1. 검색 키워드 엑셀orCSV 불러오기

In [5]:
```
import pandas as pd
from selenium import webdriver
from selenium.webdriver.chrome.options import Options
import requests
from bs4 import BeautifulSoup
import numpy as np
import time
import re

df_MCN = pd.read_csv('phone_info.csv')
```

2. 불러온 엑셀데이터로 차례대로 크롤링

2-1. 크롤링 데이터를 누적할 데이터프레임 생성

In [10]:
```
my_dict = {
    "maker": "",
    "group": "",
    "code": "",

    "name": "",
    "model_name": "",
    "storage": "",
    "color_name": "",

    "hidden":"",
    "storage_hidden": "",
    "color_hidden": "",
    "class_hidden": "",

    "size": "",
    "display_inch": "",
    "display_cm": "",
    "weight": "",
    "battery": "",
    "front_camera": "",
    "back_camera": "",

    "price": "",
    "score": "",
    "summary": ""
    }
crawl_data = pd.DataFrame(my_dict,index=[0])
```

2-2. 크롤링시 검색결과 존재여부 확인해주는 함수

In [11]:
```
def hasxpath(xpath):
    try:
        driver.find_element_by_xpath(xpath)
        return True
    except:
        return False
```

2-3. 세티즌 사이트에서 차례대로 검색후 수집 => 검색되지 않을시 빈 열로 남겨둔다

In [6]:
```
import os
from pyvirtualdisplay import Display
from selenium import webdriver
display = Display(visible=0, size=(800, 600))
display.start()
driver = webdriver.Chrome()
driver.implicitly_wait(3)

for i in range(len(df_MCN)):
# for i in range(1):
    url1 = "https://review.cetizen.com/review.php?q=phone&just_one=&just_one_name=&jus
    url2 = "#&p_data=3&p_split=&recnum=10"
    driver.get(url1+df_MCN.iloc[i][2]+url2)

    if hasxpath('//*[@id="product_list"]/div/div[3]/div[1]/div[1]/div/a/span') == Tru

        driver.find_element_by_xpath('//*[@id="product_list"]/div/div[3]/div[1]/div[1]
        driver.implicitly_wait(10)

        name = driver.find_element_by_xpath('/html/body/div[10]/div[3]/div[1]/div[2]/d
        model_name = driver.find_element_by_xpath('/html/body/div[10]/div[3]/div[1]/di
        storage = driver.find_element_by_xpath('//*[@id="product_specview"]/div[1]/for
        color_name = ""
        size = driver.find_element_by_xpath('//*[@id="product_specview"]/div[1]/form/d
        display_inch = driver.find_element_by_xpath('//*[@id="product_specview"]/div[1
        weight = driver.find_element_by_xpath('//*[@id="product_specview"]/div[1]/form
        battery = driver.find_element_by_xpath('//*[@id="product_specview"]/div[1]/for
        front_camera = driver.find_element_by_xpath('//*[@id="product_specview"]/div[1
        back_camera = driver.find_element_by_xpath('//*[@id="product_specview"]/div[1]
        price =  driver.find_element_by_xpath('//*[@id="product_specview"]/div[1]/form

        my_dict = {
            "maker":df_MCN.iloc[i][0],
            "group": df_MCN.iloc[i][1],
            "code": df_MCN.iloc[i][2],

            "name": name,  #모델한국명
            "model_name": model_name, #모델영문명
            "storage": storage, #용량
            "color_name": "", #모델색상

            "hidden": "", #모델사용여부 => 일단비워두자
            "storage_hidden": "", #용량사용여부 => ''
            "color_hidden": "", #색상사용여부
            "class_hidden": "", #등급사용여부?

            "size": size,
            "display_inch": display_inch,
            "display_cm": "",
#             "display_cm": str(round(float(display_inch)*2.5399,1)),
            "weight": weight,
            "battery": battery,
            "front_camera": front_camera,
            "back_camera": back_camera,

            "price": price, #출고가
```

```
            "score": "", #별점
            "summary": "" #스펙요약
        }
    else:
        my_dict = {
            "maker":df_MCN.iloc[i][0],
            "group": df_MCN.iloc[i][1],
            "code": df_MCN.iloc[i][2],
            "name": "",
            "model_name": "",
            "storage": "",
            "color_name": "",

            "hidden":"",
            "storage_hidden": "",
            "color_hidden": "",
            "class_hidden": "",

            "size": "",
            "display_inch": "",
            "display_cm": "",
            "weight": "",
            "battery": "",
            "front_camera": "",
            "back_camera": "",

            "price": "",
            "score": "",
            "summary": ""
        }
    a= pd.DataFrame(my_dict,index=[0])
    crawl_data = crawl_data.append(a)
    crawl_data = crawl_data.reset_index(drop=True)
```

3. 크롤링 결과 처리

3-1 모델명이 같은 기종들의 스펙데이터를 통합해주는 함수

In [17]:
```
def join_models(df):
    for i in df.index:
        if df.loc[i, "size"] == "":
            for j in df.index:
                if df.loc[j,"group"] == df.loc[i,"group"] and df.loc[j,"size"] != "":

                    df.loc[i,"name"] = df.loc[j,"name"]
                    df.loc[i,"size"] = df.loc[j,"size"]
                    df.loc[i,"storage"] = df.loc[j,"storage"]
                    df.loc[i,"model_name"] = df.loc[j,"model_name"]
                    df.loc[i,"display_inch"] = df.loc[j,"display_inch"]
                    df.loc[i,"weight"] = df.loc[j,"weight"]
                    df.loc[i,"battery"] = df.loc[j,"battery"]
                    df.loc[i,"front_camera"] = df.loc[j,"front_camera"]
                    df.loc[i,"back_camera"] = df.loc[j,"back_camera"]
                    df.loc[i,"price"] = df.loc[j,"price"]
                    break

join_models(crawl_data)
```

3-2. 크롤링 결과 데이터프레임 출력

In [2]:
```
with pd.option_context('display.max_rows', None, 'display.max_columns', None):  # mo
    crawl_data

pd.set_option('display.max_row', 500)
pd.set_option('display.max_columns', 100)
crawl_data
```

3-2. 크롤링 결과를 CSV파일로 저장

In [14]:
```
crawl_data.to_csv('crawl_data.csv',
                  sep=',')
```

3-3. 크롤링 결과를 마리아DB에 저장

In [ ]:
```
import pymysql
from pandas import Series, DataFrame
from sqlalchemy import create_engine

a = pd.read_csv("crawl_data2.csv")
engine = create_engine("mysql+pymysql://crawl:"+"0309"+"@localhost:3306/crawl?charset=
conn = engine.connect()
a.to_sql(name="cetizen", con=engine, if_exists="replace", index=False)
```