Do IT Quant

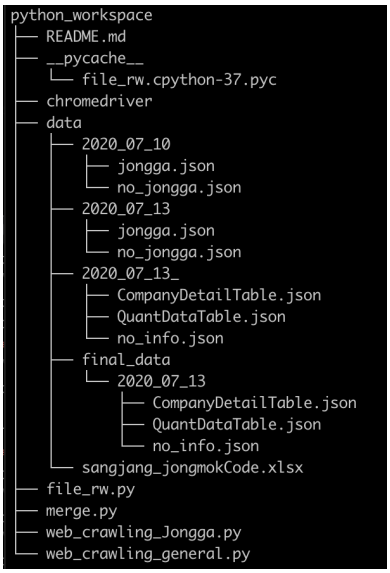
DIQ

본 문서는 KB데이타시스템 연수원 프로젝트에서 데이터를 처리하기 위한 문서이며 상장된 법인의 재무 정보를 크롤링해서 저장하는 코드입니다.

> 프로젝트를 성공적으로 마무리할 수 있기를 기원합니다!

개요

데이터 크롤링을 하는 스크립트는 python_workspace 디렉토리에 다음과 같은 형태로 구성되어 있습니다:



실행 순서는 다음과 같습니다:

1. ./web_crawling_Jongga.py (월요일부터 금요일까지 매일 5시에 실행)
2. ./web_crawling_general.py (6개월에 한 번씩 실행)
3. ./merge.py (월요일부터 금요일까지 매일 6시에 실행)

file_rw.py

이 스크립트는 파일 입출력을 할 때 깔끔하게 처리하기 위해 존재합니다. (사실 별 기능을 하지 않고 있기 때문에 제거하고 입출력 하는 부분은 파이썬에서 기본적으로 제공하는 함수로 대체하여 사용해도 무방합니다.)

▼ 기능

- JsonWrite(fn, data):

  fn 경로에 data를 json 형식으로 저장합니다.

- JsonRead(fn):

fn 경로에 있는 json 파일을 읽어옵니다.

- ExcelRead(fn):

  fn 경로에 있는 엑셀 파일을 읽어옵니다.

web_crawling_Jongga.py

web_crawling_Jongga.py 스크립트를 실행하면 "./data/오늘날짜" 폴더에 오늘 자 종가 데이터를 웹에서 가져와 jongga.json 파일로 저장합니다. (작명 센스 😂)

종가 데이터를 제공하지 않는 종목에 관해서는 따로 리스트에 저장하여 no_jongga.json 파일로 저장합니다. (사실 별로 중요하지 않은 파일이라 추후에 이 부분은 제거할까 합니다.)

▼ 필요 모듈:

- **urllib.request:** url을 통해 요청 → http 요청 가능 모듈 필요
- **BeautifulSoup:** beautifulsoup4 패키지 사용
- **html_table_parser:** html_table_parser 패키지 사용

▼ 코드 설명

```
def crawl(c_code):

    url = "http://www.thinkpool.com/itemanal/i/chart.jsp?code="
    url += str(c_code)
    html = urlopen(url)
    soup = BeautifulSoup(html, 'html.parser')

    table = soup.find("table", attrs={"class":"tbl1"})

    if type(table) == type(None):
        return -1

    trs = table.find_all('tr')

    for tr in trs:
        if "전일종가" in str(tr):
            tds = tr.find_all('td')
            jongga = tds[0].text.strip()

    return jongga
```

종가 데이터는 http://www.thinkpool.com/itemanal/i/chart.jsp?code=<기업코드> 사이트에서 크롤링해왔습니다.

urlopen 함수를 사용하여 원하는 주소로부터 웹페이지를 가져온 후, BeautifulSoup 객체로 변환합니다.

이 객체에서 html_table_parser을 사용해서 파싱한 후 필요한 데이터만을 가져와 리턴합니다.

```
jongmok_code = ExcelRead("./data/sangjang_jongmokCode.xlsx")
```

이 데이터는 http://comp.fnguide.com/SVO2/ASP/SVD_UJRank.asp?pGB=1&gicode=A005930&cID=&MenuYn=Y&ReportGB=&NewMenuID=301&stkGb=701 에서 다운로드 받아서 사용했습니다.

(업종별 순위[MKF500 | 연결 | 전체 | 최근결산월기준])

xls 파일로 제공하기 때문에 편의를 위해 xlsx 파일로 변환한 후 python의 xlrd 라이브러리를 사용해서 파싱했습니다.

```
jongga_data = {}
no_jongga = []

for i in jongmok_code.index:
```

```
    c_name = jongmok_code.iloc[i]["회사명"]
    jongga_data[c_name] = {}
    c_code = str(jongmok_code.iloc[i]["종목코드"])
    if len(c_code) < 6:
        ii = 6 - len(c_code)
        c_code = '0' * ii + str(c_code)
    jongga = crawl(c_code)
    if jongga == -1:
        no_jongga.append(c_name)
        del jongga_data[c_name]
    else:
        jongga = jongga.replace(',', '')
        jongga_data[c_name]["code"] = c_code
        jongga_data[c_name]["endPrice"] = jongga
```

서버에서 필요로 하는 형식으로 데이터 가공

```
now = datetime.datetime.now()
nowDate = now.strftime('%Y_%m_%d')
nds = str(nowDate)

fn1 = './data/' + str(nds) + '/jongga.json'
fn2 = './data/' + str(nds) + '/no_jongga.json'

dn = './data/' + str(nds)
if os.path.isdir(dn):
    shutil.rmtree(dn)
os.mkdir(dn)

#json 파일로 저장
JsonWrite(fn1, jongga_data)
JsonWrite(fn2, no_jongga)
```

가공한 데이터를 오늘자 날짜로 저장합니다.

cron을 사용하면 이 스크립트를 원하는 시간에 자동으로 반복적으로 실행시킬 수 있습니다.

다음 명령어를 터미널에서 crontab -e를 치면 열리는 스크립트에 저장해 놓으면 월요일부터 금요일까지 매일 17시에 스크립트를 실행합니다.

```
1 0 17 * * 1-5 python python_workspace/web_crawling_Jongga.py
```

설명이 잘 나와 있는 사이트를 첨부합니다

https://zetawiki.com/wiki/리눅스_반복_예약작업_cron,_crond,_crontab

web_crawling_general.py

web_crawling_general.py 스크립트를 실행하면 "./data_/오늘날짜" 폴더에 필요한 기업 재무 데이터를 웹에서 가져와 jongga.json 파일로 저장합니다. (작명 센스 😂)

재무 데이터를 제공하지 않는 종목에 관해서는 따로 리스트에 저장하여 no_info.json 파일로 저장합니다.

▼ 필요 모듈:

web_crawling_Jongga.y 에 쓰인 패키지와 동일합니다.

▼ 코드 설명:

```
# 웹에서 필요한 데이터 크롤링 해오는 함수
def crawl(j_code, c_name):
    url = "http://comp.fnguide.com/SVO2/asp/SVD_Main.asp?pGB=1&gicode=A" + str(j_code) +"&cID=&MenuYn=Y&ReportGB=&NewMenuID=101"
    html = urlopen(url)
    bsObj = BeautifulSoup(html, "html.parser")
    # 웹사이트에서 필요한 부분을 가져오는 코드
    tags = bsObj.find_all("table", attrs={"class":"us_table_ty1 h_fix zigbg_no"})
```

```
    # 재무 정보를 제공을 안한 경우 --> [제외]
    if len(tags) < 5:
        return "no_info"
    tags = bsObj.find_all("table", attrs={"class":"us_table_ty1 h_fix zigbg_no"})[4]

    html_table = parser.make2d(tags)
    del html_table[0]

    # 예외처리: 데이터가 적게 제공된 경우
    if len(html_table[0]) < 9:
        for l in html_table:
            if l[0] == 'IFRS(연결)':
                l[0] = "desc"
            if l[0] == 'GAAP(연결)':
                l[0] = "desc"
    else:
        # 필요 없는 데이터 제거
        for l in html_table:
            if l[0] == 'IFRS(연결)':
                l[0] = "desc"
            if l[0] == 'GAAP(연결)':
                l[0] = "desc"
            for _ in range(4):
                del l[1]

    # 만약 추정치인 경우(그 분기의 데이터가 시기상 아직 미정인 경우)
    # 표에 이상하게 표시되므로 보기 좋게 수정
    ## 필수 코드 아님 미관용.
    for i in range(len(html_table[0])):
        if "(E)" in html_table[0][i]:
            html_table[0][i] = html_table[0][i][26:]
        if "(P)" in html_table[0][i]:
            html_table[0][i] = html_table[0][i][24:]

    # null 처리
    for l in html_table[1:]:
        for i in range(len(l)):
            if l[i] == '':
                l[i] = None

    df = pd.DataFrame(data=html_table[1:], index=range(0, len(html_table)-1), columns=html_table[0])
    df.name = c_name

    return df
```

웹에서 필요한 데이터를 크롤링 해오는 함수입니다.

데이터는 http://comp.fnguide.com/SVO2/asp/SVD_Main.asp?pGB=1&gicode=A<기업코드>&cID=&MenuYn=Y&ReportGB=&NewMenuID=101 사이트에서 크롤링 해왔습니다.

```
def crawl2(c_code):
    url = "http://comp.fnguide.com/SVO2/asp/SVD_Finance.asp?pGB=1&gicode=A" + str(c_code) + "&cID=&MenuYn=Y&ReportGB=&NewMenuID=
    html = urlopen(url)
    bsObj = BeautifulSoup(html, "html.parser")

    tables = bsObj.find_all("table", attrs={"class":"us_table_ty1 h_fix zigbg_no"})

    if len(tables) < 3:
        return "no_info"

    table = tables[2]
    html_table = parser.make2d(table)
    flag = True
    cnt = 0
    for row in html_table:
        if "이익잉여금" in row[0]:
            flag = False
            ri = cnt - 1
        # null 처리
        for i in range(len(row)):
            if row[i] == '':
                row[i] = None
        cnt += 1
    # 이익잉여금 정보 없으면 return
```

```
    if flag:
        return "no_info"

    df = pd.DataFrame(data=html_table[1:], index=range(0, len(html_table)-1), columns=html_table[0])
    d = df.columns[len(df.columns)-1]

    return str(df[d].iloc[ri])
```

이익잉여금 데이터는 위의 crawl 함수의 url에서 제공하지 않기 때문에 따로 다른 url에서 가져왔습니다.

해당 정보는 http://comp.fnguide.com/SVO2/asp/SVD_Finance.asp?pGB=1&gicode=A<기업코드>&cID=&MenuYn=Y&ReportGB=&NewMenuID=103&stkGb=701 에서 크롤링해왔습니다.

```
def dataProcess(df, c_code, tds):

    flag = True
    QuantDataTable = {}
    CompanyDetailTable = {}
    QuantDataTable["cmpName"] = df.name
    QuantDataTable["code"] = c_code
    CompanyDetailTable["code"] = c_code
    CompanyDetailTable["name"] = df.name
    tmp = df[df.columns[0]]

    if tds not in df.columns:
        return "no_info", "no_info"

    for i in range(len(df[tds])):
        # 2020/03 데이터가 있는지 확인
        if df[tds][i] != None:
            flag = False

        # 2020/03 데이터가 없으면 return
        if flag:
            return "no_info", "no_info"

        # 데이터 타입 처리
        if type(df[tds][i]) != type(None):
            if df[tds][i] != "완전잠식":
                if df[tds][i] != "N/A":
                    if ',' in df[tds][i]:
                        df[tds][i] = df[tds][i].replace(',', '')
                    df[tds][i] = float(df[tds][i])

        # QuantDataTable
        if 'PER' in tmp[i]:
            QuantDataTable['per'] = df[tds][i]
        elif 'PBR' in tmp[i]:
            QuantDataTable['pbr'] = df[tds][i]
        elif 'ROA' in tmp[i]:
            QuantDataTable['roa'] = df[tds][i]
        elif 'ROE' in tmp[i]:
            QuantDataTable['roe'] = df[tds][i]
        elif '부채비율' in tmp[i]:
            QuantDataTable['debtRatio'] = df[tds][i]
        elif '영업이익률' in tmp[i]:
            QuantDataTable['operatingProfitRatio'] = df[tds][i]
        elif '유보율' in tmp[i]:
            QuantDataTable['reserveRatio'] = df[tds][i]

        # CompanyDetailTable
        # 종가는 매일 갱신
        elif tmp[i] == '자산총계':
            CompanyDetailTable['totalAsset'] = df[tds][i]
        elif tmp[i] == '자본총계':
            CompanyDetailTable['totalEquity'] = df[tds][i]
        elif tmp[i] == '부채총계':
            CompanyDetailTable['totalDebt'] = df[tds][i]
        elif tmp[i] == '매출액':
            CompanyDetailTable['sales'] = df[tds][i]
        elif tmp[i] == '영업이익':
            CompanyDetailTable['operatingProfit'] = df[tds][i]
        elif tmp[i] == '당기순이익':
            CompanyDetailTable['netIncome'] = df[tds][i]
```

```
    return QuantDataTable, CompanyDetailTable
```

crawl, crawl2에서 가져온 정보를 서버에서 필요로 하는 형태로 가공하는 함수입니다.

```
def setTime():
    now = datetime.datetime.now()
    nowDate = now.strftime('%Y_%m_%d')
    nds = str(nowDate)
    nds += '_'
    nds_l = nds.split('_')
    y = int(nds_l[0])
    m = int(nds_l[1])
    if 3 < m <= 12:
        tds = str(y) + '/03'
    else:
        tds = str(y - 1) + '/12'

    return str(nds), str(tds)
```

파일/디렉토리 이름 세팅용 함수입니다.

general 데이터는 현재 날짜의 데이터를 가져오는 것이 아니라 주기적으로 3월 12월 등의 날짜에 업데이트 되므로 날짜를 조금 변경해서 세팅합니다.

```
QTable = []
CTable = []
no_info = []

nds, tds = setTime()

for i in jongmok_code.index:
  c_name = jongmok_code.iloc[i]["회사명"]
  c_code = str(jongmok_code.iloc[i]["종목코드"])
  market = jongmok_code.iloc[i]["업종"]
  desc = jongmok_code.iloc[i]["주요제품"]
  if len(c_code) < 6:
      ii = 6 - len(c_code)
      c_code = '0' * ii + str(c_code)
  df = crawl(c_code, c_name)
  if type(df) == type("no_info"):
      no_info.append(c_name)
  else:
      Qdata, Cdata = dataProcess(df, c_code, tds)
      if type(Qdata) == type("no_info"):
          no_info.append(c_name)
      else:
          Cdata["description"] = desc
          Cdata["market"] = market
          retainedEarnings = crawl2(c_code)
          if retainedEarnings == "no_info":
              no_info.append(c_name)
          else:
              Cdata["retainedEarnings"] = retainedEarnings
              QTable.append(Qdata)
              CTable.append(Cdata)
```

이제 위에서 만든 함수를 사용하여 실제로 데이터를 받아온 후,

```
fn1 = './data/' + str(nds) + '/QuantDataTable.json'
fn2 = './data/' + str(nds) + '/CompanyDetailTable.json'
fn3 = './data/' + str(nds) + '/no_info.json'

dn = './data/' + str(nds)
if os.path.isdir(dn):
    shutil.rmtree(dn)
```

```
os.mkdir(dn)

#json 파일로 저장
JsonWrite(fn1, QTable)
JsonWrite(fn2, CTable)
JsonWrite(fn3, no_info)
```

가공한 데이터를 오늘자 날짜로 저장합니다

다음 명령어를 터미널에서 crontab -e를 치면 열리는 스크립트에 저장해 놓으면 6개월에 한번 18시에 스크립트를 실행합니다.

```
0 18 1 6 * python python_workspace/web_crawling_general.py
```

merge.py

merge.py 스크립트를 실행하면 ./data/final/오늘날짜 디렉토리에 CompanyDetailTable.json, QuantDataTable.json, no_info데이터를 저장합니다.

종가 데이터는 매일 갱신되고 나머지 데이터는 6개월에 한 번씩 갱신되므로 따로 실행시켜 매일 다시 합쳐서 오늘자 날짜로 디렉토리에 저장해놓습니다.

단순히 데이터를 합쳐서 저장해놓는 코드이므로 코드 설명은 생략하겠습니다.

다음 명령어를 터미널에서 crontab -e를 치면 열리는 스크립트에 저장해 놓으면 월요일부터 금요일까지 매일 17시에 스크립트를 실행합니다.

```
0 18 * * 1-5 python python_workspace/merger.py
```