



암호화폐 트레이딩 교육

Day 1 (2/2)

웹 스크래핑

Kangwuk Heo
calvin.heo@gmail.com

Contents

웹스크래핑 (WEB SCRAPING)	3
SECTION 1: 웹스크래핑 실습	4
SECTION 2: RESTFULAPI 호출 방식	9
SECTION 3: PANDAS	11

Overview

암호화폐 트레이딩 교육 과정에서, 실습을 통해서, 트레이딩이 무엇인지, 트레이딩을 어떤 방식으로 구성하고 진행하는지를 배울 수 있습니다.

웹스크래핑 (Web Scraping)

Section 1 은 웹스크래핑은 무엇인지, 그리고 웹스크래핑과 웹크롤링의 차이점, 실습을 수행하게 됩니다.

Section 1: 웹스크래핑 실습

웹스크래핑은 웹페이지에서 원하는 정보를 가져오는 것입니다.

BeautifulSoup 모듈을 통해서 HTML 파일에서 원하는 데이터를 파싱하는데 사용하며, html5lib 모듈은 HTML 데이터와 HTML 문서를 파싱하는데 활용합니다.

__1. 웹페이지 URL 정보를 기반으로, HTML 정보를 가져와서 PER 값을 출력해 봅니다.

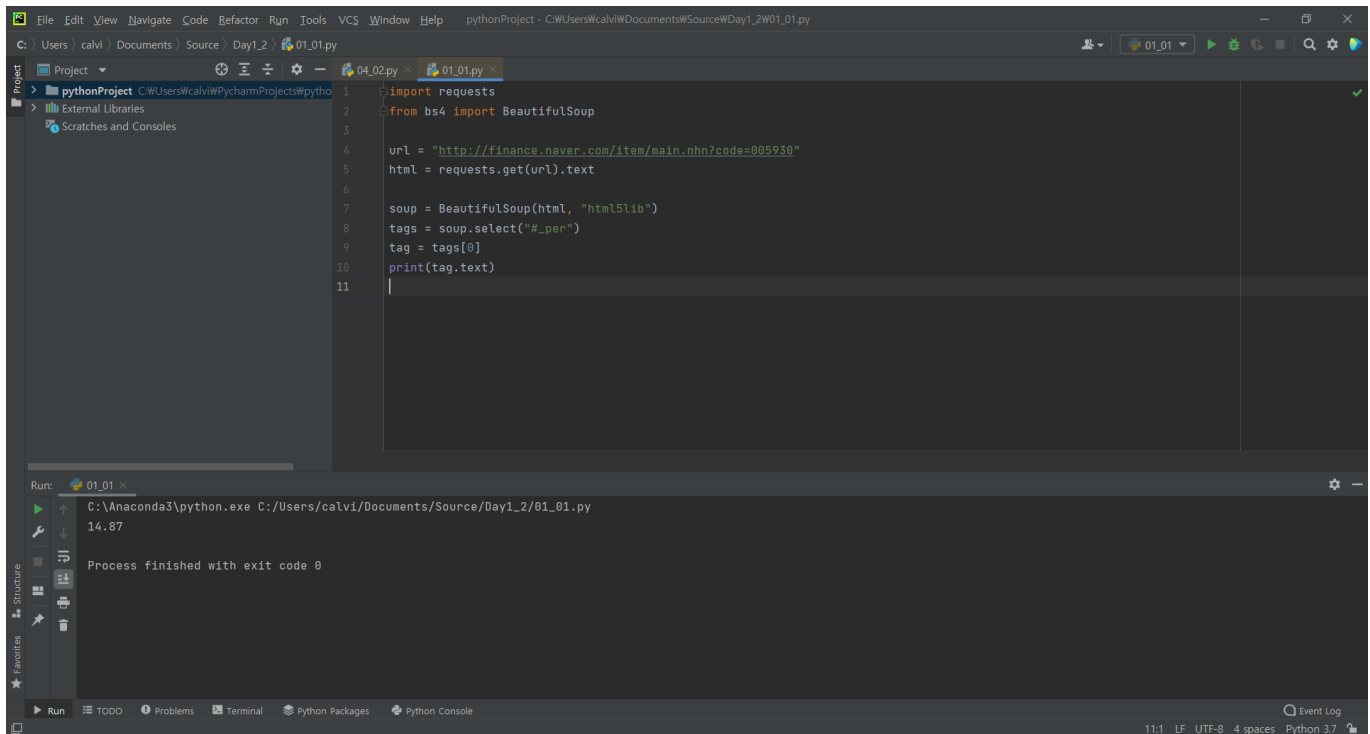
삼성전자: 네이버 금융 (naver.com)

```
import requests
from bs4 import BeautifulSoup

url = "http://finance.naver.com/item/main.nhn?code=005930"
html = requests.get(url).text

soup = BeautifulSoup(html, "html5lib")
tags = soup.select("#_per")
tag = tags[0]
print(tag.text)
```

파이썬 로직을 실행하게 되면, 해당정보를 가져오게 됩니다.



__2. 웹페이지에서 한영역을 선택 후, 마우스 오른쪽 버튼을 통해서 보여지는 메뉴에서 검사를 실행합니다. 검사 기능이 실행되면, 웹브라우저에서 보여지는 웹페이지에 대한 HTML 페이지가 보여지게 됩니다.

The screenshot shows the Naver Finance page for Samsung Electronics (삼성전자) with the following data:

- Current Price:** 70,600 (Change: +400, +0.57%)
- Previous Close:** 70,200
- High:** 70,600 (Range: 91,700)
- Low:** 69,800 (Range: 49,500)
- Volume:** 10,945,362
- Market Cap:** 768,110 billion KRW
- PER (2021.06):** 14.87x
- BPS (2021.06):** 4,749 KRW
- EPS (2021.06):** 5,841 KRW
- Dividend Yield:** 4.24%
- Dividend Date:** 2020.12
- Dividend Yield (PER):** 13.09%
- Dividend Yield (BPS):** +0.34%

The browser developer tool (F12) is open, showing the DOM structure and CSS styles. The DOM tree highlights the table containing the PER and EPS information. The CSS styles section shows the default styles for the table and its components.

__3. 웹페이지 상에서 PER 값과 배당수익률을 스크래핑하는 로직을 실행해 봅니다.

```

import requests
from bs4 import BeautifulSoup

def get_per(code):
    url = "http://finance.naver.com/item/main.nhn?code=" + code
    html = requests.get(url).text

    soup = BeautifulSoup(html, "html5lib")
    tags = soup.select("#_per")
    tag = tags[0]
    return float(tag.text)

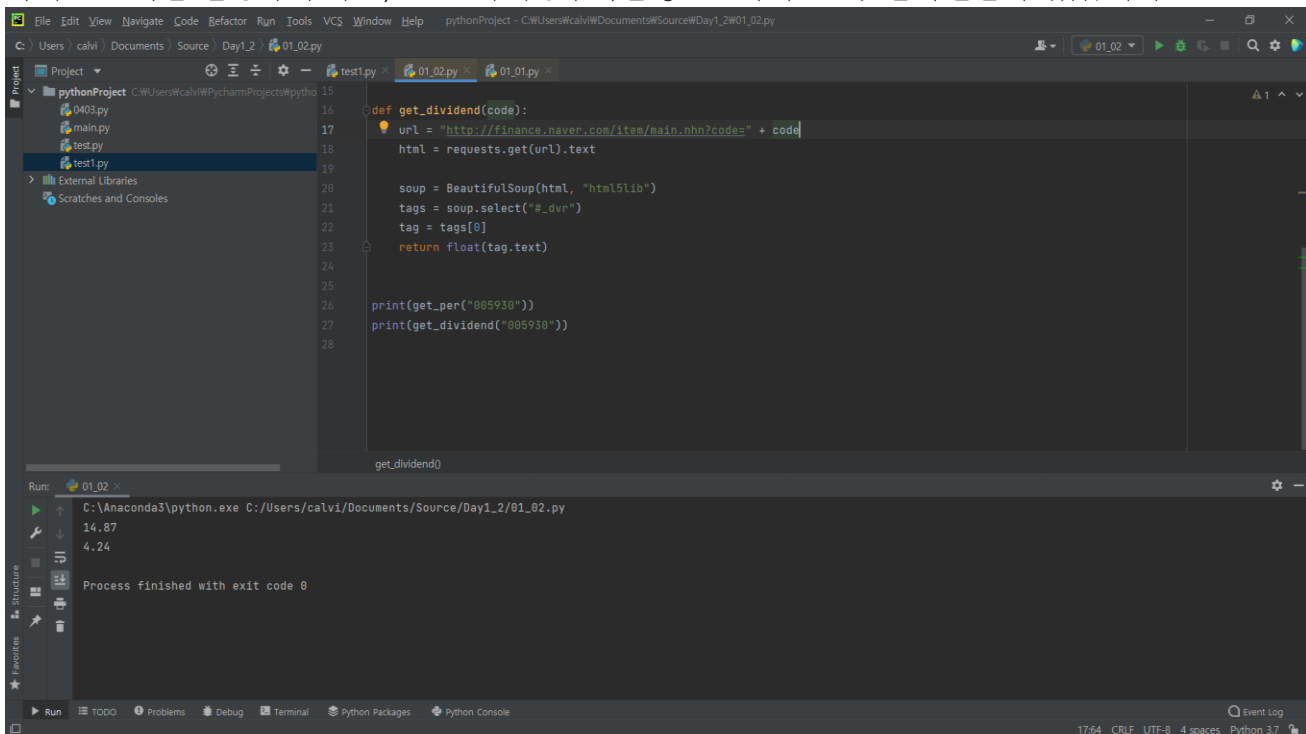
def get_dividend(code):
    url = "http://finance.naver.com/item/main.nhn?code=" + code
    html = requests.get(url).text

    soup = BeautifulSoup(html, "html5lib")
    tags = soup.select("#_dvr")
    tag = tags[0]
    return float(tag.text)

print(get_per("005930"))
print(get_dividend("005930"))

```

파이썬 로직을 실행하게 되면, PER 과 배당수익률정보를 가져오는 것을 확인할 수 있습니다.



__4. 웹페이지 상에서 외국인소진율을 스크래핑하는 로직을 실행해 봅니다.

```

```

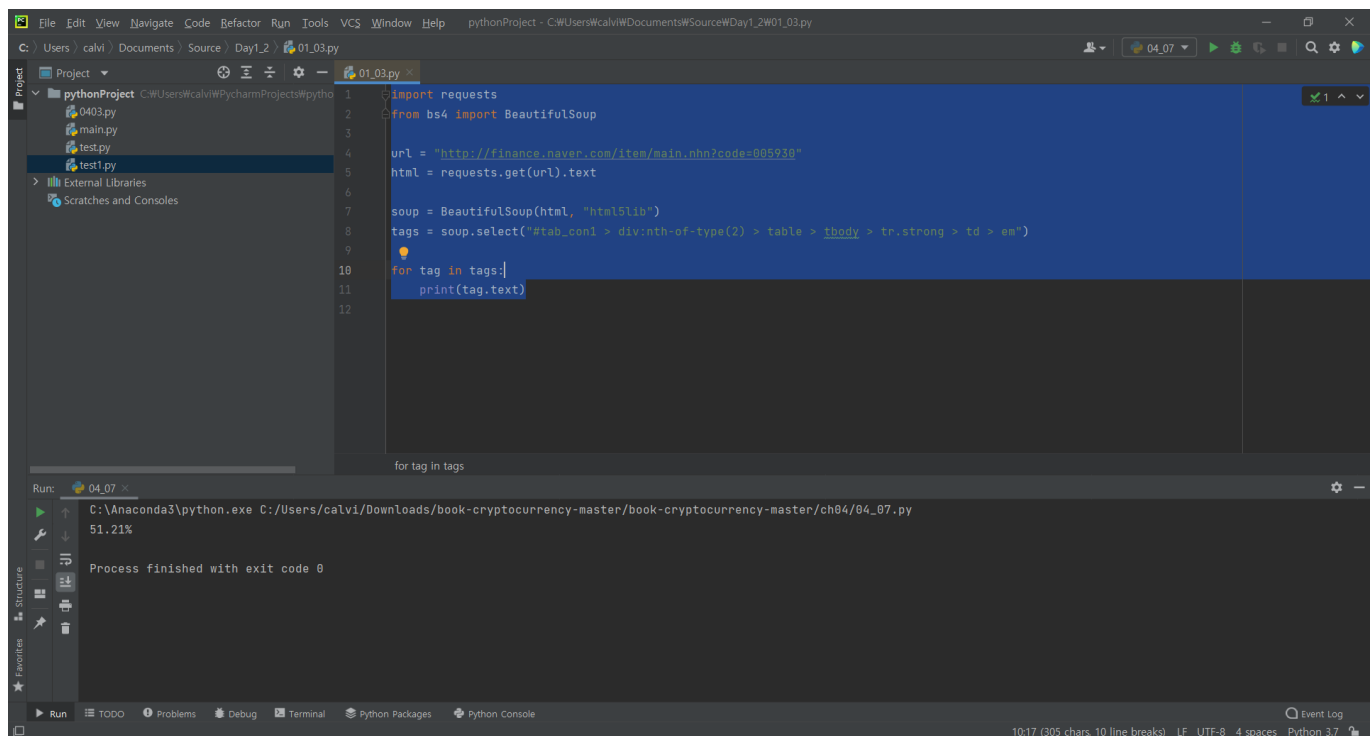
```
import requests
from bs4 import BeautifulSoup

url = "http://finance.naver.com/item/main.nhn?code=005930"
html = requests.get(url).text

soup = BeautifulSoup(html, "html5lib")
tags = soup.select("#tab_con1 > div:nth-of-type(2) > table > tbody >
tr.strong > td > em")

for tag in tags:
    print(tag.text)
```

로직을 실행하게 되면, 외국인소진율 정보를 가져오는 것을 확인할 수 있습니다.



Section 2: RestfulAPI 호출 방식

웹페이지에서 데이터를 얻어오는 방법에는 웹 API 를 사용하는 방법이 있습니다.

1. 거래소에서 제공하는 API 중에 암호화폐 시세정보를 가져오는 API 를 웹브라우저에서 실행해 봅니다.

https://api.korbit.co.kr/v1/ticker/detailed?currency_pair=btc_krw

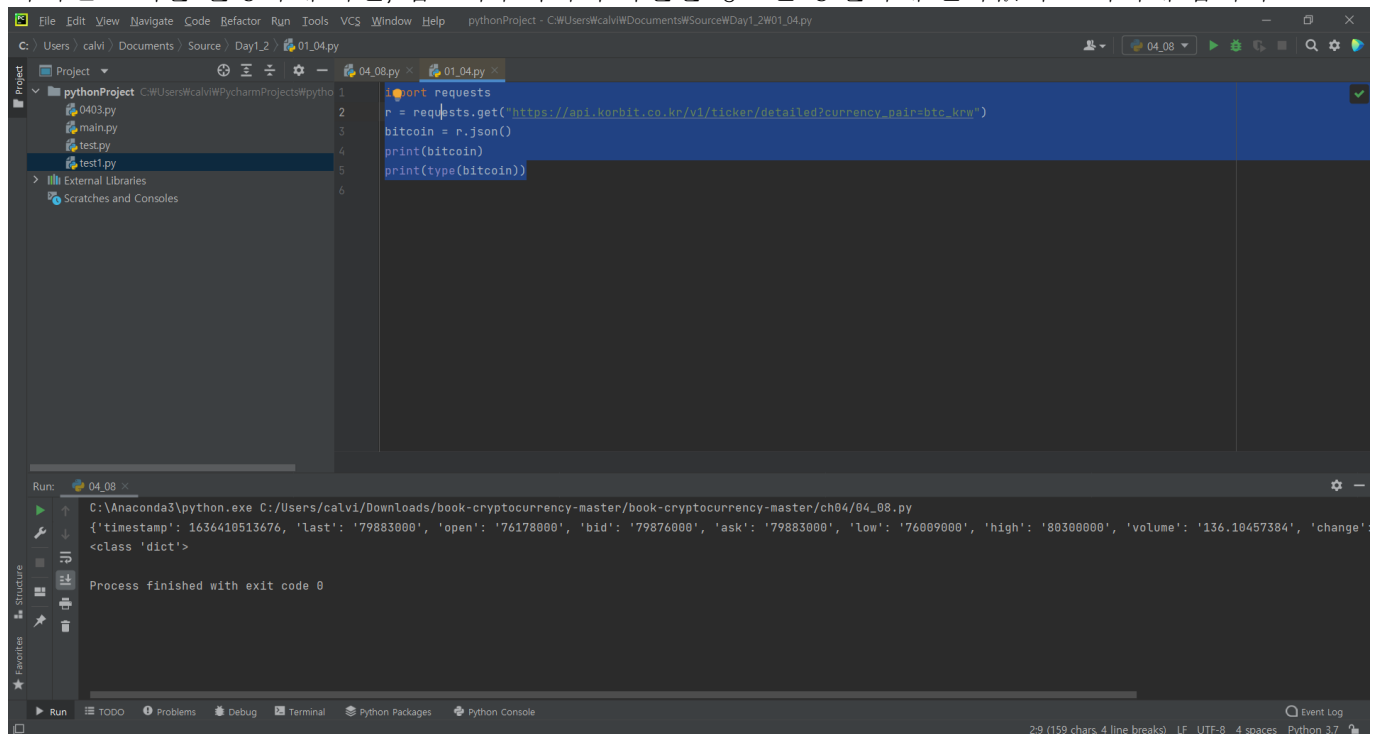
API 를 통해서 가져오는 정보는 현재 시세 정보가 보이게 됩니다.

```
{"timestamp":1636410240386,"last":"79876000","open":"76199000","bid":"79876000","ask":"79882000","low":"76009000","high":"80300000","volume":"136.02506656","change":"3677000","changePercent":"4.83"}
```

2. 웹브라우저에서 보여지는 시세정보를 파이썬 로직에서 확인해 봅니다.

```
import requests
r = requests.get("https://api.korbit.co.kr/v1/ticker/detailed?currency_pair=btc_krw")
bitcoin = r.json()
print(bitcoin)
print(type(bitcoin))
```

파이썬 로직을 실행하게 되면, 웹브라우저에서 확인한 정보를 동일하게 결과값이 보여지게 됩니다.



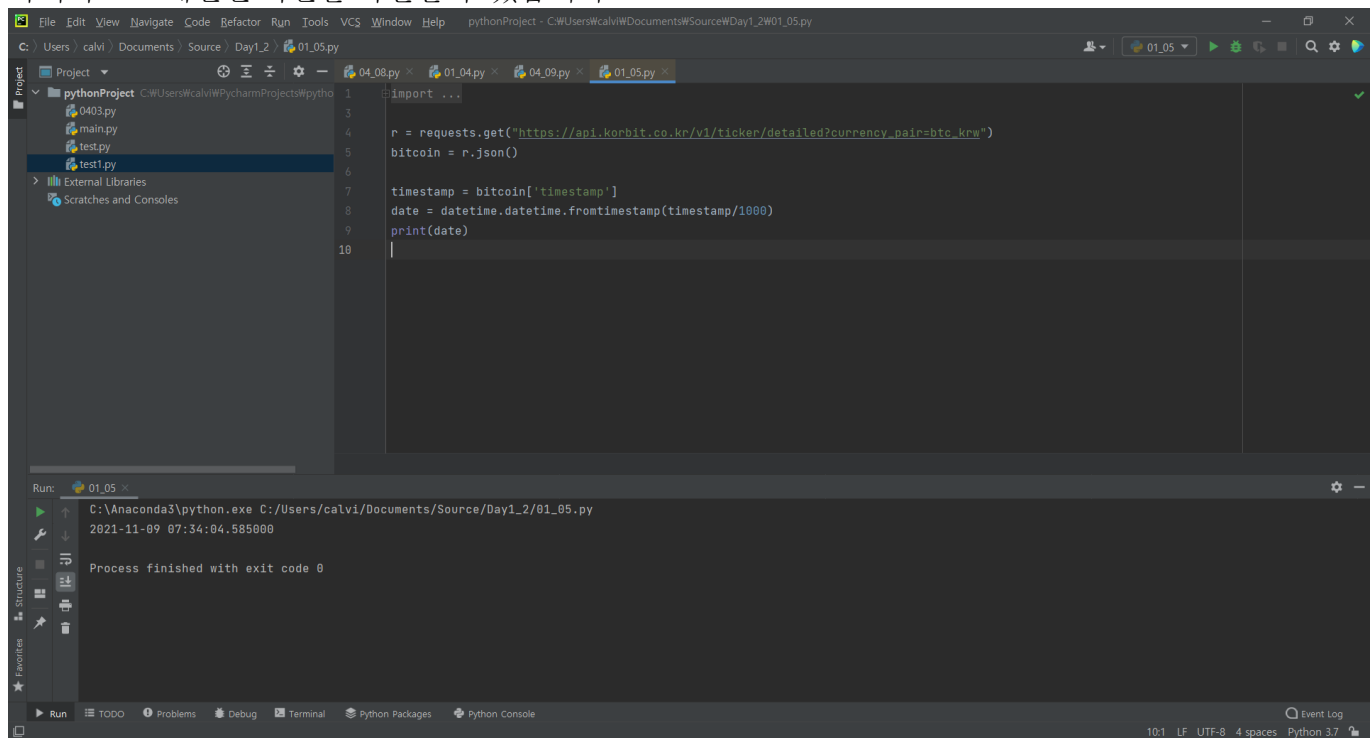
__3. 시세정보중, 마지막으로 체결된 시간을 가져오는 로직을 실행해 봅니다.

```
import requests
import datetime

r =
requests.get("https://api.korbit.co.kr/v1/ticker/detailed?currency_pair=btc_krw")
bitcoin = r.json()

timestamp = bitcoin['timestamp']
date = datetime.datetime.fromtimestamp(timestamp/1000)
print(date)
```

마지막으로 체결된 시간을 확인할 수 있습니다.



Section 3: Pandas

Pandas 는 모듈형태로 구성되어 있으며, 1 차원 데이터와 2 차원 데이터 구조로 구성되어 있습니다.

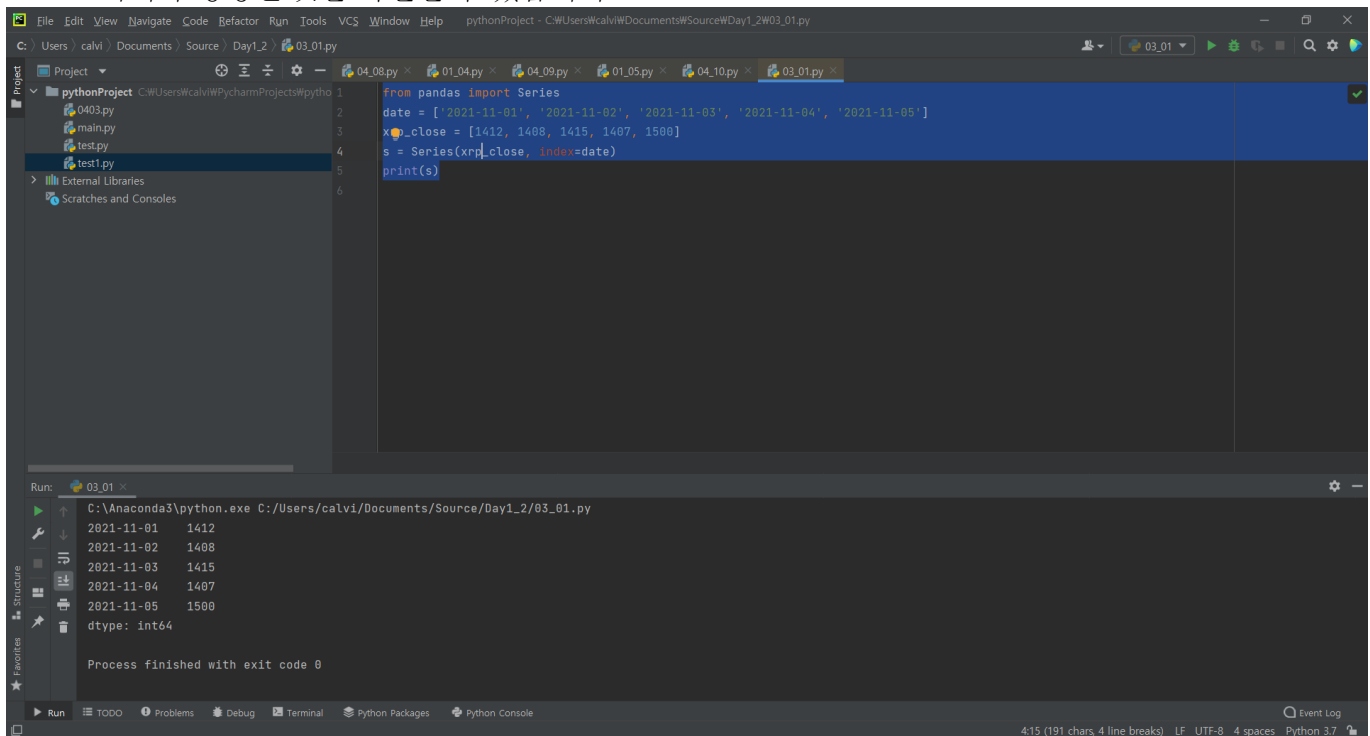
Pandas 모듈을 사용하기 위해서는 아래와 같은 import 선언문을 정의해야 합니다.

```
from pandas import Series
```

__1. Pandas 를 기반으로 Series 라는 객체를 생성합니다.

```
from pandas import Series
date = ['2021-11-01', '2021-11-02', '2021-11-03', '2021-11-04', '2021-11-05']
xrp_close = [1412, 1408, 1415, 1407, 1500]
s = Series(xrp_close, index=date)
print(s)
s.datetime.fromtimestamp(timestamp/1000)
print(date)
```

Series 객체가 생성된 것을 확인할 수 있습니다.



```
File Edit View Navigate Code Refactor Run Tools VCS Window Help pythonProject - C:\Users\calvi\Documents\Source\Day1_2\03_01.py
C:\Users\calvi\Documents\Source\Day1_2\03_01.py
Project
  pythonProject C:\Users\calvi\PycharmProjects\pythonProject
    0403.py
    main.py
    test.py
    test1.py
  External Libraries
  Scratches and Consoles
Run: 03_01
C:\Anaconda3\python.exe C:/Users/calvi/Documents/Source/Day1_2/03_01.py
2021-11-01    1412
2021-11-02    1408
2021-11-03    1415
2021-11-04    1407
2021-11-05    1500
dtype: int64
Process finished with exit code 0
4:15 (191 chars, 4 line breaks) LF UTF-8 4 spaces Python 3.7
```

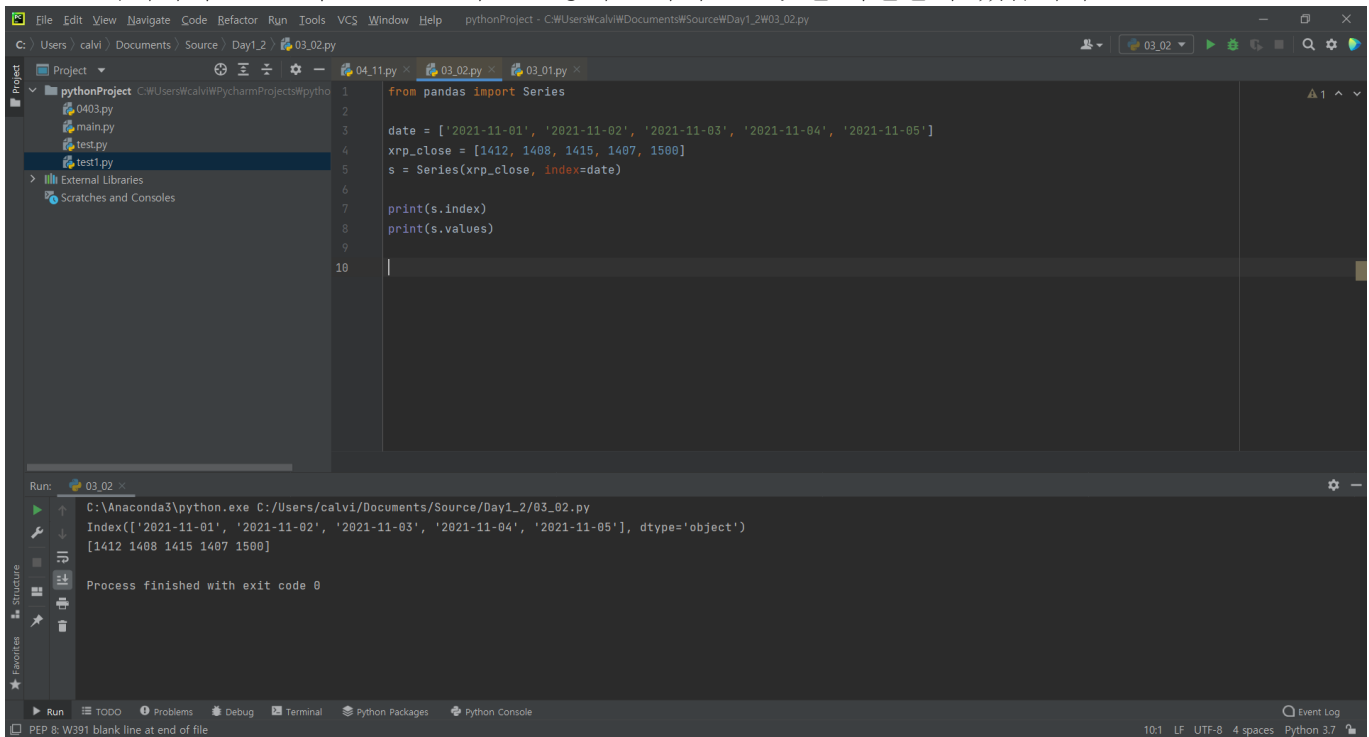
__2. 생성된 Series 객체에서 index 와 Value 속성 값을 가져옵니다.

```
from pandas import Series

date = ['2021-11-01', '2021-11-02', '2021-11-03', '2021-11-04', '2021-11-05']
xrp_close = [1412, 1408, 1415, 1407, 1500]
s = Series(xrp_close, index=date)

print(s.index)
print(s.values)
```

Series 객체에서 Index 와 Value 를 리스트 형태로 가져오는 것을 확인할 수 있습니다.



```
File Edit View Navigate Code Refactor Run Tools VCS Window Help pythonProject - C:\Users\calvi\Documents\Source\Day1_2\03_02.py
C:\Users\calvi\Documents\Source\Day1_2\03_02.py
Project
pythonProject C:\Users\calvi\PycharmProjects\pythonProject
0403.py
main.py
test.py
test1.py
External Libraries
Scratches and Consoles
Run: 03_02.py
C:\Anaconda3\python.exe C:\Users\calvi\Documents\Source\Day1_2\03_02.py
Index(['2021-11-01', '2021-11-02', '2021-11-03', '2021-11-04', '2021-11-05'], dtype='object')
[1412 1408 1415 1407 1500]
Process finished with exit code 0
PEP 8: W391 blank line at end of file 10:1 LF UTF-8 4 spaces Python 3.7
```

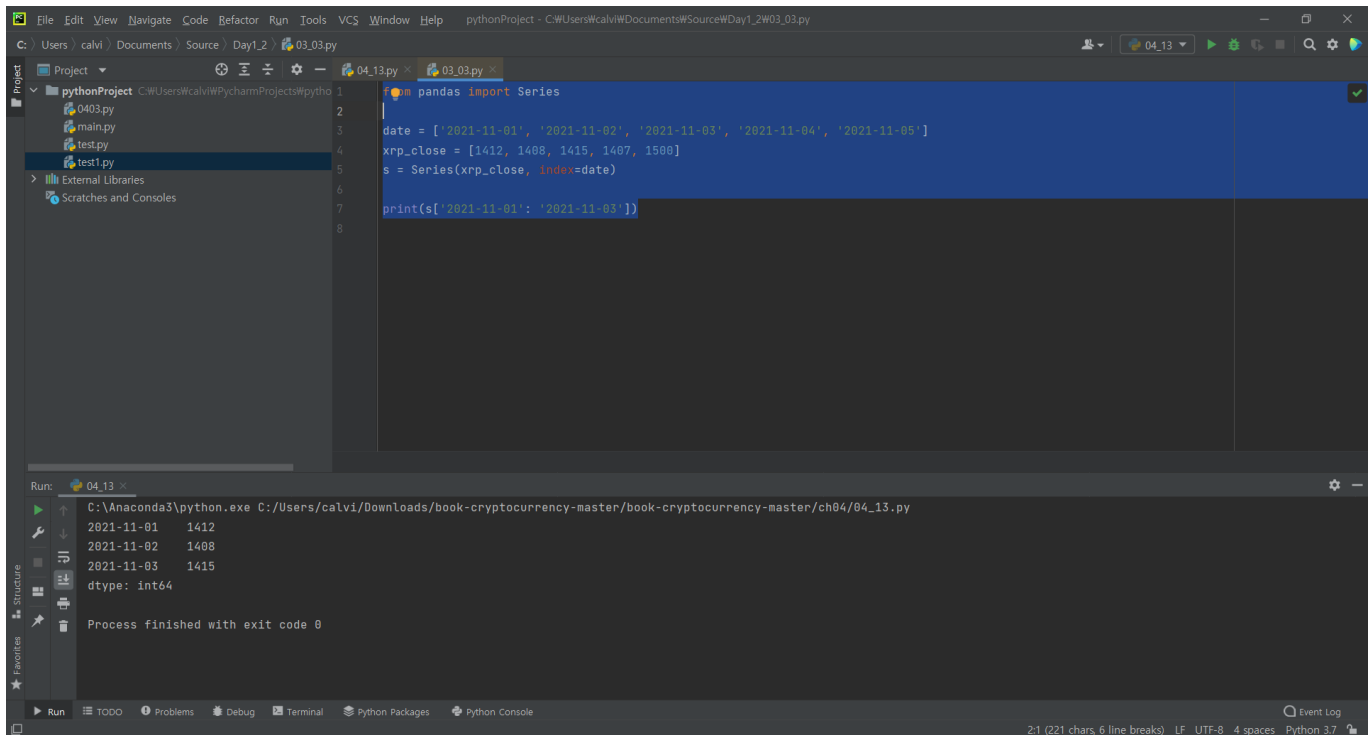
__3. 생성된 Series 객체에서 기간 기준으로 데이터를 슬라이싱 해봅니다.

```
from pandas import Series

date = ['2021-11-01', '2021-11-02', '2021-11-03', '2021-11-04', '2021-11-05']
xrp_close = [1412, 1408, 1415, 1407, 1500]
s = Series(xrp_close, index=date)

print(s['2021-11-01': '2021-11-03'])
```

기간 기준으로 데이터를 가져오는 것을 확인할 수 있습니다.



The screenshot shows an IDE window with a Python script and its execution output. The script defines a date range and a series of closing prices for XRP. The output shows the resulting series with dates as the index and closing prices as the values.

```
1 from pandas import Series
2
3 date = ['2021-11-01', '2021-11-02', '2021-11-03', '2021-11-04', '2021-11-05']
4 xrp_close = [1412, 1408, 1415, 1407, 1500]
5 s = Series(xrp_close, index=date)
6
7 print(s['2021-11-01': '2021-11-03'])
8
```

Run: 04_13 x

```
C:\Anaconda3\python.exe C:/Users/calvi/Downloads/book-cryptocurrency-master/book-cryptocurrency-master/ch04/04_13.py
2021-11-01    1412
2021-11-02    1408
2021-11-03    1415
dtype: int64
Process finished with exit code 0
```

__4. Pandas 의 2 차원 데이터인 DataFrame 기반으로 데이터를 처리해 봅니다.

```
from pandas import DataFrame

data = {'open': [400, 500], "high": [410, 510]}
df = DataFrame(data)
print(df)
```

DataFrame 객체 기반 데이터가 저장된 것을 확인할 수 있습니다.

The screenshot shows the PyCharm IDE interface. The main editor window displays a Python script named `03_04.py` with the following code:

```
1 from pandas import DataFrame
2
3 data = {'open': [400, 500], 'high': [410, 510]}
4 df = DataFrame(data_)
5 print(df)
6
```

The Run window at the bottom shows the output of the script:

```
Run: 04_16 x
C:\Anaconda3\python.exe C:/Users/calvi/Downloads/book-cryptocurrency-master/book-cryptocurrency-master/ch04/04_16.py
open high
0 400 410
1 500 510

Process finished with exit code 0
```

__5. DataFrame 객체에 데이터를 excel 파일을 로드해서 생성해 봅니다..

```
import pandas as pd
df = pd.read_excel("test.xlsx")
df = df.set_index('date')
print(df)
df.to_excel("test-2.xlsx")
```

엑셀 파일에 있는 데이터를 로드하고, 새로운 엑셀파일을 생성된 것을 확인할 수 있습니다.

The screenshot shows the PyCharm IDE interface. The main editor window displays a Python script named `03_05.py` with the following code:

```
1 import pandas as pd
2 df = pd.read_excel("test.xlsx")
3 df = df.set_index('date')
4 print(df)
5 df.to_excel("test-2.xlsx")
6
```

The Run window at the bottom shows the output of the script:

```
Run: 03_05 x
date open high low volume
2021-11-01 200 220 150 200
2021-11-02 200 300 180 300
2021-01-03 300 330 300 320
2021-11-04 320 340 320 330
2021-11-05 330 350 320 325
2021-01-06 325 360 300 335

Process finished with exit code 0
```

__6. DataFrame 객체에 데이터를 excel 파일을 로드해서 생성해 봅니다. .

```
import pandas as pd
df = pd.read_excel("test.xlsx")
df = df.set_index('date')
print(df)
df.to_excel("test-2.xlsx")
```