Goal of this lecture note

- Blockchain Core
- Program Package
- Python Blockchain Core

1 Blockchain Core

- We aim to study what a blockchain core is.
  - Cryptocurrency is built on a program suite.
  - The suite is to form and maintain a ledger in a P2P computer network.
  - The best way to learn is to develop it from scratch.
  - We need to install SW package and do a little bit of coding.

1 Blockchain Core

- A simple Python core is written.
  - This code controls a node.
  - It can have nodes interact with each other.
  - A group of such nodes can support a cryptocurrency system.

1 Blockchain Core

- List of things we aim to do:
  - Run the core at a group of nodes,
  - Have nodes register their neighbors,
  - Have nodes generate new transactions,
  - Have nodes mine new blocks,
  - Have nodes reach consensus, and show
- This network can maintain a blockchain.

1 Blockchain Core

- Define node discovery routines:
  - Be aware of neighbors
  - Give my list of addresses upon requests
  - Listen to chains and transactions announcements and get them from neighbors

1 Blockchain Core

- Define what a block is:
  - BH: Previous Hash, Merkle Root Hash, Timestamp, Nonce, Version, Difficulty
  - BB: Transactions, Tree Structure
- Make the genesis block.

1 Blockchain Core

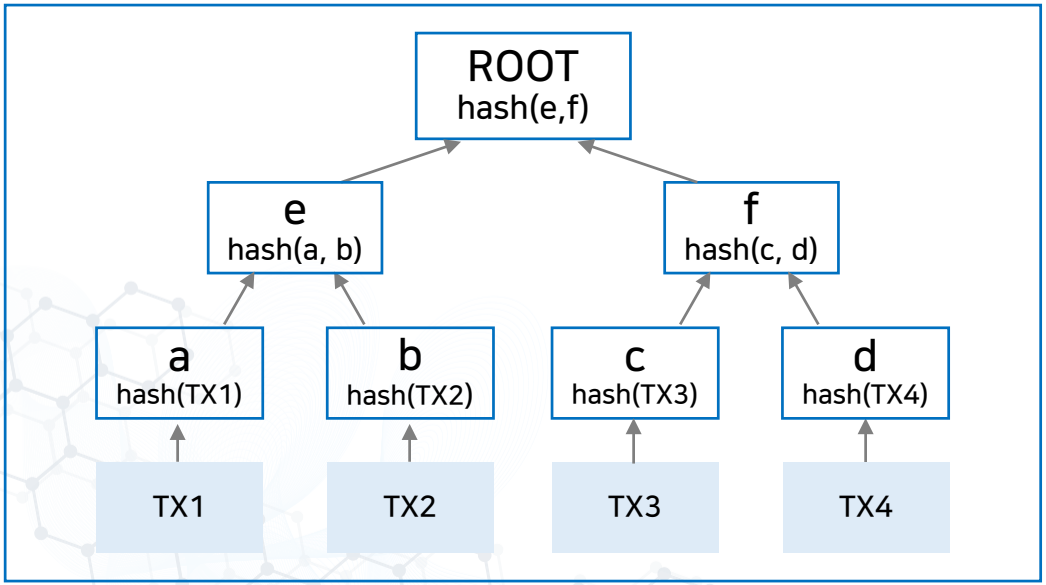
- Define transaction generation routines
  - Generate keys and addresses.
  - Make a new transaction:
    - Find UTXOs
    - Get destination addresses
    - Make a locking script per each address
    - Make TXID
  - Announce TX.
  - Track the TXs issued until fully confirmed.
    - check to see if TXs are included in the main chain.
    - re-issue those TXs not included in the main chain.

1 Blockchain Core

- Define a transaction verification routine
  - Get a TX and validate it.
  - Input UTXOs with locking scripts.
  - See if each sign unlocks the lock.
  - Check output values in the locking scripts.
  - See if the balance is enough.
  - Verify TXID = hash(inputs, outputs).

1 Blockchain Core

- Define Merkle root hash routine
  - Binary hash tree of TXIDs



1 Blockchain Core

- Define a block verification routine.
  - Verify each transaction.
  - Verify the Merkle root hash.
  - Verify the hash of the BH.
    - Put the prev. block header into SHA-256 and see if it satisfies the difficulty level.
    - The difficulty level cannot be forged since it is included in the block header.

1 Blockchain Core

- Define difficulty change routine.
  - Change `Target` periodically.

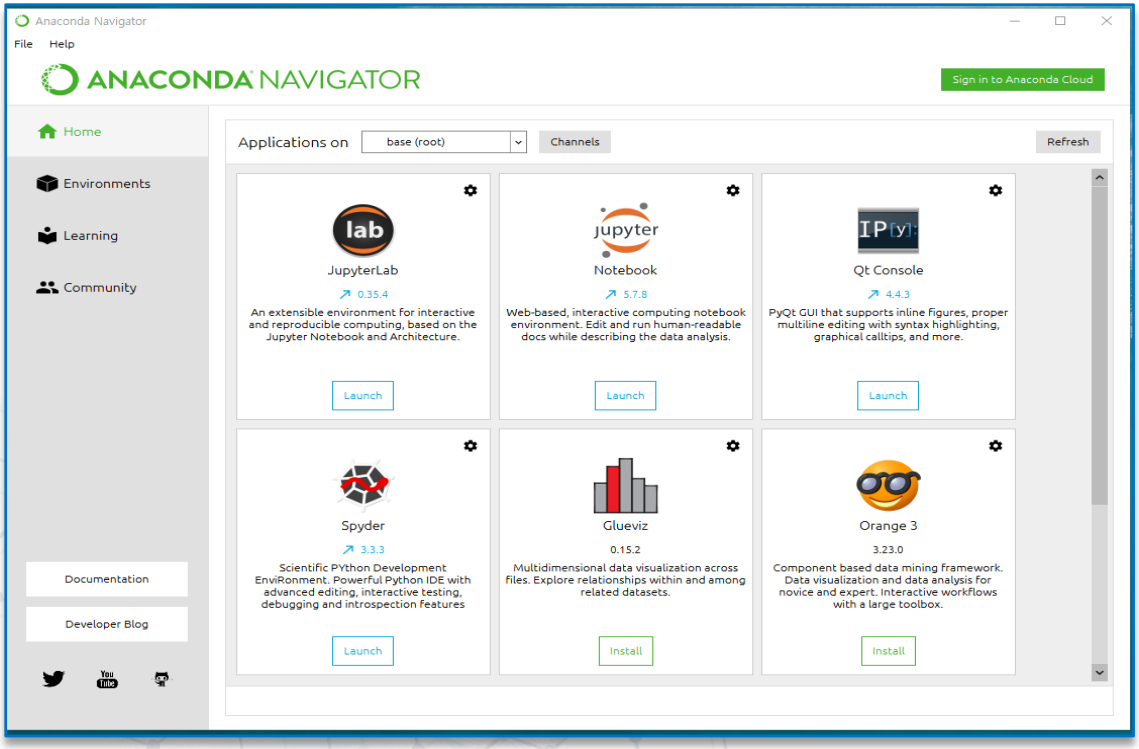
1 Blockchain Core

- Define a mining routine.
  - Collect announced transactions.
  - Get the longest chain from the neighbors.
  - Validate the imported chain.
    - Verify the blocks.
    - Verify the sequence of proofs.
  - Form a new block by finding a good nonce.
  - Announce the new chain ASAP.

2 Program Package

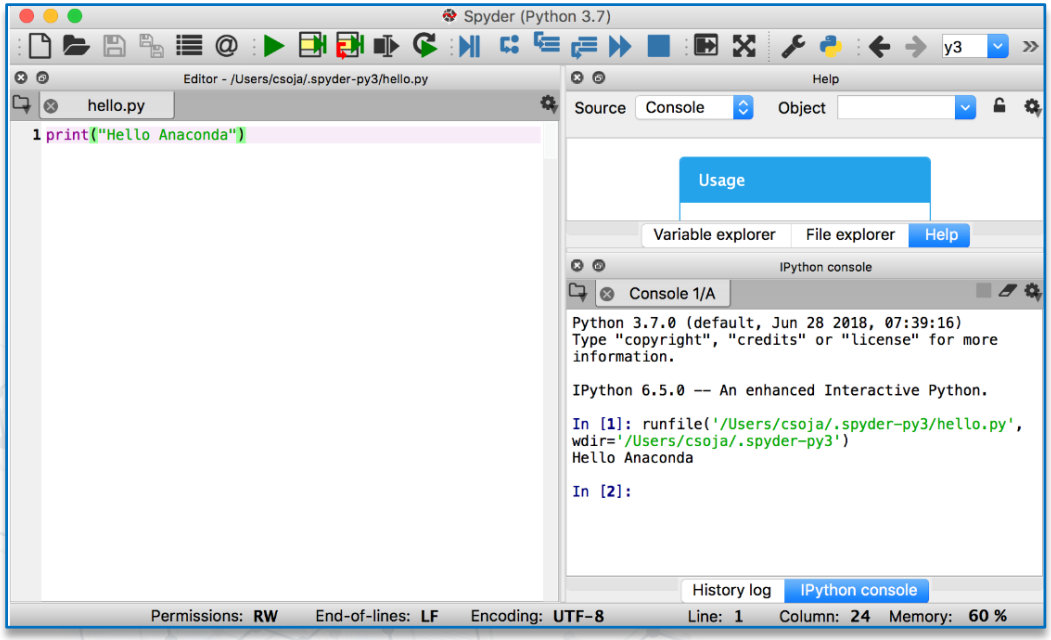
- *Anaconda*
  - https://www.anaconda.com/distribution/
  - Free OS *Python*
  - *Spyder*
    - *Write python code and run*
- *FLASK*
  - Use it to write an API in *Python*
- *Postman*
  - Use it to test APIs.
  - https://www.getpostman.com/downloads/

2 Program Package



2 Program Package

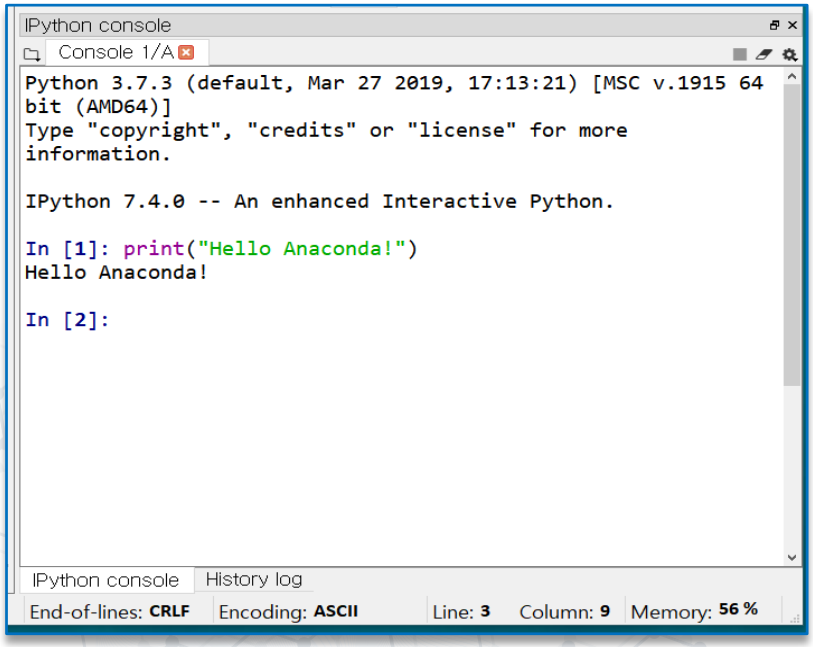
- Edit and run Python at Spyder IDE



출처: https://docs.anaconda.com/anaconda/user-guide/getting-started/

2 Program Package

- Open an Anaconda terminal and run a Python program.



2 Program Package

- FLASK
  - Flask is a micro web development tool written in Python.
  - The following code shows a simple web application that prints "Hello World!":

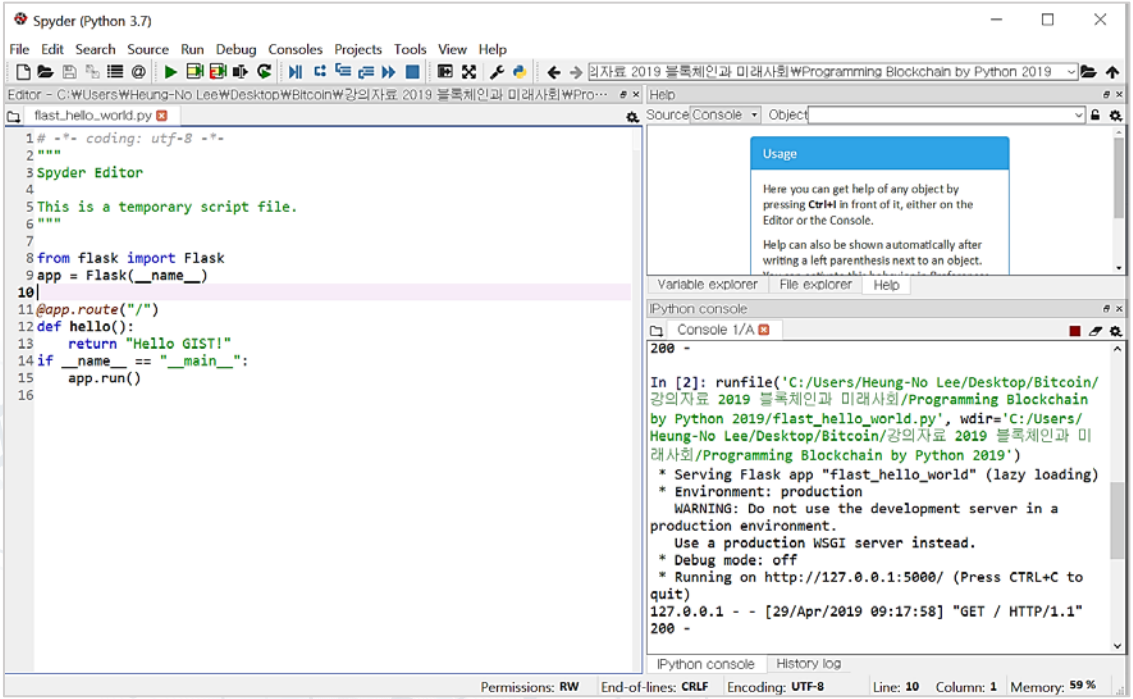
```
from flask import Flask
app = Flask(__name__)

@app.route("/")
def hello():
        return "Hello World!“

if __name__ == "__main__":
      app.run()
```

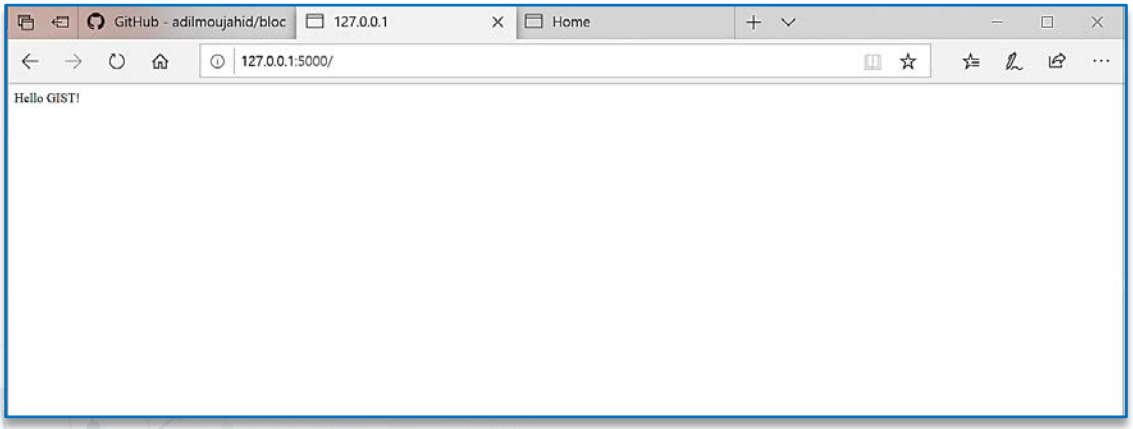
2 Program Package

- Write the First FLASK code



2 Program Package

- Confirmation



3 Python Blockchain Core

- Download the Python blockchain files.
  - https://github.com/infonetGIST/Blockchain_lecture
- There are three kinds of Python files:
  - blockchain.py,
  - miner1.py, miner2.py, miner3.py
  - trader.py

3 Python Blockchain Core

- Python code for a Blockchain
  - Open up `blockchain.py` file in Spyder
  - It defines the blockchain class under which all core routines are defined.
  - It is only 531 lines long (17 def's and 9 app's)

Chaining	Mining
Resolving the chain by its length	Posting TXs (limited to itself)

3 Python Blockchain Core

- Blockchain.py

Import libraries

Declare Blockchain class

Define Flask app's

```
Import

class Blockchain:

# Instantiate the Node
app = Flask(__name__)

# Instantiate the Blockchain
blockchain = Blockchain()
```

3 Python Blockchain Core

- Import libraries

```
from threading import Thread, Event
import time
from flask import Flask, jsonify, request
import requests
import hashlib
import json
from urllib.parse import urlparse
from uuid import uuid4
import random
```

3 Python Blockchain Core

- class blockchain has 17 definitions

```
def __init__(self):
def register_node
def valid_chain
def resolve_conflicts
def new_block
def new_transaction
def update_transactions
def is_valid_TX
def check_current_TXs_validity
def update_awaiting_TX
def make_published_TXID_list
```

3 Python Blockchain Core

- class blockchain has 17 definitions

```
def announcement
def mine
def last_block
def hash
def proof_of_work
def valid_proof
```

3 Python Blockchain Core

```
66 class Blockchain:
67     def __init__(self):
68         self.current_transactions = []
69         self.awaiting_transactions =[]
70         self.chain = []
71         self.nodes = set()
72         self.published_transactions_ID = []
73         self.mining_reward_address='0'
74         self.MY_NODE_ADDRESS='0'
75         # Generate a globally unique address for this node
76         self.node_identifier = str(uuid4()).replace('-', '')
77         # node_identifier = hex(random.randrange(1, 9999999))
78         self.interrupt_flag=False
79 
80         dummy_block = {
81             'index': 0,
82             'timestamp': 0,
83             'transactions': [],
84             'proof': 0,
85             'previous_hash': 0,
86         }
87         Ini_proof = self.proof_of_work(mining_time=0, last_block=dummy_block)
88         self.new_block(previous_hash='0', mining_time=0, proof=Ini_proof)
```

3 Python Blockchain Core

- Take a look at some def's under blockchain class

```
90      def register_node(self, address):
91          """
92          Add a new node to the list of nodes
93
94          :param address: Address of node. Eg. 'http://192.168.0.5:5000'
95          """
96
97          parsed_url = urlparse(address)
98          if parsed_url.netloc:
99              self.nodes.add(parsed_url.netloc)
100         elif parsed_url.path:
101             # Accepts an URL without scheme like '192.168.0.5:5000'.
102             self.nodes.add(parsed_url.path)
103         else:
104             raise ValueError('Invalid URL')
```

3 Python Blockchain Core

```
173     def new_block(self, mining_time, proof, previous_hash):
174         """
175         Create a new Block in the Blockchain
176 
177         :param proof: The proof given by the Proof of Work algorithm
178         :param previous_hash: Hash of previous Block
179         :return: New Block
180         """
181 
182         block = {
183             'index': len(self.chain) + 1,
184             'timestamp': mining_time,
185             'transactions': self.current_transactions,
186             'proof': proof,
187             'previous_hash': previous_hash or self.hash(self.chain[-1]),
188         }
189 
190         # Reset the current list of transactions
191         self.current_transactions = []
192 
193         self.chain.append(block)
194         self.make_published_TXID_list()
195         return block
```

3 Python Blockchain Core

```
309     def mine(self):
310         # We run the proof of work algorithm to get the next proof...
311         last_block = self.last_block
312         mining_time = time.time(),
313         randomSTR = str(uuid4()).replace('-', '')
314         self.new_transaction(
315             sender="Coinbase transaction",
316             recipient=self.mining_reward_address + '  #' + randomSTR,
317             amount=1,
318         )
319         proof = self.proof_of_work(mining_time, last_block)
320 
321         # We must receive a reward for finding the proof.
322         # The sender is "0" to signify that this node has mined a new coin.
323 
324         # Forge the new Block by adding it to the chain
325         if proof==0:
326             del self.current_transactions[-1]
327         else:
328             previous_hash = self.hash(last_block)
329             block = self.new_block(mining_time, proof, previous_hash)
330             print("Mining success!")
331             self.announcement()
332 
```

3 Python Blockchain Core

```
def proof_of_work(self, mining_time, last_block):

    last_proof = last_block['proof']
    if len(self.chain) == 0:
        last_hash = '0'
    else:
        last_hash = self.hash(last_block)

    proof = 0
    test_block = {
        'index': len(self.chain) + 1,
        'timestamp': mining_time,
        'transactions': self.current_transactions,
        'proof': proof,
        'previous_hash': last_hash or self.hash(self.chain[-1]),
    }
    while self.valid_proof(test_block) is False:
        if self.interrupt_flag:
            blockchain.interrupt_flag = False
            return 0

        proof += 1
        test_block = {
            'index': len(self.chain) + 1,
            'timestamp': mining_time,
            'transactions': self.current_transactions,
            'proof': proof,
            'previous_hash': last_hash or self.hash(self.chain[-1]),
        }

    return proof
```

3 Python Blockchain Core

- Flask app's
  - GETs and POSTs
    - Mine a block
    - Post a TX
    - Make a chain
    - Get transactions
    - Get chain updates
    - Register a node
    - Make consensus
    - Shut down

```
@app.route('/mine', methods=['GET'])
def mine():

@app.route('/transactions/new', methods=['POST'])
def new_transaction():

@app.route('/chain', methods=['GET'])
def full_chain():

@app.route('/get_transactions', methods=['GET'])
def full_transactions():

@app.route('/get_awaiting_transactions')
def awaiting_transactions():
```

3 Python Blockchain Core

- Flask app's
  - GETs and POSTs
    - Mine a block
    - Post a TX
    - Make a chain
    - Get transactions
    - Get chain updates
    - Register a node
    - Make consensus
    - Shut down

```
@app.route('/get_updates')
def receiving_longest_chain_and_update_TX_list():

@app.route('/nodes/register', methods=['POST'])
def register_nodes():

@app.route('/nodes/resolve', methods=['GET'])
def consensus():

@app.route("/shutdown")
def shutdown()
```