**컴퓨팅 사고(01)**

**Project**

웹사이트 자동 오픈 알고리즘

**담당교수 :**

**제출일 : 2019. 11. 24.**

**팀: 2조**

**학과 : 컴퓨터정보공학부**

**학번 : 2017202011**

**이름 : 김 영 우**

1. **Introduction**

* 팀 & 팀구성원 & 업무분장

1. 주제
   * + 웹사이트 자동 오픈 알고리즘

사용자가 등록한 웹사이트를 등록한 시간에 자동으로 열어주는 프로그램 구현

만약 컴퓨터가 꺼져 있으면 카카오톡으로 알림 송신

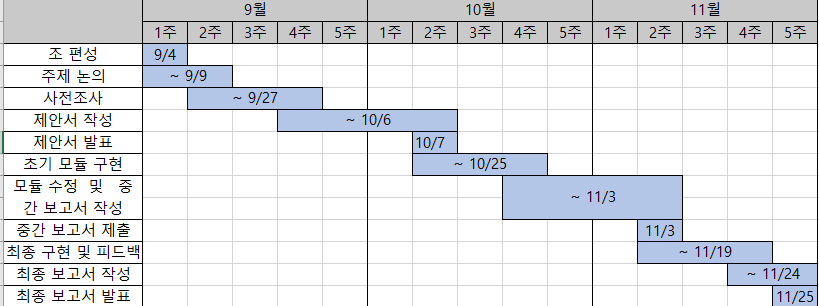
주간반복, 월간반복, 연간반복기능 추가

1. 팀 구성원 및 업무 분장

|  |  |  |
| --- | --- | --- |
| 분업표 | | |
| 이름 | 역할 | 비고 |
| 김하슬 | 자료 정리, DB작성 및 관련 모듈 | 조장 |
| 김재윤 | ppt제작, DB스캔 모듈과 main함수 | 중간 발표 |
| 김영우 | 자료조사, 서버-클라이언트 소켓, Kakao API | 최종 발표 |
| 이홍섭 | 서류작업 |  |

* 프로젝트 스케쥴

1. 간트 차트 및 스케쥴표





1. 회의록

실제 오프라인 상에서 만나서 회의 후 그후 사항은 카카오톡 메신저를 통해 의사소통을 진행하였다. 아래의 표는 오프라인상의 회의를 내용이다.

|  |  |
| --- | --- |
| 회의 일자 | 9/25 |
| 회의 내용 | 주제 선정 : |
| 1. 미리 설정한 웹사이트를 자동으로 켜지게 만들기 |
| ┗인강 같은 경우 자동 재생도 가능하도록 |
| ┗전원 연결 없을 시 카카오톡 푸쉬 알림 |
| 2. 또는 스도쿠 지뢰찾기 같은 간단한 게임 해결 알고리즘 |
|  |
| 간단한 역할 분담 |

|  |  |
| --- | --- |
| 회의 일자 | 10/23 |
| 회의 내용 | 베이스를 먼저 제작 |
| 인강 자동 재생은 제외하는 것으로 결정 |
| DB는 일단 SQLite를 쓰는 방향 |
| 입력방법은 PC를 기본으로 하되, 차후에는 모바일도 고려 |
| 기한은 10/26 토요일까지로 정함 |

|  |  |
| --- | --- |
| 회의 일자 | 11/8 |
| 회의 내용 | 기존 DB에서 문제점 발견 새로 DB모듈을 작성하기로 결정 |
| 기존 DB 문제점 : db테이블의 저장방식이 효율적이지 못함 |
| 날짜가 포함되어 있지 않아 주간, 월간, 연간 반복이 불가 |
| main함수는 schedule 모듈을 사용해서 구현 예정 |

|  |  |
| --- | --- |
| 회의 일자 | 11/20 |
| 회의 내용 | 테스트 진행 |
| 발생한 오류들의 해결방안 구상 |
| 최종적으로 역할 및 세부사항 재점검 |
| 메인 함수에 사용할 모듈을 schedule모듈에서 multiprocessing모듈로 변경 |

1. **Making Problem**

* 컴퓨팅 사고 모델에 기반한 문제 만들기

**2-1. Making Problem Introduction**

프로젝트 주제 : 웹사이트 자동 오픈 알고리즘

* + - 사용자가 등록한 웹사이트를 등록한 시간에 자동으로 열어주는 프로그램 구현
    - 만약 컴퓨터가 꺼져 있으면 카카오톡으로 알림 송신
    - 주간반복, 월간반복, 연간반복기능 추가
  1. 분해

서버-클라이언트 구현

-url을 전달할 서버-클라이언트

-DB에 입력할 데이터를 전달할 서버-클라이언트

-카카오톡 전송 구현

DB 작성

-DB생성

-DB입력

-DB에서 값 찾기

-DB 전체 출력

클라이언트의 생존 여부 확인

현재시간과 오픈하고 싶은 url의 시간이 일치하는지 확인

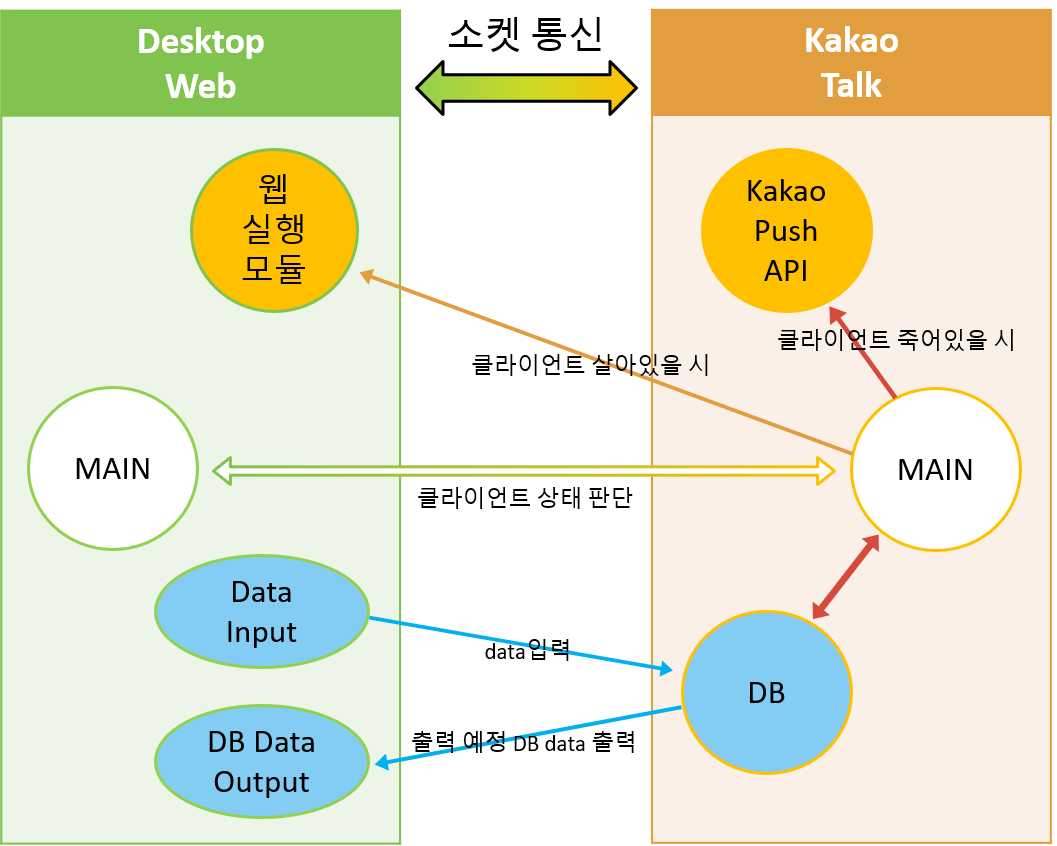
* 1. 패턴인식
     + 현재 시간과 열고 싶은 url의 시간이 일치하는지 확인하는 패턴
     + DB를 입력/출력 패턴
     + 서버- 클라이언트 통신 패턴
  2. 추상화
     + 컴퓨터가 꺼져 있는 것을 인식하려면?
       - 데이터를 저장하고 꺼진 것을 판단하는 곳이 컴퓨터와 분리 되어야한다
         * 소켓 통신을 통해 서버- 클라이언트로 꺼진 것을 판단하는 곳을 나눈다

소켓통신에서 클라이언트의 응답이 없으면 클라이언트는 꺼진 것이다.

클라이언트가 꺼졌다고 판단되면 컴퓨터는 꺼진 것이다.

결론 : 클라이언트의 응답이 일정시간 없으면 컴퓨터가 꺼져 있는 것이다.

* + - 위의 결론을 바탕으로 전체적인 구조의 추상화



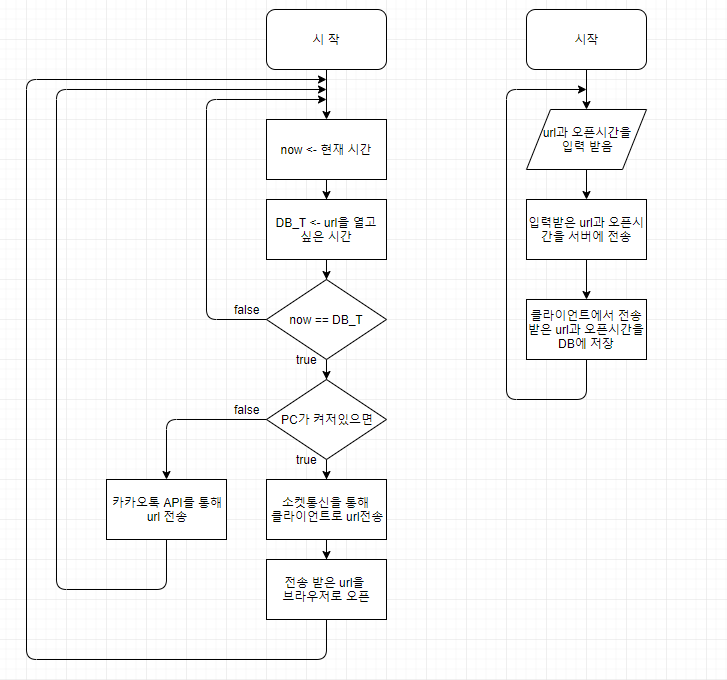
각 기능들을 원에 있는 기능으로 크게 서버 클라이언트 데이터베이스로 등으로

추상화하여 서로의 연관성을 보다 쉽게 인식

* 1. 알고리즘

아래의 플로우차트와 내용이 같음

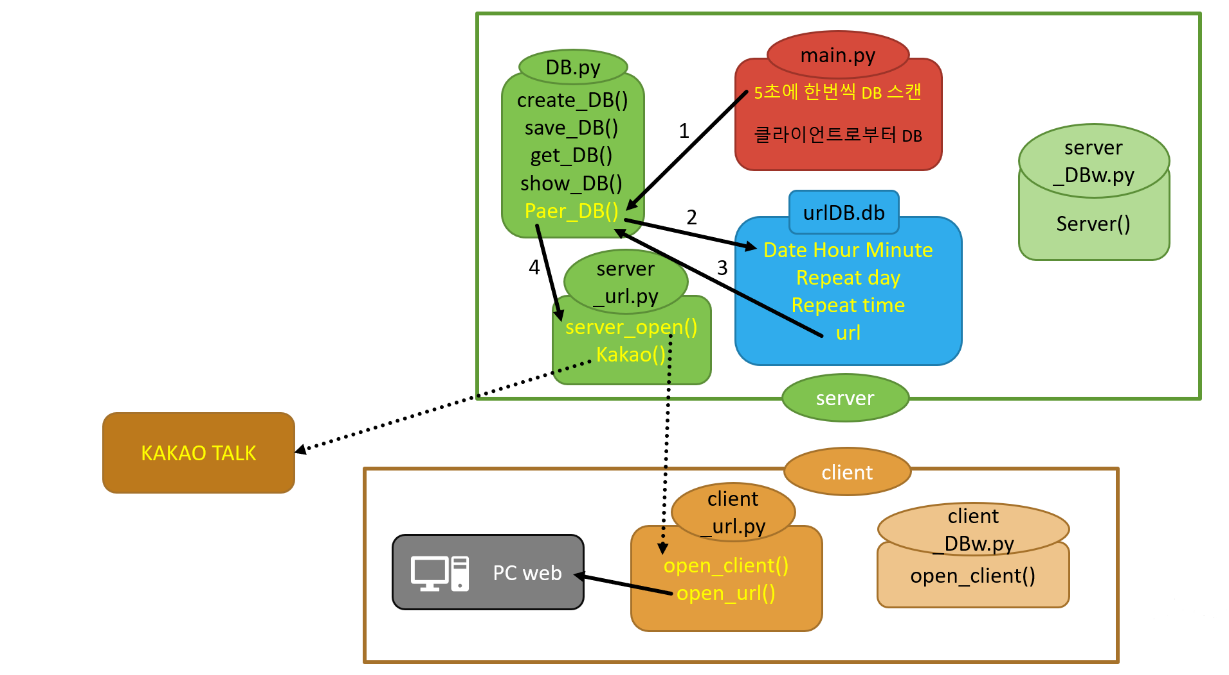
**2-2. Flow Chart**



* + - 1. url을 열고 싶은 시간과 현재 시간이 같으면 url을 클라이언트에 전달하여 PC브라우저로 오픈한다. 단 PC가 꺼져있을 시 카카오톡 API를 이용하여 카톡을 전송한다.
      2. 클라이언트가 정보를 입력하여 전송할 때까지 서버를 대기시키고 정보가 서버로 전송되면 DB에 저장한다.

**2-3. Algorithm**

**메인함수 1번째 동작**



1. Main.py 에서 5초에 한번씩 Paer\_DB()를 실행한다.
2. Paer\_DB에서는 현재시간과 ulrDB.db에 저장되어 있는 시간이 같은 지 확인한다.
3. 같다면, Paer\_DB는 url값을 반환하게 되고 url을 인자로 하여 client\_url.server\_open(url)함수를 실행한다.

4-1. 이때 client\_url.py가 켜져 있다면 open\_client()가 url을 수신한다.

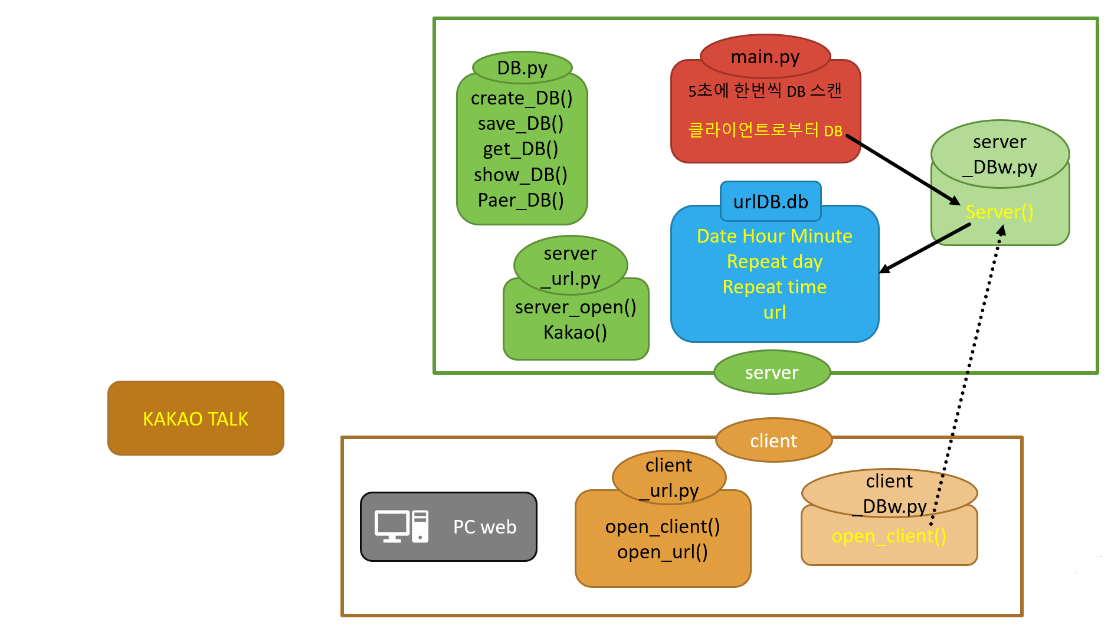
4-2. 수신된 url을 open\_url()함수를 통해 pc의 웹브라우저에 웹페이지를 띄운다.

4-1. client\_url.py가 꺼져 있어 client\_url.server\_open() 함수에서 클라이언트를

5초간 호출해도 응답이 없으면

4-2. url을 인자로 주어 Kakao(url)함수를 호출하여 카카오톡 메시지로 url을 전송한다.

**메인함수 2번째 동작**



클라이언트로부터 값을 수신 받기 위해 server()는 항상 대기중이며 client\_DBw.open\_client()을 실행해 키보드로부터 값을 입력 받고 입력 받은 값을 서버로 송신하면 받은 값을 재처리하여 DB에 저장한다.

**2-4. Code(text)**

1. **Server 부분**
   1. **main.py**

from multiprocessing import Process

def main\_fnc():

import server\_url

import DB

import time

print('scan start')

while True :

url = DB.Paer\_DB()

if str(url) != 'None' :

print(url)

server\_url.server\_open(url)

time.sleep(5)

def get\_data() :

import server\_DBw

while True :

server\_DBw.server()

if \_\_name\_\_ == '\_\_main\_\_' :

Process(target=main\_fnc).start()

Process(target=get\_data).start()

* 2개의 함수를 동시에 실행하기 위해 multiprocessing 모듈을 이용한다.
* 1번째 함수 : DB를 스캔하는 함수를 5초에 한번씩 호출 반환되는 값이 있으면 server\_url.server\_open()으로 전달한다.
* 2번째 함수 : 서버 server\_DBw.server()함수 클라이언트에서 DB에 입력할 값을 송신하는 것을 수신하기위해 대기한다.
* 무한반복문과 multiprocessing 모듈을 이용하여 한번에 위 2가지 함수를 실행한다.
  1. **DB.py**

def create\_DB () : # DB가 존재하지 않으면 새로 여는 함수

import sqlite3

conn= sqlite3.connect('urlDB.db')

cursor = conn.cursor()

cursor.execute(''' CREATE TABLE IF NOT EXISTS openlist(date text,url text,day int,repeat int)''')

# db안에 openlist테이블이 없으면 테이블 생성

res = cursor.fetchall()

#print(res)test 용

conn.commit()

conn.close()

def save\_DB() : # 키보드를 통해 값을 받아 저장하는 함수

import datetime

import time

import sqlite3

now = time.localtime()

mn = int(now.tm\_mon)

url = input('url을 입력하세요 : ')

d = input('일 입력(0:오늘 // 1~31) : ')

if d == '0' :

d = now.tm\_mday

elif int(d) < now.tm\_mday : # 만약 오늘보다 전날이면

mn = mn + 1 # 다음달로 저장

h = input('시간 입력(24시간) : ')

m = input('분 입력 : ')

day = input('반복(0 = 없음, 1= 매주, 2=매월 3=매년) :')

if(day =='1' or day =='2' or day =='3') :

re = input('얼마나 반복하겠습니까? (1회 ~~) :')

else:

re = '0'

s = "%04d-%02d-%02d %02d:%02d" % (now.tm\_year, mn, int(d), int(h), int(m))

#입력받은 시간을 년-월-일 시:분 형식으로 합쳐 저장

conn= sqlite3.connect('urlDB.db')

cursor = conn.cursor()

sql = 'INSERT INTO openlist (date,url,day,repeat) VALUES(?,?,?,?)'

data = (s,url,day,re)

cursor.execute(sql,data)

# 입력받은 정보를 가공히여 db에 저장

conn.commit()

conn.close()

def get\_DB(s,url,day,re) :

# 클라이언트를통해 입력받은 값을 서버를통해 불러와 저장할때 쓸 함수

import sqlite3

# 기존 키보드에서 입력받는 함수에서 입력받는 부분이 빠지고, 매개변수로 들어감

conn= sqlite3.connect('urlDB.db')

cursor = conn.cursor()

sql = 'INSERT INTO openlist (date,url,day,repeat) VALUES(?,?,?,?)'

data = (s,url,day,re)

cursor.execute(sql,data)

conn.commit()

conn.close()

def show\_DB() : # DB저장값을 보여주는 함수

import sqlite3

conn= sqlite3.connect('urlDB.db')

cursor = conn.cursor()

# urlDB.db안에있는 테이블 openlist 에있는 모든값을 출력

sql = 'SELECT\*FROM openlist'

cursor.execute(sql)

res = cursor.fetchall()

for row in res :

print(row)

conn.commit()

conn.close()

def Paer\_DB(): # 현재 시간과 비교하는 함수

import sqlite3

import time

import datetime

import dateutil.relativedelta # 윤년을 포함한 월 계산 가능

now = time.localtime()

s = "%04d-%02d-%02d %02d:%02d" % (now.tm\_year, now.tm\_mon, now.tm\_mday, now.tm\_hour, now.tm\_min)

print(s)

conn= sqlite3.connect('urlDB.db')

cursor = conn.cursor()

sql = 'SELECT\*FROM openlist WHERE date = ?'

da = datetime.datetime.strptime(wd[0], '%Y-%m-%d %H:%M')

cursor.execute(sql,(s,))

for row in cursor :

if row[2] == 0 :

print(row)

sql = 'DELETE FROM openlist WHERE date= ? AND url = ? AND day = ? '

cursor.execute(sql,(row[0],row[1],row[2]))

conn.commit()

# 삭제하고

return str(row[1])

if row[2] == 1 : # 주반복

if row[3] !=0 :

old = datetime.datetime.strptime(row[0], '%Y-%m-%d %H:%M')

add = dateutil.relativedelta.relativedelta(days=7)

new = old + add

sql = 'INSERT INTO openlist (date,url,day,repeat) VALUES(?,?,?,?)'

data = (new,row[1],row[2],row[3]-1)

cursor.execute(sql,data)

conn.commit()

print(row)

sql = 'DELETE FROM openlist WHERE date= ? AND url = ? AND day = ? '

cursor.execute(sql,(row[0],row[1],row[2]))

conn.commit()

# 삭제하고

return str(row[1])

elif row[2] == 2 : # 월 반복

if row[3] !=0 :

old = datetime.datetime.strptime(row[0], '%Y-%m-%d %H:%M')

add = dateutil.relativedelta.relativedelta(months=1) #모듈을 사용한 월 계산 (윤년 마지막날 계산 가능)

new = old + add

sql = 'INSERT INTO openlist (date,url,day,repeat) VALUES(?,?,?,?)'

data = (new,row[1],row[2],row[3]-1)

cursor.execute(sql,data)

conn.commit()

print(row)

sql = 'DELETE FROM openlist WHERE date= ? AND url = ? AND day = ? '

cursor.execute(sql,(row[0],row[1],row[2]))

conn.commit()

# 기존 출력할 것을 삭제하고

return str(row[1]) # url 반

elif row[2] == 3 : # 년 반복

if row[3] !=0 :

old = datetime.datetime.strptime(row[0], '%Y-%m-%d %H:%M')

add = dateutil.relativedelta.relativedelta(year=1)

new = old + add

sql = 'INSERT INTO openlist (date,url,day,repeat) VALUES(?,?,?,?)'

data = (new,row[1],row[2],row[3]-1)

cursor.execute(sql,data)

conn.commit()

print(row)

sql = 'DELETE FROM openlist WHERE date= ? AND url = ? AND day = ? '

cursor.execute(sql,(row[0],row[1],row[2]))

conn.commit()

# 기존 출력할 것을 삭제하고

return str(row[1]) # url 반

#return url

if \_\_name\_\_ == '\_\_main\_\_' :

create\_DB()

# test로 값을 5회 입력받아 입력받은값이 제대로 저장되어있는지

#db의 테이블값을 불러와 확인하기

print('test 5번 입력\n')

for i in range(5) :

print(f'{i+1}번째 입력')

save\_DB()

print('\n')

show\_DB()

* 이번 프로젝트 중 구현에 있어 가장 어려웠던 부분이다. 중간보고서 제출 전 구현하였으나, 여러 문제점이 발생하여 새로이 조원 모두가 sql문과 sqlite3 사용법을 익혀 구현하였다.
* **create\_DB 함수 :** DB가 존재하지 않으면 openlist라는 테이블을 생성**한다.**
* **save\_DB 함수 :** input함수를 통해 값을 입력 받고 시간정보를 재가공하여 DB에 저장하는 함수**이다.** 입력하는 오늘보다 전날을 입력하면 자동으로 다음달 바뀌게 처리한다.
* **get\_DB 함수 :** 클라이언트를 통해 DB에 저장할 값을 입력 받을 때 사용할 함수 위의 save\_DB 함수와 다르게 인자를 통해 전달 받고 전달받은 값을 시간정보를 재가공하여 DB에 저장한다.
* **swho\_DB 함수 :** DB값을 sql문으로 불러오고 print문을 통해 출력한다.
* **Paer\_DB 함수 :** DB.py안의 함수 중 가장 중요한 함수 DB안에 저장되어 있는 시간정보와 현재 함수를 실행할 때의 시간정보가 같으면 DB에서 삭제하고 url을 반환한다. 이때, 저장되어 있는 주간반복, 월간반복, 연간반복 및 반복횟수를 인식한다. dateutil.relativedelta모듈의 relativedelta함수(윤년계산 및 월의 달의 마지막일 자동 계산)를 이용하여 각 해당하는 반복만큼 일수를 추가한 다음 반복횟수를 차감하여 DB에 저장한다.
  1. **server\_url.py**

def kakao(mss) :

import requests # http 전송

import json # json http관련

mss =mss+ "을 열어야 합니다"

#https://developers.kakao.com/docs/restapi/tool 토큰 관련 페이지

token = '9XU5YA4RUEfisU531o5cnNtNC9U5dyNWYxD7HAo9dZoAAAFulIeMCA'

header = {"Authorization": 'Bearer ' + token}

url = "https://kapi.kakao.com/v2/api/talk/memo/default/send"

post = {

"object\_type": "text",

"text": mss,

"link": {

"web\_url": "https://developers.kakao.com",

"mobile\_web\_url": "https://developers.kakao.com"

},

}

data = {"template\_object": json.dumps(post)}

requests.post(url, headers=header, data=data)

print('open kakao')

return

def server\_open(url) :

# 통신 정보 설정

import socket

IP = '192.168.1.64'# 5050 포트포워딩 / amiya.asuscomm.com DDNS사용

PORT = 5050

SIZE = 1024

ADDR = (IP, PORT)

# 서버 소켓 설정

with socket.socket(socket.AF\_INET, socket.SOCK\_STREAM) as server\_socket:

server\_socket.bind(ADDR) # 주소 바인딩

server\_socket.settimeout(5) # 5초간 소켓에 반응이 없으면 에러

try :

server\_socket.listen() # 클라이언트의 요청을 받을 준비

client\_socket, client\_addr = server\_socket.accept()

# 수신대기, 접속한 클라이언트 정보 (소켓, 주소) 반환 (10 waiting )

msg = client\_socket.recv(SIZE) # 클라이언트가 보낸 메시지 반환

print("[{}]-- {} --".format(client\_addr,str(msg)))

# 클라이언트가 보낸 메시지 출력

client\_socket.sendall(url.encode()) # 클라이언트에게 응답

client\_socket.close() # 클라이언트 소켓 종료

except : #에러 발생시 (5초이상 반응이 없으면)

kakao(url)

if \_\_name\_\_ == '\_\_main\_\_' :

while True :

url = input('url open : ')

server\_open(url)

* **kakao 함수 :** url을 인자로 입력받아 카카오톡으로 전송하는 함수이다.kakao에서 제공하는 REST API를 이용하였다. kakao에서 제공하는 나에게 보내기 예제를수정하여 json 형식으로 python의 request모듈을 통해 카카오에서 제공하는 링크에 전송하여 구현하였다.
* **server\_open 함수 :**  url을 인자로 입력 받아 5초간 클라이언트의 반응을 기다린다. 반응이 있으면 클라이언트에 url을 전달하고, 반응이 없으면 kakao함수에 url을 전달하여 kakao를 통하여 url을 카카오톡으로 전송한다.
* **포트설정, DDNS설정은 3번 항목 3-1 서버설정에 자세히 설명하였다.**
  1. **server\_DBw.py**

def server() :

# 통신 정보 설정

import socket

import time

import DB

IP = '192.168.1.64'# 5051 포트포워딩 / amiya.asuscomm.com DDNS사용

PORT = 5051

SIZE = 1024

ADDR = (IP, PORT)

# 서버 소켓 설정

with socket.socket(socket.AF\_INET, socket.SOCK\_STREAM) as server\_socket:

server\_socket.bind(ADDR) # 주소 바인딩

# try :

wd = [0,0,0,0]

server\_socket.listen() # 클라이언트의 요청을 받을 준비

client\_socket, client\_addr = server\_socket.accept()

# 수신대기, 접속한 클라이언트 정보 (소켓, 주소) 반환 (10 waiting )

wd[0] = client\_socket.recv(SIZE) # 클라이언트가 보낸 메시지 반환

wd[1] = client\_socket.recv(SIZE) # 클라이언트가 보낸 메시지 반환

time.sleep(0.5)

wd[2] = client\_socket.recv(SIZE) # 클라이언트가 보낸 메시지 반환

time.sleep(0.5)

wd[3] = client\_socket.recv(SIZE) # 클라이언트가 보낸 메시지 반환

print('클라이언트로 부터 아래의 정보를 입력받았습니다.')

for i in range(4): # 바이너리 파일 문자열 변환

print(wd[i].decode('utf-8'))

DB.create\_DB() #DB가 없으면 생

DB.get\_DB(wd[0].decode('utf-8'),wd[1].decode('utf-8'),wd[2].decode('utf-8'),wd[3].decode('utf-8'))

if \_\_name\_\_ == '\_\_main\_\_' :

while True :

server()

* **server 함수 :** 클라이언트에서 DB에 저장할 값을 입력해 전송하기를 기다린다. DB에 저장할 값을 입력 받았으면, get\_DB함수에 인자로 전달하여 DB에 저장한다.
* **포트설정, DDNS설정은 3번 항목 3-1 서버설정에 자세히 설명하였다.**

1. **Client 부분**
   1. **client\_url.py**

def open\_client() :

import socket

SERVER\_IP = 'amiya.asuscomm.com' #ddns연결

SERVER\_PORT = 5050 # 포트 설정

SIZE = 1024

SERVER\_ADDR = (SERVER\_IP, SERVER\_PORT)

while True :

# 클라이언트 소켓 설정

with socket.socket(socket.AF\_INET, socket.SOCK\_STREAM) as client\_socket:

try:

client\_socket.connect(SERVER\_ADDR) # 서버에 접속

client\_socket.send('ok'.encode()) # 서버에 메시지 전송

url = client\_socket.recv(SIZE) # 서버로부터 응답받은 메시지 반환

open\_url(url)

except :

print('')# 연결오류 처리 - 더 좋은 방법은 없는지?

def open\_url(url) : #웹으로 url을 여는 함수

import webbrowser

url = url.decode('utf-8')

print(f'open {url}!')

webbrowser.open(url)

if \_\_name\_\_ == '\_\_main\_\_' :

open\_client()

* **open\_client() 함수 :** 대기하고 있다가 서버가 url을 송신하면 응답을 보내고 open\_url함수에 전달한다.
* **open\_url 함수 :** url을 인자로 입력받아 webbrowser모듈을 통해 클라이언트 컴퓨터의 기본 웹브라우저로 url을 연다.
* **포트설정, DDNS설정은 3번 항목 3-1 서버설정에 자세히 설명하였다.**
  1. **client\_DBw.py**

def open\_client() :

import socket

import time

import datetime

now = time.localtime()

SERVER\_IP = 'amiya.asuscomm.com' #ddns연결

SERVER\_PORT = 5051 # 포트 설정

SIZE = 1024

SERVER\_ADDR = (SERVER\_IP, SERVER\_PORT)

while True :

# 클라이언트 소켓 설정

with socket.socket(socket.AF\_INET, socket.SOCK\_STREAM) as client\_socket:

# client 구분 위한 고유번호 할당 = ID (자동으로 부여됨)

mn = int(now.tm\_mon)

url = input('url을 입력하세요 : ')

d = input('일 입력(0:오늘 // 1~31) : ')

if d == '0' :

d = now.tm\_mday

elif int(d) < now.tm\_mday : # 만약 오늘보다 전날이면

mn = mn + 1 # 다음달로 저장

h = input('시간 입력(24시간) : ')

m = input('분 입력 : ')

day = input('반복(0 = 없음, 1= 매주, 2=매월 3=매년) :')

if(day =='1' or day =='2' or day =='3') :

re = input('얼마나 반복하겠습니까? (1회 ~~) :')

else:

re = '0'

s = "%04d-%02d-%02d %02d:%02d" % (now.tm\_year, mn, int(d), int(h), int(m))

try:

client\_socket.connect(SERVER\_ADDR) # 서버에 접속for i in range(4):

client\_socket.send(s.encode()) # 서버에 메시지 전송

client\_socket.send(url.encode()) # 서버에 메시지 전송

time.sleep(0.5)

client\_socket.send(day.encode()) # 서버에 메시지 전송

time.sleep(0.5)

client\_socket.send(re.encode()) # 서버에 메시지 전송

# url = client\_socket.recv(SIZE) # 서버로부터 응답받은 메시지 반환

except :

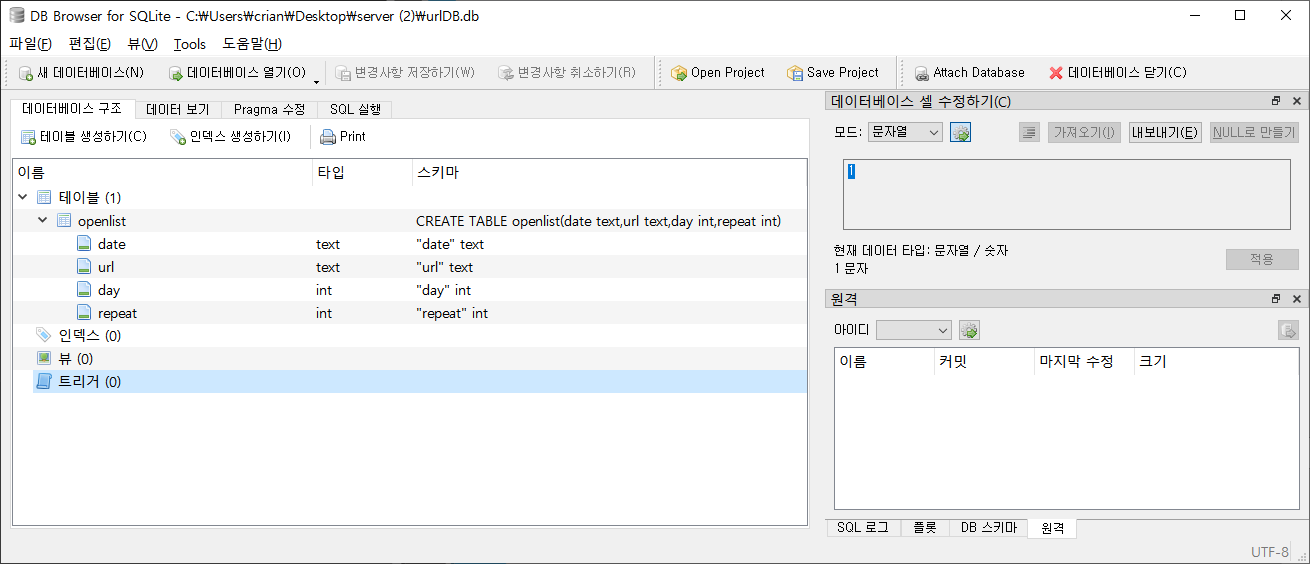
print('')# 연결오류 처리 - 더 좋은 방법은 없는지?

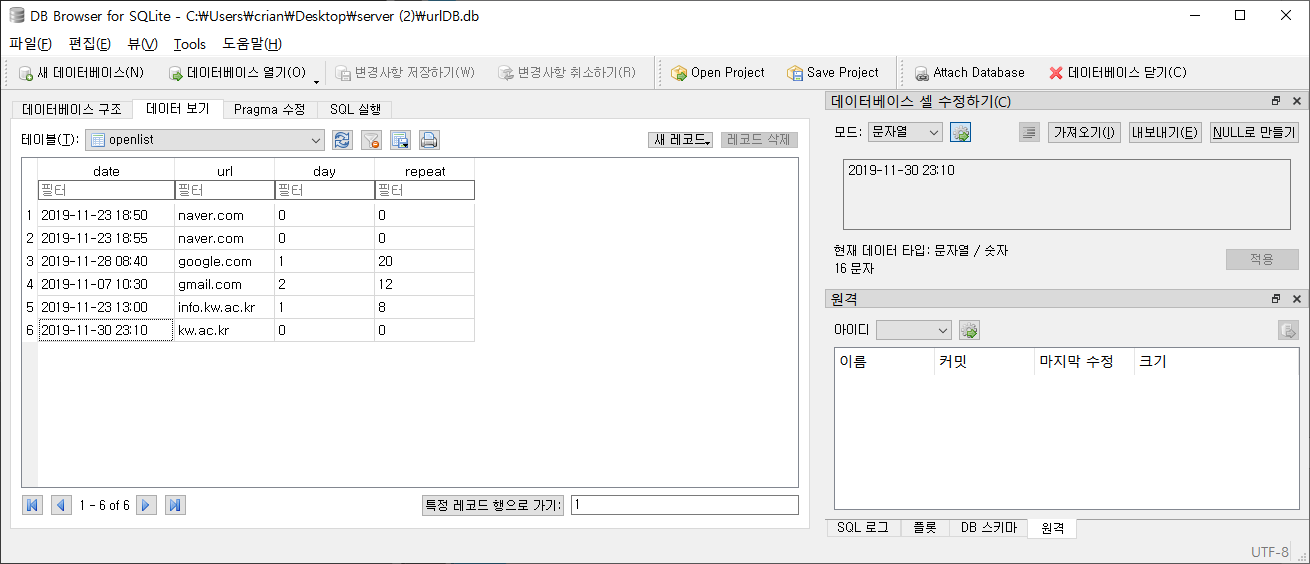
if \_\_name\_\_ == '\_\_main\_\_' :

open\_client()

* **open\_client 함수 :** DB에 저장할값을 input함수를 통해 입력. 받아 서버에 전달한다.
* **포트설정, DDNS설정은 3번 항목 3-1 서버설정에 자세히 설명하였다.**

1. **urlDB.db 구조**

****

****.

* **date :** 입력 받은 날짜를 년-월-일 시:분 의 형식으로 변환해서 저장한다**.**
* **url :** 출력할 url을 저장한다.
* **day : 반복할 기준을 저장한다 ( 0 = 반복없음, 1=주간반복, 2=월간반복, 3=연간반복)**
* **repeat : 반복할 횟수를 저장한다. (반복이 없을 시 입력 받지 않는다.)**

**2-5. Result Screen**

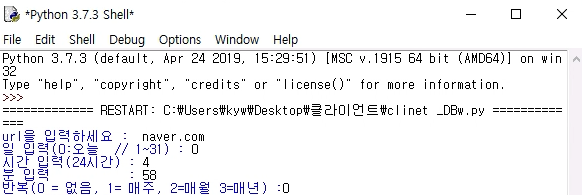
1. **최종 구현**

검은색 cmd창은 서버에서 실행된 것이며

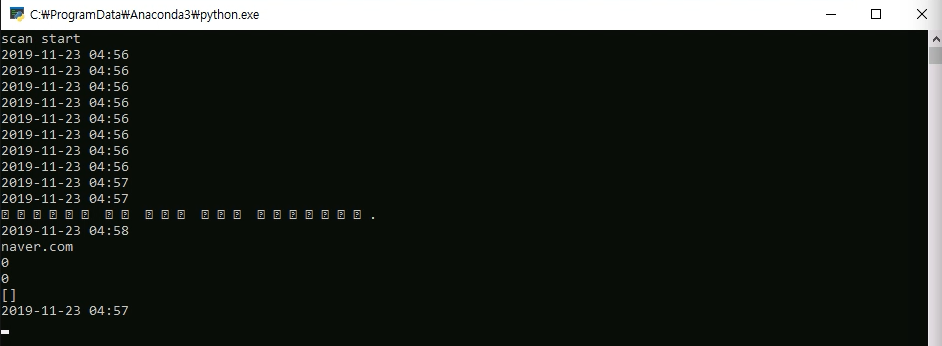
(서버가 영문판 windows이므로 cmd에서는 한글이 깨져 보이지만 정상 출력 되었음)

나머지는 클라이언트가 설치된 pc에서 실행되었음

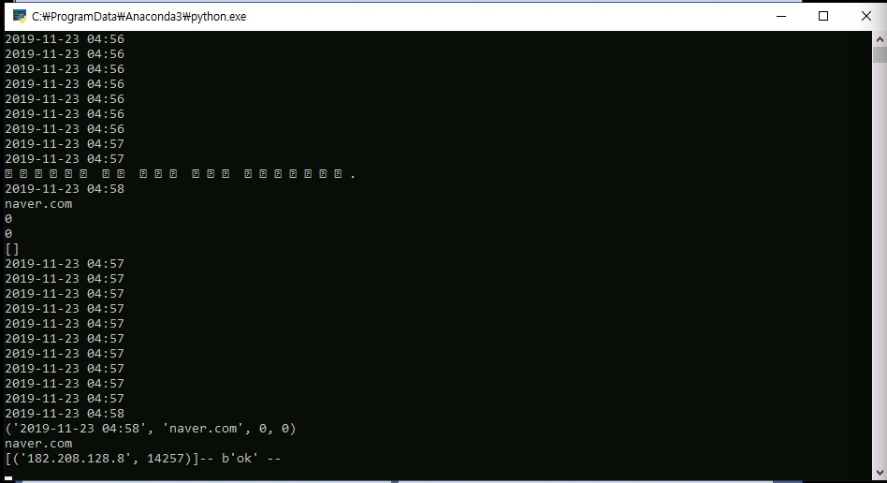
1. url을 전달받은 클라이언트가 실행되고 있을 때 (컴퓨터가 켜져있을 때)



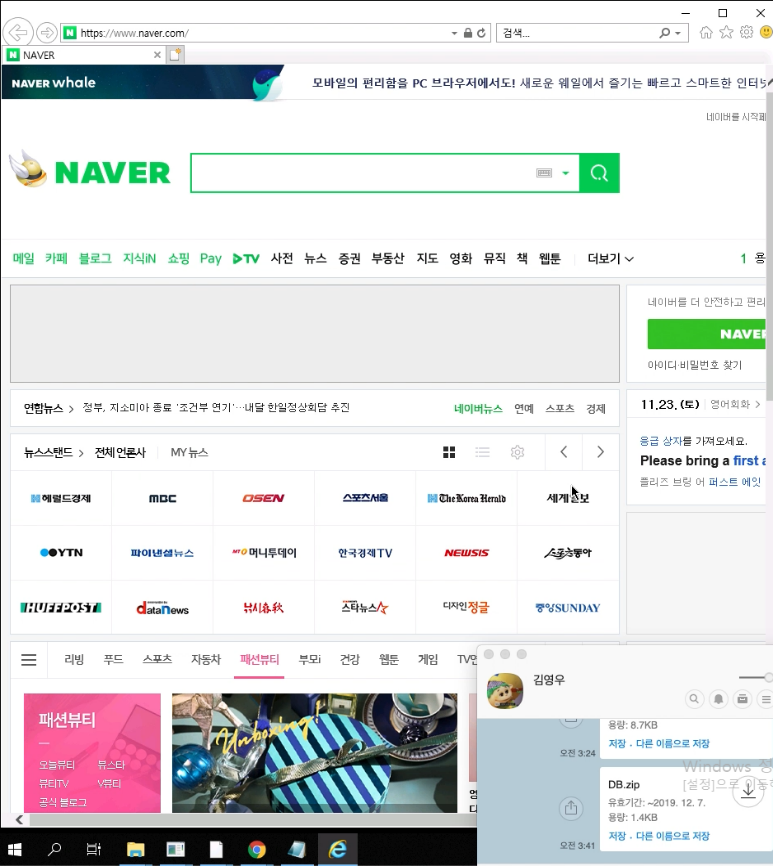
client\_DBw.py를 실행하여 DB에 저장할 값을 입력



서버 : main함수가 실행된 서버에 값이 전달되고 정상적으로DB에 저장되었음

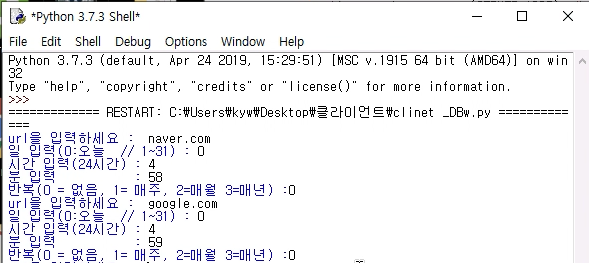


서버 : 현재 시간과 DB에 저장된 시간이 일치하자 클라이언트로 url(naver.com)값 전달

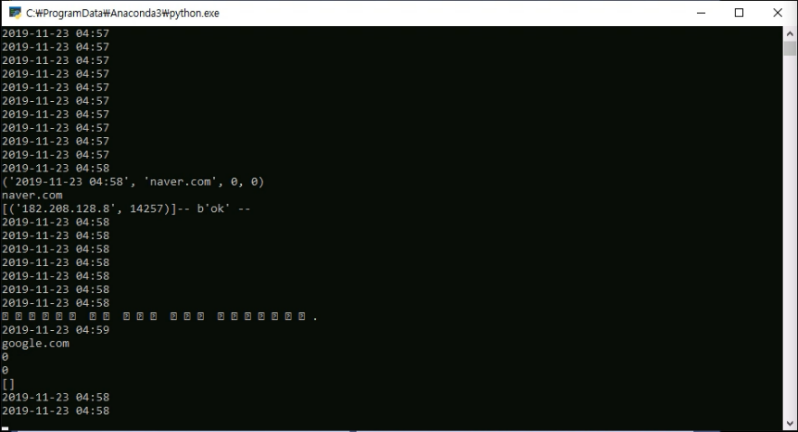


클라이언트가 살아있고 제대로 url(naver.com)이 전달되었으므로 url(naver.com)이 브라우저를 통해 켜짐

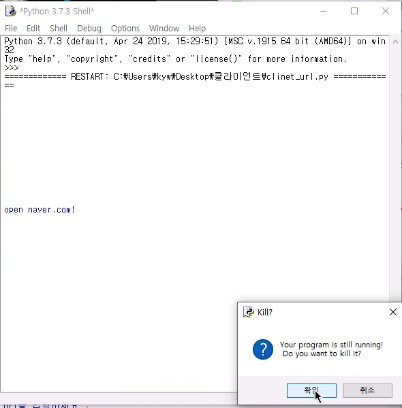
1. url을 전달받은 클라이언트가 꺼져 있을 때 (컴퓨터가 꺼져있을 때)



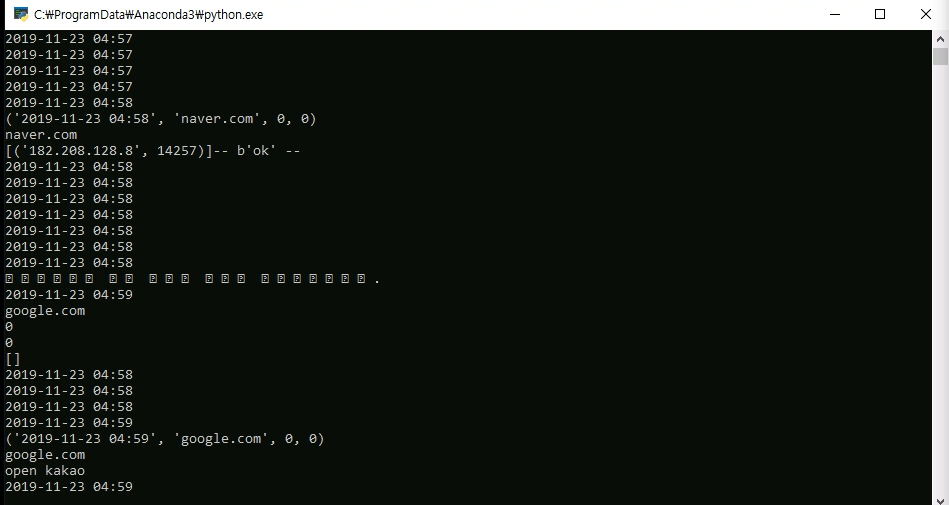
a)의 경우와 마찬가지로 DB에 저장할 새로운 값 입력



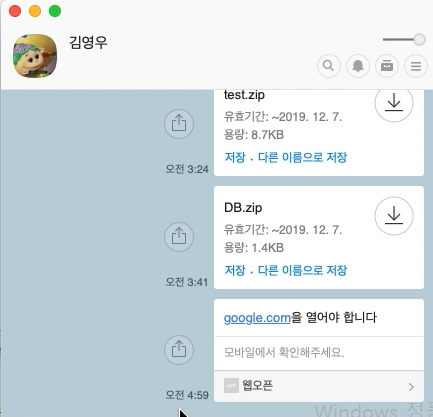
서버 : a)와 마찬가지로 정상적으로 DB에 값이 입력되었음



url을 받는 client\_url을 종료하였을때



서버 : 클라이언트가 종료된 것을 인식하고 kakao톡을 통해 url을 전송시킴



카카오톡으로 전송된 url을 확인할 수 있음

1. **본인이 구현한(&기여한) 부분 설명**
   1. **서버 설정**
2. **OS 세팅**

**모니터, 스크린샷이(가) 표시된 사진

자동 생성된 설명**

Window server 2019 사용이유

* 기존 데스크톱 pc에 마이크로소프트에서 학생에게 무료로 배포하는 Windows server 2019 설치하여 사용하였다.
* 윈도우10 보다 방화벽설정이 좀 더 세부적이며, 아직 팀원들 모두 리눅스 CUI에 익숙하지 못하여 GUI가 잘 되어있다는 장점 때문에 사용을 결정하였다.
* 모든 조원이 접근 가능하도록 windows에서 기본으로 제공하는 원격 서비스 RDP(Remote Desktop Protocol)에 접근가능한 계정을 2개 만들어 한번에 2명이 원격제어가 가능하게 설정하였다.

1. **DDNS 설정 및 포트포워딩, 방화벽 설정**

Pc가 설치되어 있는 곳은 가정용 회선이 들어온다. 가정 회선은 보통 고정IP가 아닌 유동IP가 많으므로 가정용 라우터에서 기본으로 제공하는 DDNS(Dynamic Domain Name System)을 사용하고 포트포워딩을 통해 server와 client를 연결하였다.

사용 라우터 : asus사의 가정용 라우터

사용 DDNS : amiya.asuscomm.com (라우터 기본 제공 DDNS)

* 포트포워딩 설정

1. 포트(port): 5050

용도 : 클라이언트에 url 전달하는 용도로 사용한다.

내부IP : Windows server 2019 IP (192.168.1.64)

1. 포트(port): 5051

용도 : 클라이언트로부터 값을 입력 받아 DB에 저장 하는 용도로 사용한다.

내부IP : Windows server 2019 IP (192.168.1.64)

* Windows server 2019 방화벽 설정

인바운드 규칙 및 아웃바운드 규칙 : 5050, 5051 허용

* 1. **소켓프로그래밍 (서버 – 클라이언트)**

1. **클라이언트로 url 전달 (server\_url.py – client\_url.py)**

* 클라이언트가 꺼져 있는지 판단하는 것이 어려웠다. socket 모듈의 settimeout(5)함수를 이용하여 5초간 클라이언트의 반응이 없으면 에러가 발생하게 만들었다. 에러가 발생하면 try,except 예외문을 통해 클라이언트가 꺼져 있는지를 판단하였다.
* 소켓을 통해 전달된 텍스트 데이터는 바이너리 파일로 저장되어 다시 문자열 utf-8로 인코딩하는 방법을 찾는데 어려움을 겪었다.

1. **클라이언트로부터 값을 입력 받고 DB저장 (server\_DBw.py – client\_DBw.py)**

* 서버에 DB에 저장할 값을 송신할 때 4개의 값은 전송하였다. 그러나 서버에서는 3번째 송신 때 4번째로 송신한 정보 또한 같이 들어오는 문제가 있었다. time모듈을통해 sleep함수로 0.5초의 시간차를 두고 3,4번째의 송신을 하는 것으로 문제를 해결하였다.
  1. **카카오톡 전송 함수 (server\_url.py 에서 kakao(mss) 함수)**
* 현재 구현 단계에서는 개인에게 전송해 주는 것만으로 충분하다 판단하였다.
* kakao developers에서 제공하는 REST API 서비스를 사용하였다.
* kakao developers 페이지에서 나에게 텍스트 템플릿 보내기 예제를 참조하였다. 예제를 수정하여 json 형식으로 python의 request모듈을 통해 카카오에서 제공하는 링크에 전송하여 구현하였다.
* 참고 예제 링크 : [developers.kakao.com/docs/restapi/kakaotalk-api#나에게-보내기-텍스트-템플릿-보내기](https://developers.kakao.com/docs/restapi/kakaotalk-api)

1. **참조자료**

**서버 관련:**

[**https://docs.python.org/ko/3.7/library/socket.html#socket.socket.settimeout**](https://docs.python.org/ko/3.7/library/socket.html)

[**https://docs.python.org/ko/3.7/tutorial/errors.html**](https://docs.python.org/ko/3.7/tutorial/errors.html)

[**https://stackoverrun.com/ko/q/10378431**](https://stackoverrun.com/ko/q/10378431)

**클라이언트 관련:**

[**https://wikidocs.net/16035**](https://wikidocs.net/16035)

[**https://docs.python.org/ko/3.7/library/webbrowser.html**](https://docs.python.org/ko/3.7/library/webbrowser.html)

**카카오톡 API 관련:**

[**developers.kakao.com/docs/restapi/kakaotalk-api#나에게-보내기-텍스트-템플릿-보내기**](https://developers.kakao.com/docs/restapi/kakaotalk-api)

[**https://developers.kakao.com/docs/restapi/tool**](https://developers.kakao.com/docs/restapi/tool)

**Sqlite3 DB관련:**

[**https://docs.python.org/3.7/library/sqlite3.html#sqlite3.Cursor.fetchall**](https://docs.python.org/3.7/library/sqlite3.html)

[**https://www.tutorialspoint.com/sqlite/sqlite\_using\_autoincrement.htm**](https://www.tutorialspoint.com/sqlite/sqlite_using_autoincrement.htm)

**날짜 및 시간 관련 :**

[**https://docs.python.org/ko/3/library/datetime.html**](https://docs.python.org/ko/3/library/datetime.html)[**https://codeday.me/ko/qa/20190407/188152.html**](https://codeday.me/ko/qa/20190407/188152.html)

**멀티 프로세스 관련 :**

[**https://docs.python.org/ko/3/library/multiprocessing.html**](https://docs.python.org/ko/3/library/multiprocessing.html)