

# < 블록체인 개요 >

2018. 11. 07(수)

1. 개발자의 stackoverflow => 일 하는 프로그래머(즐거는 사람), 평균적인 프로그래머(회사원),
2. 언어의 종류? Swipt, RUBY, Typescript(->확장? [앵귤러](#)), C, C++, C#, PHP, python, bash/shell, (15위 → 1위 순) JAVA, SQL(블록체인은 DB의 확장판), CSS, HTML, JavaScript
3. 세계에서 blockchain을 인정해주고 있는 3개 집단? 미국, 일본, 유럽
4. bitcoin에 만원 정도 투자해 볼 것을 권함 -> 업비트(작년 11월 나옴), 빗썸, 코빗
5. 가장 돈 많이 받는 개발자가 사용하는 언어 => GO언어(Typescript, JavaScript를 발전시킴)
6. *gartner 선정 top 10 strategic technology trends for 2019 조사 후 이메일 보내기 과제!!*
  - 18년 선정주제(인공지능, 클라우드, 블록체인 등등등)
  - 클라우드가 굉장히 발전했지만 아직까지 참조할만한 reference나 전문가가 없음  
(AWS 등 나와있는 많은 것들을 한 번 공부해볼 것을 권유함)
7. visual studio code 파일 다운로드(기반 자체가 Linux의 vi에디터)
  - 맨 밑에 Extension 가서 라이브러리 몇 개 다운로드 필요!
    - ① python
    - ② vscode-icon : File – Preference – Icon Theme에서 기본값을 바꿔주기
    - ③ Django
8. git bash -> powershell을 git bash로 변경해야함(visual studio code에서) : 경로변경
9. postman이라는 chrome app 하나 추가 설치 -> 이거하면 put방식으로 데이터 처리가능!!
10. block data로 저장할 수 있는 것들이 뭐가 있을까 고민 필요!(entity 설정 각 조별로 실시)
11. 터미널창에 'pip list'해서 원하는게 다 설치되어있나 확인
  - Flask와 requests(없으면 'pip install Flask requests'로 설치)

## ★ JSON형식

- > JavaScript Object Notation의 약자로 자바스크립트에서 만들어진 대표적 표현방식(표기법)
- > 사용을 위해서는 반드시 import를 통해 모듈을 가져와야 함.
- > JSON 문자열을 만들기 위해서는 dumps 함수를 사용!
- > JSON 문자열을 다시 딕셔너리로 변환하기 위해서는 loads 함수를 사용!
- > JSON 형태의 파일을 읽어드리는 함수는 load를 사용!

## ★ @staticmethod vs @classmethod (정적메소드)

- > instance 없이도 클래스에 직접 접근할 수 있는 메소드(python에서는 instance에서도 접근가능)
- > 상속에서 차이가 있음(static은 불가능, class는 가능!)
- > 블록체인 코드에서 Blockchain은 class / new\_block, new\_transaction 등의 함수는 method / 밖에서 class를 지정할 객체가 instance

## ★ @property

- > 파이썬은 private, protected가 없이 전부 public을 사용하기 때문에 클래스 내부 변수에 접근 할 때 문제 발생 가능성 있음(캡슐화, 코드복잡 등)
- > @property를 사용하면 약간의 private 느낌나도록 사용 가능!  
(변수를 변경할 때 제한 가능, get/set/함수 만들지 않고 더 간단히 접근가능, 하위호환성 도움 등)

# < 블록체인 구현 >

2018. 11. 08(목)

1. 2016개의 block이 각 10분 정도 걸려 생성  
= 14일에 만들어지는 개수(10분\*6\*24시간\*14일)

2. Transaction Data에 들어갈 수 있는 크기는 블록체인 기준 약 944byte(1MB - 80byte)  
생각보다 크지 않기 때문에 불필요한 데이터가 들어갈 필요는 없음  
status와 관련된 정보만 포함시키는 것이 좋음(거래기준? 송신자, 수신자, 금액)

3. 합의 알고리즘의 종류? PoW, PoS, PoA 등

4. 비트코인의 주소? 개인키 → 공개키 → 주소로 생성됨(Tx data에 들어가는 정보)  
(cf - blockchain.info)

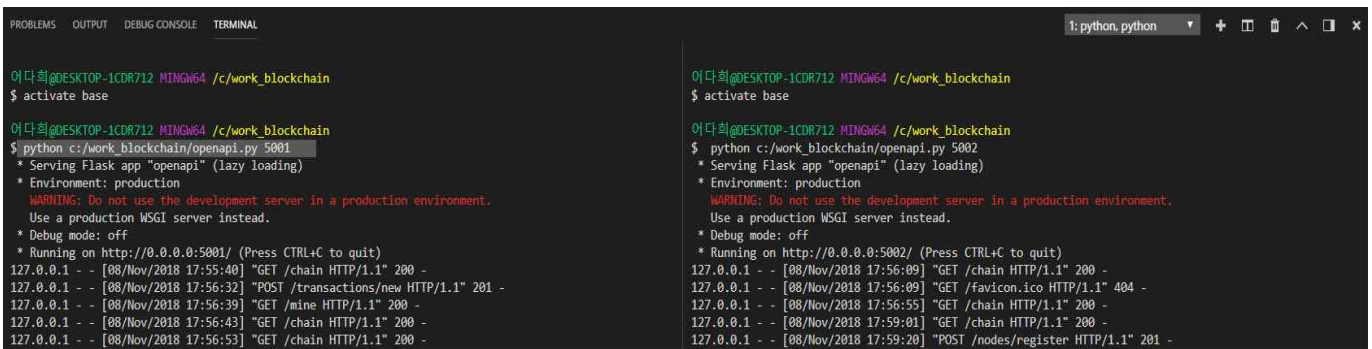
5. 매개변수(Parameter) vs 전달인자(Argument)  
말 그대로 변수                      값을 의미

## 6. API → OpenAPI → HTTP CRUD

- /transaction/new -> new\_transaction()  
: 경로를 만들어서 함수를 불러올 것임(parameter : sender, recipient, amount 사용)
- /mine -> 서버에 새로운 블록 정보를 갱신 요청(채굴)
- /chain -> 전체블록체인의 정보를 확인

-----현재 나 혼자만 쓸 수 있으니 node들을 더 만들어 보자

- /nodes/register -> 노드 추가
  - /nodes/resolve -> 정확한 블록 설정
- ※ 터미널 창 2개를 띄워서 port번호 2개를 생성 후 run server



```
어다희@DESKTOP-1CD712 MINGW64 /c/work_blockchain
$ activate base

어다희@DESKTOP-1CD712 MINGW64 /c/work_blockchain
$ python c:/work_blockchain/openapi.py 5001
* Serving Flask app "openapi" (lazy loading)
* Environment: production
WARNING: Do not use the development server in a production environment.
Use a production WSGI server instead.
* Debug mode: off
* Running on http://0.0.0.0:5001/ (Press CTRL+C to quit)
127.0.0.1 - - [08/Nov/2018 17:55:48] "GET /chain HTTP/1.1" 200 -
127.0.0.1 - - [08/Nov/2018 17:56:32] "POST /transactions/new HTTP/1.1" 201 -
127.0.0.1 - - [08/Nov/2018 17:56:39] "GET /mine HTTP/1.1" 200 -
127.0.0.1 - - [08/Nov/2018 17:56:43] "GET /chain HTTP/1.1" 200 -
127.0.0.1 - - [08/Nov/2018 17:56:53] "GET /chain HTTP/1.1" 200 -

어다희@DESKTOP-1CD712 MINGW64 /c/work_blockchain
$ activate base

어다희@DESKTOP-1CD712 MINGW64 /c/work_blockchain
$ python c:/work_blockchain/openapi.py 5002
* Serving Flask app "openapi" (lazy loading)
* Environment: production
WARNING: Do not use the development server in a production environment.
Use a production WSGI server instead.
* Debug mode: off
* Running on http://0.0.0.0:5002/ (Press CTRL+C to quit)
127.0.0.1 - - [08/Nov/2018 17:56:09] "GET /chain HTTP/1.1" 200 -
127.0.0.1 - - [08/Nov/2018 17:56:09] "GET /favicon.ico HTTP/1.1" 404 -
127.0.0.1 - - [08/Nov/2018 17:56:55] "GET /chain HTTP/1.1" 200 -
127.0.0.1 - - [08/Nov/2018 17:59:01] "GET /chain HTTP/1.1" 200 -
127.0.0.1 - - [08/Nov/2018 17:59:20] "POST /nodes/register HTTP/1.1" 201 -
```

각 port별로 최초 /chain에 genesis블록만 있는 것을 확인 → 5001포트의 /transaction/new에 들어가서(포스트맨 이용) 거래데이터 전송(POST) → 5001포트의 /mine하여 채굴(블록생성) → 5001포트의 /chain에 가서 블록이 추가되었는지 확인 → 5001의 /nodes/register에 가서(포스트맨) 5002포트의 노드 추가 → 5002의 /nodes/register에 가서(포스트맨) 5001포트의 노드 추가 → 5002의 /nodes/resolve(동기화)를 해보면 'Our chain was replaced.'라는 멘트 뜸(5001의 블록이 더 길고 유효하기 때문), 5001의 /nodes/resolve를 해보면 'Our chain is authoritative.'라고 뜸(5002와 비교해서 제일 길고 유효하므로) → 5002의 /chain으로 확인 (데이터를 입력하지 않았음에도 node가 연결되면서 5001의 거래내역이 5002에 담김 -합의알고리즘)  
※ 현재 내 PC에서 노드 2개만을 만들어서 해봤지만, 옆 컴퓨터와 실습도 가능함  
이런 식으로 전 세계의 node들이 거래정보를 공유하고 기록!!

7. 장고를 사용하지 않고 간단히 Flask를 이용해 웹서버를 띄우는 방법  
- @app.route를 사용

8. openapi.py 파일에서 /chain 주소를 호출하면 나오는 결과값(json형식)



```
{
  "blockchain": [
    {
      "index": 1,
      "previous_hash": 1,
      "proof": 100,
      "timestamp": 1541651168.3136106,
      "transactions": []
    }
  ],
  "length": 1
}
```

★ parse vs compile  
 분석하다(문법적 해부) 번역하다(기계어 번역)  
 urlparse 같은 경우 해당 주소를 받아와서 사용 할 준비 => 네트워크의 위치를 추가할 수 있도록

```
def register_node(self, address) :
    ...
    노드리스트에 새로운 노드를 하나 추가
    param address : http://127.0.0.1:5000 이 전달됨
    ...

    parse_url = urlparse(address) # 해당 라이브러리 import 해주기
    self.nodes.add(parse_url.netloc)
```

## 1. urlparse 모듈

이 모듈은 URL의 분해, 조립, 변경 등을 처리하는 함수를 제공하며, parse한 결과를 리턴한다.

```
from urlparse import urlparse
result = urlparse("http://www.python.org:80:80/guido/python.html:philosophy?overall=3#n10")
result
ParseResult(scheme='http', netloc='www.python.org:80:80', path='/guido/python.html',
param s='philosophy', query='overall=3', fragment='n10')
```

### \* 속성값의 의미

속성 값	의미
scheme	URL에 사용된 프로토콜을 의미
netloc	네트워크의 위치, <b>user:password@host:port</b> 형식으로 표현되며, HTTP 프로토콜일 경우 <b>host:port</b> 형식으로 지정된다.
path	파일이나 애플리케이션 경로를 의미
params	애플리케이션에 전달될 매개변수
query	질의 문자열로 앰퍼샌드(&)로 구분된 <b>키=값</b> 쌍 형식으로 표현
fragment	문서내의 앵커 등 조각을 지정

★ Flask는 파이썬으로 작성된 마이크로 웹 프레임워크의 하나(아주 작고 가벼우며 빠르게 동작!) 동적인 웹 페이지나, 웹 애플리케이션, 웹 서비스 개발 보조용으로 만들어지는 애플리케이션 프레임워크의 일종이다.

웹 페이지를 개발하는 과정에서 겪는 어려움을 줄이는 것이 주 목적으로 통상 데이터베이스 연동, 템플릿 형태의 표준, 세션 관리, 코드 재사용 등의 기능을 포함

플라스크



개발자 **Armin Ronacher**  
 발표일 2010년 4월 1일 (8년 전)  
 최근 버전 0.12.2 / 2017년 5월 16일 (16달 전)<sup>[1]</sup>  
 프로그래밍 언어 파이썬  
 운영 체제 크로스 플랫폼  
 종류 웹 프레임워크  
 라이선스 BSD  
 웹사이트 flask.pocoo.org

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

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

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

< 기본코드의 구조 >

## ★ 파이썬 출력 방법(format 이용)

### 7.1.1. 포맷 문자열 리터럴

**포맷 문자열 리터럴**(간단히 **f-문자열**이라고도 합니다)은 문자열에 `f` 또는 `F` 접두어를 붙이고 표현식 `{expression}`로 작성하여 문자열에 파이썬 표현식의 값을 삽입할 수 있게 합니다.

선택적인 포맷 지정자가 표현식 뒤에 올 수 있습니다. 이것으로 값이 포맷되는 방식을 더 정교하게 제어할 수 있습니다. 다음 예는 원주율을 소수점 이하 세 자리로 반올림합니다.

```
>>> import math
>>> print(f'The value of pi is approximately {math.pi:.3f}.')
The value of pi is approximately 3.142.
```

`:` 뒤에 정수를 전달하면 해당 필드의 최소 문자 폭이 됩니다. 열을 줄 맞춤할 때 편리합니다.

```
>>> table = {'Sjoerd': 4127, 'Jack': 4098, 'Dcab': 7678}
>>> for name, phone in table.items():
...     print(f'{name:10} ==> {phone:10d}')
...
Sjoerd      ==>      4127
Jack        ==>      4098
Dcab        ==>      7678
```

다른 수정자를 사용하면 포맷되기 전에 값을 변환할 수 있습니다. `'!a'`는 `ascii()`를, `'!s'`는 `str()`을, `'!r'`는 `repr()`을 적용합니다.:

```
>>> animals = 'eels'
>>> print(f'My hovercraft is full of {animals}.')
My hovercraft is full of eels.
>>> print(f'My hovercraft is full of {animals!r}.')
My hovercraft is full of 'eels'.
```

★ `request.get_json()`은 Flask에서 넘어온 데이터를 json형식으로 읽어들이어서 dictionary 형식으로 쓸 수 있게 해줌(POST 방식에 해당)



## < 블록체인 확장 >

2018. 11. 09(금)

1. Error 종류 확인? 500대 에러(server 오류)  
400대 에러(client 오류) : 404(페이지 없음), 405(GET/PUT 잘못), 403(허가받지 못한자)
2. network 상 가상의 주소를 할당해줄 때? 보통 192.168.xx.xx로 시작
3. python은 순차코딩!!  
클래스를 생성하고 def로 함수를 만들게 되면 메모리에 정보가 차곡차곡 쌓인다!  
이후 맨 밑에 실행코드를 넣어줘야 함!(if \_\_name\_\_ == '\_\_main\_\_': ~~)  
그렇게 해야 메모리에 데이터가 쌓이고, 실행코드가 실행될 수 있다.
4. 만약 내 코드가 정상작동하지 않을 때, 정상 작동하는 하나의 코드를 받아서 현 위치에 붙여  
넣기 → 두 개의 파일을 ctrl로 선택 후 마우스 우클릭 → 'Compare selected'를 선택하면,  
어느 위치의 어떤 내용이 다른지(바뀌었는지) 알려줌(파이참도 제공해주는 기능)
5. 오류가 났을 때?? visual studio code의 터미널 창에 해당 오류 내용이 뜬다.  
**맨 밑에서부터 순차적으로 올라가기!!**  
또는 터미널창 젤 좌측에 **problem**칸 눌러보면 오류내용 뜨니까 그걸로 확인!
6. 디버깅(오류를 잡아가는 과정) : 모든 프로그래밍에서 동일하다!  
visual studio code에서는 F5 단축키를 눌러 실행(실행모드가 아닌 프로그래머 입장에서)



```
CALL STACK
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Python Debug Console

여다희@DESKTOP-1CD712 MINGW64 /c/work_blockchain
$ activate base

여다희@DESKTOP-1CD712 MINGW64 /c/work_blockchain
$ cd "c:\work_blockchain"; env "PYTHONIOENCODING=UTF-8" "PYTHONUNBUFFERED=1" "C:\Anaconda3\python.exe" "c:\Users\여다희\.vscode\extensions\ms-python.python-2018.9.2\pythonFiles\experimental\ptvsd_launcher.py" 49720 "c:\work_blockchain\openapi.py"
* Serving Flask app "openapi" (lazy loading)
* Environment: production
WARNING: Do not use the development server in a production environment.
Use a production WSGI server instead.
* Debug mode: off
* Running on http://0.0.0.0:5001/ (Press CTRL+C to quit)
```

빨간점은 break\_point로 이 지점까지만 실행하고 잠시 멈춰라!



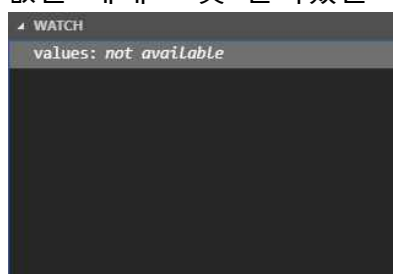
```
81 return jsonify(response), 200
82 @app.route('/transactions/new', methods=['POST']) # 새로운 Tx Data를 생성(POST)
83 def new_transaction():
84     ...
85     { 'sender' : '0x00000001', 'recipient' : '0x99999990', 'amount' : 5 }
86     ...
```

그 다음 아래 아이콘들을 기준으로 디버깅 실시



(멈추기, 건너뛰기, 들어가기, 탈출하기, 재실행, 취소)

break-point를 적절하게 이용하여 문제점을 하나씩(코드 한줄 씩) 실행해가면서 찾아보자!!  
값을 제대로 못 받아왔을 수 있기 때문에 좌측 배너의 watch를 확인하는 것도 병행



watch의 values에 알고자하는 값 넣으면 이 서버가 잘 받아오고 있는지 알 수 있음

**Debugging : 버그(논리적 오류)를 해결하는 과정!!**

## 7. API → OpenAPI → HTTP CRUD(어제 nodes/resolve 했던것에 이어서)

- /nodes/list -> node의 목록출력(GET)
- /nodes/resolve나 /nodes/list에서 reponse가 없는 node들을 삭제
- /getinfo -> 블록의 정보를 출력(GET)  
parameter : 블록의 index 전달 (블록의 index값-1을 통해 해당 방 번호 찾아가기)  
list는 0부터 시작하니까 -1해줌!

오디오북 듣기 / 한 권 책 사면 5번은 읽어보라..! / 'block화폐 2.0'도서

상태를 바꿀 때! 금액이 드는 것(이런 문제로 검색 할 때 비용이 든다고 말한 것)

bitcoin은 DDOS 공격을 예방하기 위해서 for문을 없앴(반복문 없이 쓰고, 받아오기만 가능하게)

→ ethereum은 생각이 달랐음(bitcoin에 없었던 반복문 등을 넣어 고차원적으로 변화)

모든 for문을 사용하기 위해서 GAS라는 개념을 도입

(모든 프로그램을 실행 할 때 비용 받기 위해서 / ddos를 예방하기 위해서 이 개념을 도입한 것)

코인과 토큰의 차이? 토큰은 거래하기에 부족한.....무언가고 코인은 실제 거래소에서 이용되는 것!