Out[1]:

Click here to toggle on/off the raw code.

ㅁOne Point Tutorial III - TIMESERIES

Python을 활용한 데이터 사이언스

---

0. 목차소개

1. PSL의 datetime 모듈에 대하여
2. Time Series Data Handling
3. Period
4. Resampling
5. TimeSeries 그리기

목표 : 시계열 데이터를 다루는 데에 있어서 필요한 기본기들을 익히도록 하자

1. PSL의 datetime 모듈에 대하여

파이썬에는 기본라이브러리(Python Standard Library)에 날짜와 시간을 담당하는 모듈이 있다. *datetime.py*

1.1 datetime 객체의 주요 attribute

```
datetime.year

datetime.month

datetime.day

datetime.hour

datetime.second

datetime.microsecond
```

1.2 datetime 객체의 생성 방법

Out[2]:

```
datetime.datetime(2019, 8, 30, 18, 10, 3, 904273)
```

표현식에서는 second와 microsecond가 0일 경우에 표시하지 않는다.

Out[3]:

```
datetime.datetime(2019, 8, 27, 0, 0)
```

1.3 datetime 객체의 산술 연산

datetime간의 - 연산

Out[4]:

```
datetime.timedelta(days=3, seconds=65404, microseconds=744619)
```

datetime과 timedelta간의 산술 연산

Out[7]:

```
datetime.datetime(2019, 1, 13, 0, 0)
```

Out[8]:

```
datetime.datetime(2019, 1, 25, 0, 0)
```

Out[9]:

```
datetime.datetime(2018, 12, 20, 0, 0)
```

1.4 string, datetime, pd.timestamp 객체 간의 변환

외부에서 긁어모은 데이터를 원하는 형식으로 나타내고자 할 때 변환을 많이 한다.

1.4.1 str ==> datetime

문자열에서 시간에 관련한 정보가 어떻게 저장되어있는 지 포맷을 전달하면 된다.

Out[10]:

```
datetime.datetime(2019, 8, 1, 0, 0)
```

Out[11]:

```
datetime.datetime(2019, 8, 1, 0, 0)
```

Out[12]:

```
datetime.datetime(2019, 8, 1, 0, 0)
```

1.4.2 datetime ==> str

Out[13]:

```
'2019-08-01'
```

1.4.3 str ==> timestamp

Out[14]:

```
DatetimeIndex(['2019-08-01', '2019-08-02'], dtype='datetime64[ns]', freq=N
one)
```

2. Time Series Data Handling

2.1 Time Series 객체를 생성하는 방법

pd.Series의 parameter **index**에 시간을 넣어주면 시계열 데이터이다.

index에 시간을 넣어주는 방법

> 1. datetime을 원소로 갖는 iterable한 객체
> 2. pd.DatetimeIndex 객체

Out[17]:

```
2019-08-01    0
2019-08-03    1
2019-08-08    2
2019-09-11    3
2019-10-01    4
dtype: int32
```

Out[18]:

```
DatetimeIndex(['2019-08-01', '2019-08-03', '2019-08-08', '2019-09-11',
               '2019-10-01'],
              dtype='datetime64[ns]', freq=None)
```

Out[20]:

```
2019-08-01    0
2019-08-03    1
2019-08-08    2
2019-09-11    3
2019-10-01    4
dtype: int32
```

2.2 TimeSeries의 Index를 만드는 방법

위에서 우리는 모든 것을 날짜를 하여서 Index를 만들었지만, **특정한 규칙**을 통해서 만들고 싶은 경우가 많다.

이 때, `pd.date_range` 라는 함수를 사용한다.

freq에 들어갈 수 있는 argument들

> D : 매일
> A : 매년
> B : 매근무일
> T : 매초
> M : 매월말
> MS : 매월초
> BM : 매근무월말
> BMS : 매근무월초
> W-MON : 매주월요일
> W-1MON : 매월;첫월요일
> Q-JAN : 매분기;1월말기준
> QS-JAN : 매분기;1월초기준
> BQ-JAN : 매분기;근무1월말기준
> BQS-JAN : 매분기;근무1월초기준

Out[21]:

```
DatetimeIndex(['2000-01-01', '2000-01-02', '2000-01-03', '2000-01-04',
               '2000-01-05', '2000-01-06', '2000-01-07', '2000-01-08',
               '2000-01-09', '2000-01-10'],
              dtype='datetime64[ns]', freq='D')
```

Out[70]:

```
DatetimeIndex(['2000-01-31', '2000-02-29', '2000-03-31', '2000-04-30',
               '2000-05-31', '2000-06-30', '2000-07-31', '2000-08-31',
               '2000-09-30', '2000-10-31'],
              dtype='datetime64[ns]', freq='M')
```

Out[72]:

```
2000-01-31    0
2000-02-29    1
2000-03-31    2
2000-04-30    3
2000-05-31    4
2000-06-30    5
2000-07-31    6
2000-08-31    7
2000-09-30    8
2000-10-31    9
Freq: M, dtype: int32
```

시계열 데이터에서 lagging은 **freq**기반으로 움직인다.

Out[25]:

```
2000-01-31    NaN
2000-02-29    NaN
2000-03-31    0.0
2000-04-30    1.0
2000-05-31    2.0
2000-06-30    3.0
2000-07-31    4.0
2000-08-31    5.0
2000-09-30    6.0
2000-10-31    7.0
Freq: M, dtype: float64
```

Out[26]:

```
2000-03-31    0
2000-04-30    1
2000-05-31    2
2000-06-30    3
2000-07-31    4
2000-08-31    5
2000-09-30    6
2000-10-31    7
2000-11-30    8
2000-12-31    9
Freq: M, dtype: int32
```

2.3 Time Series 데이터를 indexing&slicing을 통해 접근&변경하는 법

Out[76]:

```
Timestamp('2000-02-29 00:00:00', freq='M')
```

index를 그대로 입력하는 것은 어려운 일이지만 아래처럼 다양한 형태로 입력을 하여도 찾아준다.

Out[77]:

```
0
```

Out[78]:

```
2000-01-31    0
2000-02-29    1
2000-03-31    2
2000-04-30    3
2000-05-31    4
2000-06-30    5
2000-07-31    6
2000-08-31    7
2000-09-30    8
2000-10-31    9
Freq: M, dtype: int32
```

Out[82]:

```
2000-01-31      0
2000-02-29    100
2000-03-31      2
2000-04-30      3
2000-05-31      4
2000-06-30      5
2000-07-31      6
2000-08-31      7
2000-09-30      8
2000-10-31      9
Freq: M, dtype: int32
```

Out[79]:

```
2000-01-31    0
Freq: M, dtype: int32
```

Out[83]:

```
2000-01-31      0
2000-02-29    100
2000-03-31      2
Freq: M, dtype: int32
```

3. Period

3.1 Period란?

우리가 지금까지 다뤘던 pd.DatetimeIndex의 원소인 Timestamp는 **point**라면 pd.Periods는 **구간**이다

3.2 Period 객체의 주요 attribute

`Period.day`

> 이 구간의 끝점의 일자이다.

`Period.dayofweek`

> 이 구간의 끝점의 요일이다.

`Period.starttime`

> 이 구간의 시작점이다.

`Period.endtime`

> 이 구간의 끝점이다.

3.3 Period 객체를 생성하는 방법

2007년에서 끝점(`OCT`)이 10월 31일인 1년짜리(`A-`) 구간을 생성하고 싶다면

Out[30]:

```
Period('2007', 'A-OCT')
```

Out[31]:

```
Timestamp('2006-11-01 00:00:00')
```

Out[32]:

```
Timestamp('2007-10-31 23:59:59.999999999')
```

3.4 Period 객체들을 생성하고 시계열 데이터를 만들기

끝점(`2019Q4`)을 12월 말(`DEC`)로 뒀을 때 시작점(`2018Q1`)부터 분기(`Q-`)마다 구간을 나눠서 range를 만들고!

Out[33]:

```
PeriodIndex(['2018Q1', '2018Q2', '2018Q3', '2018Q4', '2019Q1', '2019Q2',
             '2019Q3', '2019Q4'],
            dtype='period[Q-DEC]', freq='Q-DEC')
```

이를 이용하여 시계열 데이터를 만들어보자.

Out[35]:

```
2018Q1    0
2018Q2    1
2018Q3    2
2018Q4    3
2019Q1    4
2019Q2    5
2019Q3    6
2019Q4    7
Freq: Q-DEC, dtype: int32
```

Out[36]:

```
PeriodIndex(['2018Q1', '2018Q2', '2018Q3', '2018Q4', '2019Q1', '2019Q2',
             '2019Q3', '2019Q4'],
            dtype='period[Q-DEC]', freq='Q-DEC')
```

4. Resampling

우리는 특정한 기준을 가지고 주어진 시계열 데이터셋을 줄이거나 늘리고 싶을 때가 있다.

줄이는 행위를 **downsampling**이라하고 늘리는 행위를 **upsampling** 이라고 한다.

4.1 Downsampling

Downsampling의 핵심 키워드는 **aggregating**이다.

Out[39]:

```
2019-01-01 00:00:00     0
2019-01-01 00:01:00     1
2019-01-01 00:02:00     2
2019-01-01 00:03:00     3
2019-01-01 00:04:00     4
2019-01-01 00:05:00     5
2019-01-01 00:06:00     6
2019-01-01 00:07:00     7
2019-01-01 00:08:00     8
2019-01-01 00:09:00     9
2019-01-01 00:10:00    10
2019-01-01 00:11:00    11
2019-01-01 00:12:00    12
2019-01-01 00:13:00    13
Freq: T, dtype: int32
```

Out[40]:

```
2019-01-01 00:00:00    10
2019-01-01 00:05:00    35
2019-01-01 00:10:00    46
Freq: 5T, dtype: int32
```

Out[41]:

```
2019-01-01 00:50:00    10
2019-01-01 00:55:00    35
2019-01-01 01:00:00    46
Freq: 5T, dtype: int32
```

Out[42]:

	open	high	low	close
2019-01-01 00:00:00	0	4	0	4
2019-01-01 00:05:00	5	9	5	9
2019-01-01 00:10:00	10	13	10	13

4.2 Upsampling

Upsamlpling하면 핵심 키워드는 **interpolation**이다. 더 채워 넣는다는 것은 자료들의 빈칸을 채워야하기 때문이다.

예를 들어, 주가의 경우 주말에 기록된 것이 없기 때문에 일자별로 Upsampling하게 되면 채워줘야한다.

Out[44]:

	Naver	Kakao
2019-08-01	100	90
2019-08-02	101	100
2019-08-05	103	110
2019-08-06	105	90

Out[45]:

	Naver	Kakao
2019-08-01	100	90
2019-08-02	101	100
2019-08-03	101	100
2019-08-04	101	100
2019-08-05	103	110
2019-08-06	105	90

Out[46]:

	Naver	Kakao
2019-08-01	100.0	90.0
2019-08-02	101.0	100.0
2019-08-03	101.0	100.0
2019-08-04	NaN	NaN
2019-08-05	103.0	110.0
2019-08-06	105.0	90.0

4.3 Resampling with Periods

시계열 데이터의 index가 timestamp가 아닌 period인 경우에 resampling을 어떻게 할까?

```
  File "<ipython-input-47-b484bddfbe88>", line 1
    frame = pd.DataFrame(np.)
                             ^
SyntaxError: invalid syntax
```

Out[48]:

	Naver	Kakao
2019-08-01	100	90
2019-08-02	101	100
2019-08-05	103	110
2019-08-06	105	90

Out[50]:

	Naver	Kakao
2008	100	90
2009	101	100
2010	103	110
2011	105	90

2년 단위로 downsampling을 하고 싶다면?

Out[51]:

	Naver	Kakao
2008	201	190
2010	208	200

Out[52]:

	Naver	Kakao
2008	100.5	95
2010	104.0	100

분기 단위로 upsampling을 하고 싶다면?

Out[53]:

	Naver	Kakao
2008Q1	100	90
2008Q2	100	90
2008Q3	100	90
2008Q4	100	90
2009Q1	101	100
2009Q2	101	100
2009Q3	101	100
2009Q4	101	100
2010Q1	103	110
2010Q2	103	110
2010Q3	103	110
2010Q4	103	110
2011Q1	105	90
2011Q2	105	90
2011Q3	105	90
2011Q4	105	90

Out[54]:

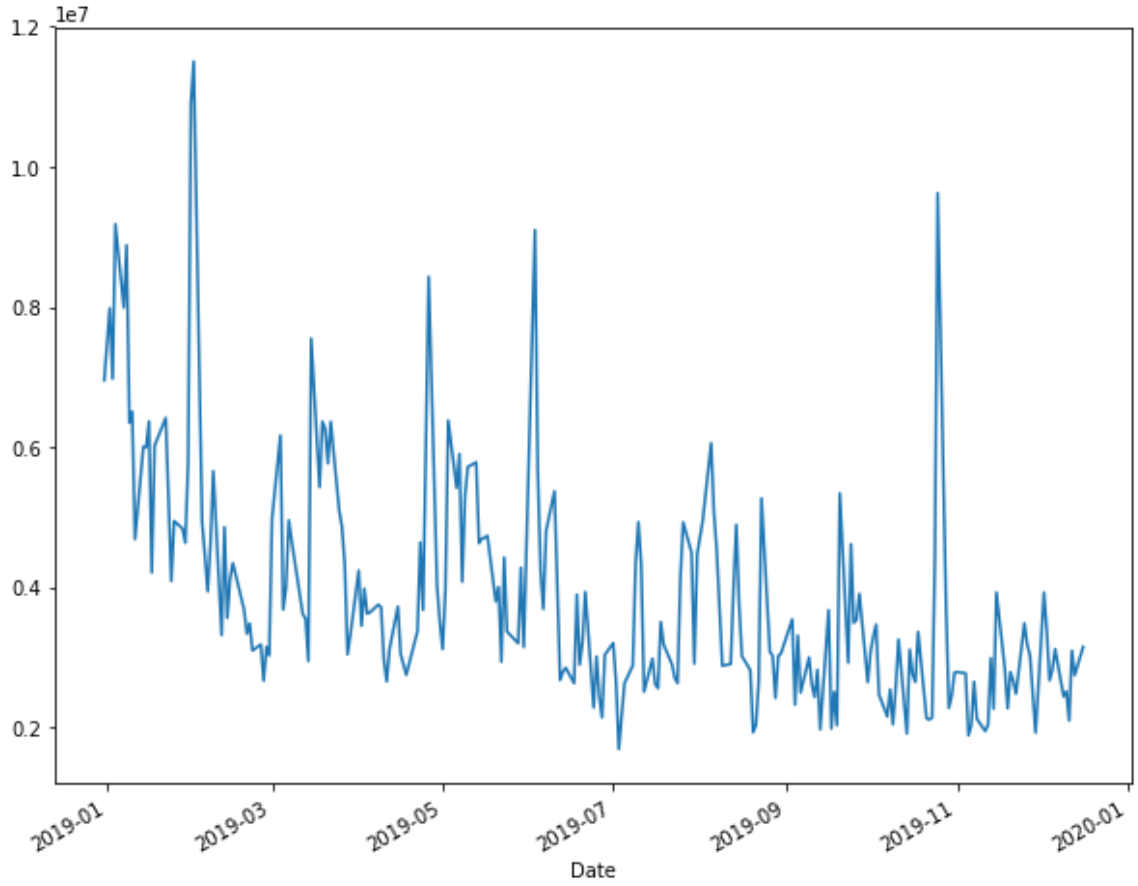
	Naver	Kakao
2008Q1	100	90
2008Q2	100	90
2008Q3	100	90
2008Q4	100	90
2009Q1	101	100
2009Q2	101	100
2009Q3	101	100
2009Q4	101	100
2010Q1	103	110
2010Q2	103	110
2010Q3	103	110
2010Q4	103	110
2011Q1	105	90
2011Q2	105	90
2011Q3	105	90
2011Q4	105	90

5. Time Series Plotting

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 243 entries, 2018-12-31 to 2019-12-16
Data columns (total 3 columns):
apple    243 non-null int64
ms       243 non-null int64
amz      243 non-null int64
dtypes: int64(3)
memory usage: 7.6 KB
```

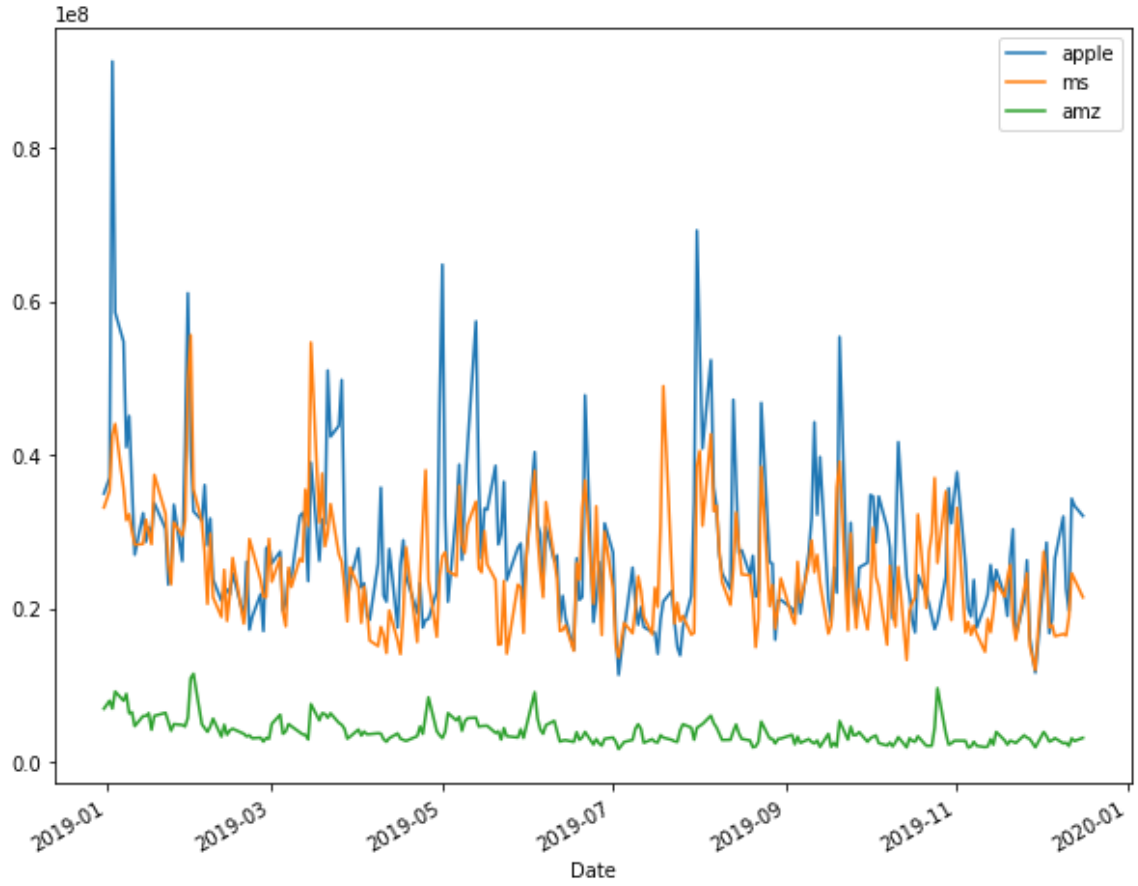
Out[14]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x28936be93c8>
```



Out[15]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x28936caa9e8>
```



Moving Window Functions

Out[16]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x28936c16208>
```