: ~/iot/lesson10
File Edit View Search Terminal Help
gabriel@gabriel-Lenovo-920-13IKB:~/iot/lesson10$ cat snakecoin-server-full-code.py
# Gerald Nash, "Let's Make the Tiniest Blockchain Bigger Part 2: With More Lines of Python"
# Referred to https://www.pythonanywhere.com/forums/topic/12382/ that fixed sha.update() TypeError: Unicode-objects must be encoded before hashing
# Running on http://127.0.0.1:5000/mine (Reload the page to mine and press CTRL+C to quit)
from flask import Flask
from flask import request
import json
import requests
import hashlib as hasher
import datetime as date
from flask import send_from_directory
import os
node = Flask(__name__)
# Define what a Snakecoin block is
class Block:
def __init__(self, index, timestamp, data, previous_hash):
self.index = index
self.timestamp = timestamp
self.data = data
self.previous_hash = previous_hash
self.hash = self.hash_block()
def hash_block(self):
sha = hasher.sha256()
# sha.update(str(self.index) + str(self.timestamp) + str(self.data) + str(self.previous_hash))
sha.update((str(self.index) + str(self.timestamp) + str(self.data) + str(self.previous_hash)).encode("utf-8"))
return sha.hexdigest()
# Generate genesis block
def create_genesis_block():
# Manually construct a block with
# index zero and arbitrary previous hash
return Block(0, date.datetime.now(), {
"proof-of-work": 9,
"transactions": None
}, "0")
# A completely random address of the owner of this node
miner_address = "q3nf394hjg-random-miner-address-34nf3l4nflkn3oi"
# This node's blockchain copy
blockchain = []
blockchain.append(create_genesis_block())
# Store the transactions that
# this node has in a list
this_nodes_transactions = []
# Store the url data of every
# other node in the network
# so that we can communicate
# with them
peer_nodes = []


Activities
Terminal
Mon 11:10
gabriel@gabriel-Lenovo-920-13IKB: ~/iot/lesson10
File Edit View Search Terminal Help
curl: (7) Failed to connect to localhost port 5000: Connection refused
gabriel@gabriel-Lenovo-920-13IKB:~/iot/lesson10$ curl localhost:5000/mine
curl: (7) Failed to connect to localhost port 5000: Connection refused
gabriel@gabriel-Lenovo-920-13IKB:~/iot/lesson10$ python3 snakecoin-server-full-code.py
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
New transaction
[2022-04-18 11:09:29,564] ERROR in app: Exception on /txion [POST]
Traceback (most recent call last):
File "/usr/local/lib/python3.6/dist-packages/flask/app.py", line 1982, in wsgi_app
response = self.full_dispatch_request()
File "/usr/local/lib/python3.6/dist-packages/flask/app.py", line 1614, in full_dispatch_request
rv = self.handle_user_exception(e)
File "/usr/local/lib/python3.6/dist-packages/flask/app.py", line 1517, in handle_user_exception
reraise(exc_type, exc_value, tb)
File "/usr/local/lib/python3.6/dist-packages/flask/_compat.py", line 33, in reraise
raise value
File "/usr/local/lib/python3.6/dist-packages/flask/app.py", line 1612, in full_dispatch_request
rv = self.dispatch_request()
File "/usr/local/lib/python3.6/dist-packages/flask/app.py", line 1598, in dispatch_request
return self.view_functions[rule.endpoint](**req.view_args)
File "snakecoin-server-full-code.py", line 74, in transaction
print(("FROM: {}".format(new_txion['from'].encode('ascii','replace'))))
TypeError: tuple indices must be integers or slices, not str
127.0.0.1 - - [18/Apr/2022 11:09:29] "POST /txion HTTP/1.1" 500 -
[2022-04-18 11:09:39,991] ERROR in app: Exception on /mine [GET]
Traceback (most recent call last):
File "/usr/local/lib/python3.6/dist-packages/flask/app.py", line 1982, in wsgi_app
response = self.full_dispatch_request()
File "/usr/local/lib/python3.6/dist-packages/flask/app.py", line 1614, in full_dispatch_request
rv = self.handle_user_exception(e)
File "/usr/local/lib/python3.6/dist-packages/flask/app.py", line 1517, in handle_user_exception
reraise(exc_type, exc_value, tb)
File "/usr/local/lib/python3.6/dist-packages/flask/_compat.py", line 33, in reraise
raise value
File "/usr/local/lib/python3.6/dist-packages/flask/app.py", line 1612, in full_dispatch_request
rv = self.dispatch_request()
File "/usr/local/lib/python3.6/dist-packages/flask/app.py", line 1598, in dispatch_request
return self.view_functions[rule.endpoint](**req.view_args)
File "snakecoin-server-full-code.py", line 183, in mine
"hash": last_block_hash
File "/usr/lib/python3.6/json/__init__.py", line 231, in dumps
return _default_encoder.encode(obj)
File "/usr/lib/python3.6/json/encoder.py", line 199, in encode
chunks = self.iterencode(o, _one_shot=True)
File "/usr/lib/python3.6/json/encoder.py", line 257, in iterencode
return _iterencode(o, 0)
File "/usr/lib/python3.6/json/encoder.py", line 180, in default
o.__class__.__name__)
TypeError: Object of type 'ellipsis' is not JSON serializable
127.0.0.1 - - [18/Apr/2022 11:09:39] "GET /mine HTTP/1.1" 500 -
gabriel@gabriel-Lenovo-920-13IKB: ~
File Edit View Search Terminal Help
gabriel@gabriel-Lenovo-920-13IKB:~$ curl "localhost:5000/txion" \
> -H "Content-Type: application/json" \
> -d '{"from": "akjflw", "to":"fjlakdj", "amount": 3}'
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
<title>500 Internal Server Error</title>
<h1>Internal Server Error</h1>
The server encountered an internal error and was unable to complete your request. Either the server is overloaded or there is an error in the application.
gabriel@gabriel-Lenovo-920-13IKB:~$ curl localhost:5000/mine
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
<title>500 Internal Server Error</title>
<h1>Internal Server Error</h1>
The server encountered an internal error and was unable to complete your request. Either the server is overloaded or there is an error in the application.
gabriel@gabriel-Lenovo-920-13IKB:~$


Activities
Terminal
Mon 11:19
gabriel@gabriel-Lenovo-920-13IKB: ~/iot/lesson10/python_blockchain_app
File Edit View Search Terminal Help
[2022-04-18 11:09:39,991] ERROR in app: Exception on /mine [GET]
Traceback (most recent call last):
File "/usr/local/lib/python3.6/dist-packages/flask/app.py", line 1982, in wsgi_app
response = self.full_dispatch_request()
File "/usr/local/lib/python3.6/dist-packages/flask/app.py", line 1614, in full_dispatch_request
rv = self.handle_user_exception(e)
File "/usr/local/lib/python3.6/dist-packages/flask/app.py", line 1517, in handle_user_exception
reraise(exc_type, exc_value, tb)
File "/usr/local/lib/python3.6/dist-packages/flask/_compat.py", line 33, in reraise
raise value
File "/usr/local/lib/python3.6/dist-packages/flask/app.py", line 1612, in full_dispatch_request
rv = self.dispatch_request()
File "/usr/local/lib/python3.6/dist-packages/flask/app.py", line 1598, in dispatch_request
return self.view_functions[rule.endpoint](**req.view_args)
File "snakecoin-server-full-code.py", line 183, in mine
"hash": last_block_hash
File "/usr/lib/python3.6/json/__init__.py", line 231, in dumps
return _default_encoder.encode(obj)
File "/usr/lib/python3.6/json/encoder.py", line 199, in encode
chunks = self.iterencode(o, _one_shot=True)
File "/usr/lib/python3.6/json/encoder.py", line 257, in iterencode
return _iterencode(o, 0)
File "/usr/lib/python3.6/json/encoder.py", line 180, in default
o.__class__.__name__)
TypeError: Object of type 'ellipsis' is not JSON serializable
127.0.0.1 - - [18/Apr/2022 11:09:39] "GET /mine HTTP/1.1" 500 -
gabriel@gabriel-Lenovo-920-13IKB:~/iot/lesson10$ ^Ct clone https://github.com/satwikkansal/python_blockch
ain_app.git
gabriel@gabriel-Lenovo-920-13IKB:~/iot/lesson10$ git clone https://github.com/satwikkansal/python_blockch
ain_app.git
Cloning into 'python_blockchain_app'...
remote: Enumerating objects: 146, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 146 (delta 0), reused 1 (delta 0), pack-reused 143
Receiving objects: 100% (146/146), 223.96 KiB | 386.00 KiB/s, done.
Resolving deltas: 100% (67/67), done.
gabriel@gabriel-Lenovo-920-13IKB:~/iot/lesson10$ cd ~/python_blockchain_app
bash: cd: /home/gabriel/python_blockchain_app: No such file or directory
gabriel@gabriel-Lenovo-920-13IKB:~/iot/lesson10$ cd python_blockchain_app/
gabriel@gabriel-Lenovo-920-13IKB:~/iot/lesson10/python_blockchain_app$ python3 run_app.py
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
* Restarting with stat
* Debugger is active!
* Debugger PIN: 920-955-606
^Cgabriel@gabriel-Lenovo-920-13IKB:~/iot/lesson10/python_blockchain_app$ cim nodserver.py
Command 'cim' not found, but there are 20 similar ones.
gabriel@gabriel-Lenovo-920-13IKB:~/iot/lesson10/python_blockchain_app$ vim node_server.py
gabriel@gabriel-Lenovo-920-13IKB:~/iot/lesson10/python_blockchain_app$
gabriel@gabriel-Lenovo-920-13IKB: ~/iot/lesson10/python_blockchain_app
File Edit View Search Terminal Help
gabriel@gabriel-Lenovo-920-13IKB:~$ vncserver
Cleaning stale pidfile '/home/gabriel/.vnc/gabriel-Lenovo-920-13IKB:2.pid'!
New 'gabriel-Lenovo-920-13IKB:2 (gabriel)' desktop at :2 on machine gabriel-Lenovo-920-13IKB
Starting applications specified in /home/gabriel/.vnc/xstartup
Log file is /home/gabriel/.vnc/gabriel-Lenovo-920-13IKB:2.log
Use xtigervncviewer -SecurityTypes VncAuth -passwd /home/gabriel/.vnc/passwd :2 to connect to the VNC ser
ver.
gabriel@gabriel-Lenovo-920-13IKB:~$ cd iot/lesson10/python_blockchain_app/
gabriel@gabriel-Lenovo-920-13IKB:~/iot/lesson10/python_blockchain_app$ python2 run_app.py
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
* Restarting with stat
* Debugger is active!
* Debugger PIN: 338-431-234


Activities
Terminal
Mon 11:20
gabriel@gabriel-Lenovo-920-13IKB: ~/iot/lesson10
File Edit View Search Terminal Help
Collecting pyota[ccurl]
Downloading PyOTA-2.1.0-py2.py3-none-any.whl (113 kB)
| 113 kB 187 kB/s
Requirement already satisfied: requests[security]>=2.4.1 in /usr/local/lib/python3.6/dist-packages (from pyota[ccurl]) (2.27.1)
Requirement already satisfied: six in /usr/local/lib/python3.6/dist-packages (from pyota[ccurl]) (1.15.0)
Collecting pysha3
Downloading pysha3-1.0.2-cp36-cp36m-manylinux1_x86_64.whl (127 kB)
| 127 kB 351 kB/s
Collecting phx-filters
Downloading phx_filters-2.0.2-py3-none-any.whl (35 kB)
Collecting pyota-ccurl
Downloading PyOTA-CCurl-1.0.9.tar.gz (5.8 kB)
Preparing metadata (setup.py) ... done
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.6/dist-packages (from requests[security]>=2.4.1->pyota[ccurl]) (2021.10.8)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.6/dist-packages (from requests[security]>=2.4.1->pyota[ccurl]) (3.3)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /usr/local/lib/python3.6/dist-packages (from requests[security]>=2.4.1->pyota[ccurl]) (1.26.8)
Requirement already satisfied: charset-normalizer~=2.0.0 in /usr/local/lib/python3.6/dist-packages (from requests[security]>=2.4.1->pyota[ccurl]) (2.0.12)
Collecting regex>=2018.8.17
Downloading regex-2022.3.15-cp36-cp36m-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (749 kB)
| 749 kB 189 kB/s
Collecting phx-class-registry
Downloading phx_class_registry-3.0.5-py3-none-any.whl (11 kB)
Requirement already satisfied: pytz in /home/gabriel/.local/lib/python3.6/site-packages (from phx-filters->pyota[ccurl]) (2021.3)
Requirement already satisfied: python-dateutil in /usr/local/lib/python3.6/dist-packages (from phx-filters->pyota[ccurl]) (2.8.2)
Building wheels for collected packages: pyota-ccurl
Building wheel for pyota-ccurl (setup.py) ... done
Created wheel for pyota-ccurl: filename=PyOTA_CCurl-1.0.9-cp36-cp36m-linux_x86_64.whl size=19744 sha256=8b7d925981ecf7e780e97796f8a0005043ae31037491b15a689696ce40724dc1
Stored in directory: /tmp/pip-ephem-wheel-cache-e9qfgz3u/wheels/d0/a9/0b/a29c6461dd61261f8caf98556e508f3da7e1919323305dcd49
Successfully built pyota-ccurl
Installing collected packages: regex, phx-class-registry, pysha3, phx-filters, pyota, pyota-ccurl
Successfully installed phx-class-registry-3.0.5 phx-filters-2.0.2 pyota-2.1.0 pyota-ccurl-1.0.9 pysha3-1.0.2 regex-2022.3.15
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the system package manager. It is recommended to use a virtual environment instead: https://pip.pypa.io/war
nings/venv
gabriel@gabriel-Lenovo-920-13IKB:~/iot/lesson10/python_blockchain_app$ cd ~/iot/lesson10
gabriel@gabriel-Lenovo-920-13IKB:~/iot/lesson10$ cat iri_node_info.py
from iota import Iota
# Create a new instance of the IOTA API object
# Specify which node to connect to
api = Iota(adapter = 'https://nodes.devnet.iota.org:443')
# Call the `get_node_info()` method for information about the node and the Tangle
response = api.get_node_info()
print(response)
gabriel@gabriel-Lenovo-920-13IKB:~/iot/lesson10$ python3 iri_node_info.py
{'latestMilestone': TransactionHash(b'WL9XOAZZKHEGZBIBKBSIDI9JZRRCY9GDOKHOEKPJGNHNABSMEZRVWCHAQDPTHHKNG9JCLMXBGYTYKN999'), 'latestSolidSubtangleMilestone': TransactionHash(b'WL9XOAZZKHEGZBIBKBSIDI9JZRRCY9GDOKHOE
KPJGNHNABSMEZRVWCHAQDPTHHKNG9JCLMXBGYTYKN999'), 'appName': 'HORNET', 'appVersion': '0.5.6', 'coordinatorAddress': 'GYISMBVRKSCEXXTUPBWTIHRCZIKIRPDYAHAYKMNTPZSCSDNADDWAEUNHKUERZCTVAYJCNFXGTNUH9OGTW', 'duration':
0, 'features': ['RemotePOW', 'WereAddressesSpentFrom'], 'isHealthy': True, 'isSynced': True, 'lastSnapshottedMilestoneIndex': 3538535, 'latestMilestoneIndex': 3538590, 'latestSolidSubtangleMilestoneIndex': 35385
90, 'milestoneStartIndex': 3209121, 'neighbors': 3, 'time': 1650295252000, 'tips': 8, 'transactionsToRequest': 0}
gabriel@gabriel-Lenovo-920-13IKB:~/iot/lesson10$