1. 자바스크립트 학습

이더리움의 web3.js 문법 이해 및 웹에서 제작하는 트랜잭션 처리의 기초

HTML, CSS는 따로

자바스크립트 – 웹 브라우저에서 많이 사용하는 프로그래밍 언어, 클라이언트 언어: 인터넷과 별도로 해석되는 언어(<->서버언어), 브라우저에서 해석(개발자가 작성한 언어는 컴퓨터에서 담당)

클라이언트는 서버에 요청, 서버는 응답

HTML: 꺽쇠(태그)로 문법이 작성, 줄바꿈, 띄어쓰기 X -> br 태그와 ;활용, CSS와 JS를 작성할 수 있는 토대, 태그는 element와 attribute로 이루어짐.

CSS: HTML문서를 디자인, 꾸며주는 역할, css파일을 HTML에 추가할 수 있고 그냥 넣을 수 있음

JS: HTML문서내에서 동적 프로그래밍 담당, script 태그 내부에서만 동작, 혹은 외부 파일 연결

<html의 최소 형식>

<html>

<head>

</head>

<style>

Body {

Background: pink;

}

</style>

<script>

Alert(‘브라우저 오픈 시 경고창’)

Var a = ‘글자’

Alert(a)

Function c(전달될 내용) {

Console.log(전달될 내용);

Return 15;

}

Var a = c(“콘솔에 출력”);

Consol.log(a)

</script>

<body>

가 <br><br> (엔터)

나 &nbsp; &nbsp; 다 (띄어쓰기)

</body>

</html>

script태그는 어느 곳이든 위치, 인터넷 안되어도 브라우저만 설치되어있다면 JS문법 사용간으

위에서 아래방향으로 한 문장씩 해석

변수로 할 수 없는 것 – 예약어, 숫자, \_와 $ 외의 특수문자, 공백 포함 문자

배열 생성

Var arr = Array(값1, 값2…);

Var arr = new Array(값1, 값2….);

배열 크기는 배열명.length, 자바와 달리 크기 변경 가능

함수

Function 함수명(매개변수…){실행부분}

Ex) function c(전달될 내용) {

Console.log(전달될 내용);

Return 15 or ‘문자’;

}

Var a = c(“콘솔에 출력”);

Consol.log(a)

선언된 function은 위에서 아래순서 브라우저에서 해석되는 것이 아니라 가장 우선 순위로 해석

함수는 실행시에 함수명+()를 씀, 함수도 하나의 자료형으로 변수에 담을 수 있음.

객체는 {} 사용

Var mouse = {

제조사: ‘hp’,

무게: 140,

‘버튼 수’: 2, //문자열로 처리 시 []로만 접근 가능 -> console.log(mouse.’버튼 수’) 불가

Is휠: true

JSON(JavaScript Object Notation): 자료를 저장하는 방식, 자바스크립트의 객체 작성과 유사.

중괄호를 활용해 키와 값을 넣음, XML보다 비교적 가벼움.

1. 파이썬 프로그래밍

다양한 모듈을 통해 블록체인 특성을 짧은 시간에 알 수 있음.

인터프리터 언어: 실행과 동시에 한줄씩 해석하여 실행

<->컴파일 언어: 작성한 코드를 기계어로 번역 후 실행

객체 지향 중심의 언어: 자료 타입이 class로 지정

기본 자료형 bool (True, False), int, float, str, list, tuple(정해진 값 바꿀 수 없음), dict(key와 value)

내장함수 print(a), type(a), dir(a): 매개변수 a에 적용할 수 있는 함수를 리스트로 반환

Range(1,10): 1~9 정수를 순차적으로 반환, len(a), input(): 콘솔창에 사용자 입력을 받아 문자열로 반환

반복문

Ex) dic = {‘key1’: ‘value1’, ‘key2:’ value2’}

For gab in dic :

Print(dic[gab])

Value1, value2 출력

함수(def)

문법: def 함수명(매개변수):

실행내용

선언된 함수의 실행 내용은 들여쓰기로 구분, 함수는 실행시에 함수명+()를 씀.

클래스

Class Calculator: #보통 대문자로 시작

Def \_\_init\_\_(self):

Self.result = 0

Def adder(self, num):

Self.result += num

Return self.result

Cal1 = Calculator()

Cal2 = Calculator()

Cal3 = Calculator()

Print(cal1.adder(10))

Print(cal2.adder(4))

Print(cal3.adder(2))

모듈

함수나 변수, 클래스들을 모아놓은 파이썬파일

모듈을 불러오기 위해 import 키워드 사용

모듈의 이름은 숫자로 시작할 수 없고, 띄어쓰기가 있으면 안됨

기본으로 제공하는 내장 모듈

외부에서 다운 받아 추가하는 외장 모듈(pip 명령어 사용)

사용자 정의 모듈

유명한 내자 모듈: time, random, pickle, threading, hashlib…

유명한 외장 모듈: request, ecdsa, flask…

웹서비스, 파일 관리, GUI프로그램, 데이터 분석

1. 자료구조

블록은 이전 블록의 해시값을 저장한 연결 리스트형 자료, 프로그램의 효율성에 기여

자료구조 – 단순구조(정수, 실수, 문자, 문자열), 선형구조(리스트, 연결 리스트, 스택, 큐), 비선형구조(트리, 그래프), 파일구조(순차파일, 색인파일)

알고리즘의 평가는 시간과 메모리 점유율에 의해 결정, 데이터 구조에 의존적

큐 – 대기열, 자료의 처리 방식으로 많이 쓰임, FIFO

Enqueue(들어가는 것), Dequeue(처음 들어온 것이 빠져나가는 경우)

파이썬에서는 용어 put, get

스택의 경우 LIFO – 쌓다, 자료의 처리 방식

Push, pop

리스트 – 데이터가 나란히 있는 구조, 자료의 저장 방식, 나열된 자료, 리스트, 연결 리스트, 양방향 연결리스트

트리 – 자료의 저장 방식, 뿌리와 가지, 검색 처리에 특화

1. 네트워크 프로그래밍

P2P통신 처리위해, 블록체인은 P2P통신 기반 알고리즘. 블록체인의 노드는 타 노드와 연결 유지해야함

소켓 – OSI 7 Layer의 2,3,4 계층을 대신해서 처리해주는 통신 프로세스

OSI 7 Layer: ISO 협회에서 만든 네트워크에 대한 표준 모델, 운영체제와 상관없이 두 컴퓨터가 통신을 하기 위한 기초 네트워크에 대한 규약 정의

7 – 응용 Application: 응용프로그램

6 - 표현 Presentation: 해당 파일 해석

5 – 세션 Session: 연결 유지

4 – 전송 Transport: Port 관리

3 – 네트워크 Network: IP 관리

2 – 데이터링크 Data link: MAC 주소 관리, 컴퓨터에 있는 네트워크 장비 정보

1 – 물리 Physical: 물리적경로, 실제 연결

ESTABLISHED: 현재 포트에 연결 존재

LISTENING: 해당 포트에서 연결 대기중

CLOSE\_WAIT: 통신 종료 대기중

cmd에서 netstat -ano

cmd에서 ipconfig /all

2계층: 해당 통신 장비에 부여된 고유번호 (MAC address) – 물리적 주소

3계층: 해당 PC까지 IP번호를 통해 통신 (IP address) – IPv4주소

4계층: 해당 PC내의 포트번호 통해 통신 (Service Port)

네트워크: 각 계층에서는 해당 계층에 필요한 모양으로 변환 진행, 7가지 계층을 통해 라우터와 라우터 연결해 통신

Layer 4 – TCP(Transmission Control Protocol 전송 제어 규약), UDP(User Datagram Protocol 사용자 데이터그램 규약)

TCP – 연결 보장, 안정, 느림, 보통의 통신 프로그래밍

안정성과 순서를 보장, 패킷을 내부적으로 알아서 순서대로 쪼갬

서로의 연결 확인하는 통신에서 사용

Socket.socket(AF\_INET, socket.SOCK\_STREAM)

UDP – 비연결형, 불완전, 빠름, 게임 프로그래밍

순서가 없고, 패킷이 크다면 수동으로 쪼개야함

Socket.socket(AF\_INET, socket.SOCK\_DGRAM)

AF\_INET -> IPv4 사용

Wireshrak – 네트워크 장비에 의해서 받은 패킷을 확인하는 프로그램, 해당되는 패킷을 더블 클릭하여 주고 받은 패킷 확인 가능

파이썬 소켓 함수

서버소켓(SS)

SOCKET.BIND((‘192.168.0.158’,12345)) – 튜플타입으로 호스트의 이름과 포트 번호 할당

SOCKET.LISTEN(0) – 준비된 소켓을 네트워크 상에서 대기

SOCKET.ACCEPT() – LISTEN 처리 후 클라이언트의 접속을 대기, 접속 시 클라이언트소켓과 주소 반환

클라이언트 소켓(CS)

SOCKET.CONNECT((‘192.168.0.158’,12345)) -> 튜플 타입으로 호스트의 이름과 포트번호로 접속

공통 메소드

SOCEKT.SEND(DATA) -> 소켓 통해 전달할 값, 데이터 타입은 반드시 BYTE타입

SCOKET.RECV(1024) -> 소켓을 통해 전달받은 값, 받을 때도 BYTE타입임 디코딩과정 필요, 숫자는 출력되는 버퍼사이즈, 2의 배수

SOCKET.CLOSE() -> 소켓 객체 소멸

자료는 byte타입으로 전송됨

블록체인 네트워크를 위해 리스트를 노드에 저장해두어야함

각 노드에 대한 네트워크 정보없이는 P2P를 구성할 수 없음.

1. 블록체인의 이해

비트코인 – 블록체인의 시초, 사토시 나카모토 2008년 10월 논문 작성 2009년 1월 처음 발행된 가상화폐

블록체인은 SC방식이 아닌 P2P 중심의 분산원장 저장방식

보안에 특화된 거래방식 저장에 용이

한번 저장된 블록은 수정불가

암호화 알고리즘

해시함수: 단방향 암호화, 복호화 불가, 비밀번호 DB저장

대칭키: 암호화 복호화 시에 같은 키를 사용, 와이파이 비번

비대칭키: 암호화 복호화 시에 다른 키를 사용, 공인인증서 방식, 공개키, 개인키 한 Set로 나옴

은행에서 사용 RSA방식, 비트코인은 ecdsa방식 -> 공개키, 개인키생성

Original PlainText –(A’s 공개키)-> Ciphertext –(A’s 개인키)-> Original Plaintext

블록 구조

블록헤더

버전: 데이터 구조의 버전 (version)

이전블록해시: 이전블록에 대한 해시값 (prevHash)

머클루트: 자기 블록의 거래정보로부터 생성된 머클트리의 루트에 대한 해시값(merkleRoot)

타임스탬프: 19700101 기준으로 경과한 초 (timestamp)

난이도 목표: 블록의 작업증명 알고리즘 (bits(난이도))

난스: 작업증명 알고리즘에 사용되는 카운터 (nonce-생성할 때까지 돌려서 나온 난수)

거래정보

트랜잭션 카운트

코인베이스 트랜잭션: 채굴 성공 시 발생되는 트랜잭션의 정보

트랜잭션: 다음 블록 생성전까지 수집한 거래정보

머클트리 – TX(거래) 2개의 값을 통해 SHA 256 통해 특정 해시 값 얻음. 최종적으로 머클루트 값 존재하게 되는 트리

블록체인 네트워크 구성원

블록체인 노드: 트랜잭션 내역 보관, 트랜잭션 승인, 분산합의 Full Node

블록체인 클라이언트: 트랜잭션 생성, 거래내역확인 Light Client

노드/클라이언트

채굴 및 블록생성 – 가능/불가

트랜잭션 내역보관 – 가능/불가

트랜잭션 승인 – 가능/불가

분산합의 – 가능/불가

트랜잭션 생성 – 불가/가능

사례 – 설치된 블록체인/코인지갑

노드 자체에서 거래가 가능한 것은 노드와 클라이언트 시스템을 같이 만든 블록체인 시스템만 가능

블록의 채굴과 생성

상태 - 채굴 전, 채굴 후 (현재 블록), 다음 블록 생성 후

거래정보(트랜잭션) – 사용불가(빈 블록), 사용가능(블록안에 들어가 확산됨), 사용불가

블록헤더 – Nonce값 변경 가능(처음에 0이고 1씩 증가시켜 목표하는 난이도보다 낮은 해시 값이 등장할 때까지의 값을 저장->찾으면 채굴 성공), 변경불가, 변경불가

블록해쉬 – 사용불가, 변경 중, 변경 불가

특징 – 채굴, 트랜잭션 반영, 거래성사

채굴난이도

비트코인 난이도는 2016블록마다 변경(약 2주), 한 블록 10분마다 생성이기 때문

TARGET이라는 것은 256bit의 숫자

난이도 = 최고목표값/현재목표값 Nonce값이 이거보다 낮아지면 블록 생성, 최고목표값 = 2.7\*10^67

현재 목표값 -> 블록헤더의 BIT를 16진수로 변환

Ex)BITS = 0X1809B31B

0X(뒤 6자리)\*2\*\*(8\*0X(앞 2자리-3)

Ex)1809B31B->0x09B31B\*2\*\*(8\*(0X18-3))

BITS의 값이 낮아질수록 난이도는 높아짐.

트랜잭션의 구조

내부에서는 개인 키와 공개키 통해 알아볼 수 없음. 개인 키로 확인 가능

1. 블록체인의 활용

이더리움 – 비탈릭 부테린이 개발한 암호화폐 플랫폼

플랫폼: 해당 시스템을 기반으로 타 서비스 제공 가능

이더리움 기반의 네트워크를 활용하여 계약, SNS, 전자토표 등이 가능

즉, 네트워크 상에 추가적인 서비스 추가가 가능

Geth

이더리움은 GO언어메인, Geth를 활용해 이더리움 네트워크 활용한 클라이언트 구축 가능

Console: 명령 창, Contract: 스마트 컨트랙트 등…

이더의 송금 및 계약은 수수료가 발생

수수료는 GAS로 지불, 같은 거래이더라도 GAS가 높을수록 채굴자에게 선정될 확률이 높고, 계약처리가 우선순위로 처리.

이더리움의 화폐단위: 1이더 = 10^18웨이

Sol -> Solidity

스마트 컨트랙트

이더리움은 이더리움 네트워크를 활용해 서비스 추가 가능

스마트 컨트랙트는 이더리움 계좌를 연동하여 특정 상황에서 디지털 거래가 자동으로 처리되도록 함.

해당 거래는 네트워크 노드의 EVM(Ethereum Virtual Machine)에서 번역 되어 트랜잭션 처리가 됨(GAS소모)

수수료 = 쓰이는 GAS의 총량 \* GAS가격(GWEI: 0.000000001)

Solidity

이더리움의 독자적인 프로그래밍 언어

스마트 계약 작성에 필요로 함. \*.SOL

작성된 솔리디티 파일은 웹에서 COMPILE(개발자가 작성한 파일을 컴퓨터가 읽도록 이진화하는 것) 을 통 하여 네트워크에 올릴 수 있음.

REMIX: 웹 상에서 Solidity 작성 및 배포를 가능하게 하는 사이트

<http://remix.ethereum.org>

Web3.js

Web3.js는 리믹스가 아닌 개별로 제작한 웹페이지에서 스마트 컨트랙트를 동작하도록 하는 자바스크립트 파일

이더리움 JSON RPC를 구현한 자바스크립트 API

(JSON RPC: JSON으로 인코딩된 원격 프로시저 호출)

별도의 웹페이지에서 이더리움을 활용한 DAPP을 제작 시 활용

토큰

DAPP에 기반한 전자 주식, 이더리움 상에서 주고 받을 수 있는 내부화폐

ERC20, ERC721 (Ethereum Request for Comment)

이더리움 표준의 토큰 제작방식

크립토키티, 크립토좀비 (사용자의 필요성 중요)