```
user@box:~/iot$ cd ~/iot/lesson10
user@box:~/iot/lesson10$ cat hash_value.py
"""
https://docs.python.org/3/using/cmdline.html#envvar-PYTHONHASHSEED
If PYTHONHASHSEED is not set or set to random, a random value is used to to seed the hashes of str and bytes objects.
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 randomizatio
n.

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 an
d 1.0.
For objects with custom __hash__() methods, note that hash() truncates the return value based on the bit width of the h
ost 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))
user@box:~/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: 1808980586519270115
The hash for a tuple of vowels is: 4254621767398073324
The hash for an object of person is: -6889168760377247615
user@box:~/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: -6792000764179658805
The hash for a tuple of vowels is: 5986796428488884241
The hash for an object of person is: 768115832896944849
```

```
user@box:~/iot/lesson10$ python3
Python 3.8.10 (default, Nov 26 2021, 20:14:08)
[GCC 9.3.0] 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()
user@box:~/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_ha
sh).encode())
```

```
sh).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!
```

```
user@box:~/iot/lesson10$ python3 snakecoin.py
Block #1 has been added to the blockchain!
Hash: ae33685286b7fb265921103ec41d863f80ebc3b34c33a9d6452ca93e1e918165

Block #2 has been added to the blockchain!
Hash: a753591fd6ff446b084672defef96029f45e33a1d77be9dd8ca99c978359ad3e

Block #3 has been added to the blockchain!
Hash: 9876bdb0201bb70e69297a2d48f5f9f1941c1bc214c8723a990fb33709a00761

Block #4 has been added to the blockchain!
Hash: f7f7a7b1ef5bce4abb81d8a0bbf6a116284e0d093a252072861376c35b99eb29

Block #5 has been added to the blockchain!
Hash: 9cae48da066bb1c46f71b8ac458de1b1ba66e6bd26d5f05609c883ae3c2442d9

Block #6 has been added to the blockchain!
Hash: d514d3ed70971457b4f899dd6e2ceef73c2d37be818831090e89bfef052305a7

Block #7 has been added to the blockchain!
Hash: 483b7cf8c4a25046c3ea9d9ea7a2e8b07912089ef35640ccc0b556f3b7daa8a8

Block #8 has been added to the blockchain!
Hash: df9a138ceae49b41887114482facfb472d7f57b139fbf3eb46800f76805310d4

Block #9 has been added to the blockchain!
Hash: 3eca9ff51915bc1300c13a45f10811bbaf737ced30a654f157a25202f5a97f2f

Block #10 has been added to the blockchain!
Hash: 06d683e93a2e92f0960c81e7347f57b0dc2971295316ab45e5fa1b9f2401a7ac
```

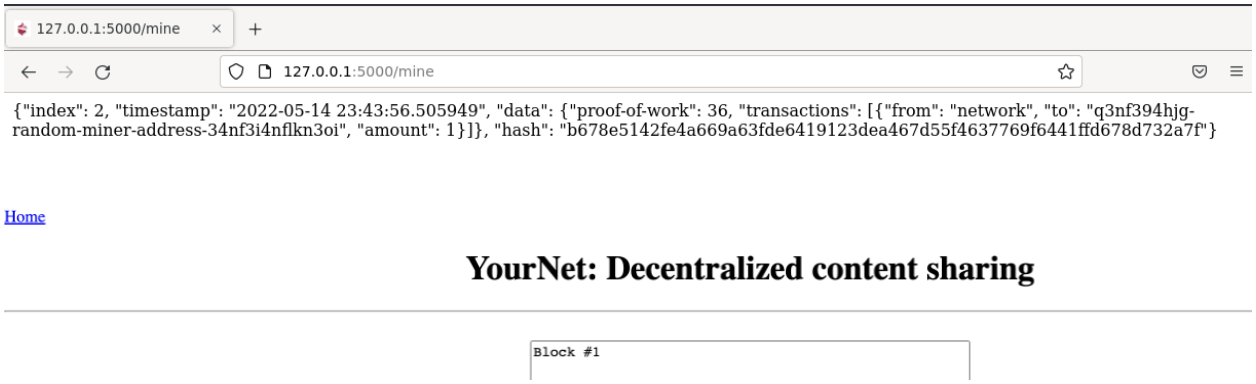
```
Block #11 has been added to the blockchain!
Hash: 01cd4e7851fd22a8296f8ce8a728413187ac00fb1de9bf22d38cc548b322482a

Block #12 has been added to the blockchain!
Hash: 13b77ba02ca9512e56707040cf6f181cc8f8d7eeae0a273d092df4327a3e1364

Block #13 has been added to the blockchain!
Hash: 89a7dd538481e6e4823e40b1ce1a3e898ca2bf7f7dec749da028b5c977eab786

Block #14 has been added to the blockchain!
Hash: ecbcf35b216fa81e6405c4e22082d7ea55b7c4e890dc6e6038e060f554e24ea0

Block #15 has been added to the blockchain!
Hash: 40082b0c7bd5cd04525b8282cebf41021ca621146cc18b3cedc2ba82369d373d

Block #16 has been added to the blockchain!
Hash: a509e08f6da6b484ab99ec6c98173b064e9bedd1b3948e9c8862f4ff248b2905

Block #17 has been added to the blockchain!
Hash: d4a4689967792453e96c19425bd9994b35afd08ba7a37aa1ae040d97bd56d8dd

Block #18 has been added to the blockchain!
Hash: 5f22194e1d0f86859e797409def58215af677710fd7761dc08fee288780d90bd

Block #19 has been added to the blockchain!
Hash: 4a924182313ead789736eedf261b324a920953b6db261aadd083e1c2fa3be066

Block #20 has been added to the blockchain!
Hash: 6a273be21b87ab2e5d92c31bc7fbff43e9a0c8e5b84a31cd4dd7e64a95becab6
```

```
user@box:~/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 mus
t 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
```

```
user@box:~/iot/lesson10$ python3 snakecoin-server-full-code.py
 * Serving Flask app 'snakecoin-server-full-code' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
New transaction
FROM: b'akjflw'
TO: b'fjlakdj'
AMOUNT: 3

127.0.0.1 - - [14/May/2022 23:41:52] "POST /txion HTTP/1.1" 200 -
127.0.0.1 - - [14/May/2022 23:41:58] "GET /mine HTTP/1.1" 200 -
127.0.0.1 - - [14/May/2022 23:42:38] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [14/May/2022 23:42:38] "GET /favicon.ico HTTP/1.1" 200 -
```

```
user@box:~$ curl "localhost:5000/txion" \
>      -H "Content-Type: application/json" \
>      -d '{"from": "akjflw", "to":"fjlakdj", "amount": 3}'
Transaction submission successful
user@box:~$ curl localhost:5000/mine
{"index": 1, "timestamp": "2022-05-14 23:41:58.607135", "data": {"proof-of-work": 18, "transactions": [{"from": "akjflw
", "to": "fjlakdj", "amount": 3}, {"from": "network", "to": "q3nf394hjg-random-miner-address-34nf3i4nflkn3oi", "amount"
: 1}]}, "hash": "11ea2aa15c89e898cce07b199eddd7ae693f0baac4975119c870893d99a6f0f4"}
```

SnakeCoin Server


127.0.0.1:5000/mine
127.0.0.1:5000/mine
{"index": 2, "timestamp": "2022-05-14 23:43:56.505949", "data": {"proof-of-work": 36, "transactions": [{"from": "network", "to": "q3nf394hjg-random-miner-address-34nf3i4nflkn3oi", "amount": 1}]}, "hash": "b678e5142fe4a669a63fde6419123dea467d55f4637769f6441ffd678d732a7f"}
Home
YourNet: Decentralized content sharing
Block #1