빅데이터 실습

13주차 1차시

시계열 데이터 기초 [1]

시계열 데이터

주요 자료형 소개

시계열 데이터

학습개요

1/ 날짜와 시간 데이터 처리를 위한 datetime 모듈

시계열 데이터

학습개요

1/ 날짜와 시간 데이터 처리를 위한 datetime 모듈

2/ 시계열 데이터(Time-Series) 소개 및 처리

01

날짜와 시간 데이터 처리를 위한 datetime 모듈

1 날짜와 시간 데이터 처리를 위한 datetime 모듈

datetime 모듈 (파이썬 모듈)

- ✓ 날짜와 시간 : 파이썬에서 제공하는 자료형에 포함되어 있지 않음
  - ◉ 데이터 분석 시 자주 처리하는 중요한 데이터 자료형
- ✓ 파이썬으로 날짜와 시간 데이터를 효율적으로 처리할 수 있는 자료형과 함수 제공

1 날짜와 시간 데이터 처리를 위한 datetime 모듈

7 datetime 모듈의 주요 객체

날짜와 시간을 모두 저장하고 활용할 수 있어 date나 time보다 많이 사용

datetime	날짜와 시간 정보를 저장하는 객체
date	날짜만 저장하는 객체
time	시간만 저장하는 객체
timedelta	datetime 객체 간 시간 차이 정보를 저장하는 객체

1 날짜와 시간 데이터 처리를 위한 datetime 모듈

7 datetime 모듈의 주요 객체

```
datetime.datetime(
    year, // 년
    month, // 월
    day, // 일
    hour = 0, // 시
    minute = 0, // 분
    second = 0, // 초
    microsecond = 0, // 마이크로초, 10^-6
    timezone, // 타임존
```

timezone

- 시간 timezone 지정 가능
- 지정하지 않을 경우
  현재 살고 있는 곳 기준으로 자동 설정

datetime 모듈 실습

◎ 시계열 데이터 타입(datetime)

✔ datetime 모듈 추가하기

Run Code

In [11]:
```
import pandas as pd
from pandas import Series, DataFrame
```

1. 시계열 데이터 기초

1.1 시계열 데이터 타입(datetime)

In [ ]:
```
import datetime as dt
```

In [ ]:

In [ ]:

datetime 모듈은 파이썬 기본 모듈에 내장되어 있지 않기 때문에 import를 별도로 해 주어야 합니다.

datetime 모듈 실습

◎ 시계열 데이터 타입(datetime)

✓ datetime 객체 생성

▶ Run ■ ↻ ▶▶ Code

In [11]:
```
import pandas as pd
from pandas import Series, DataFrame
```

1. 시계열 데이터 기초

1.1 시계열 데이터 타입(datetime)

datetime 날짜와 시간 지정하는 자료형

In [1]:
```
import datetime as dt
```

In [ ]:
```
dt.datetime()
```

In [ ]:

In [ ]:

datetime 모듈 실습

◎ 시계열 데이터 타입(datetime)

✓ datetime 객체 생성


```
ndas as pd
as import Series, DataFrame
```

1. 시계열 데이터 기초

1.1 시계열 데이터 타입(datetime)

In [1]:
```
import datetime as dt
```

In [2]:
```
dt.datetime(2021, 3,2)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-2-24e79f5a7fbe> in <module>
----> 1 dt.datetime()

TypeError: function missing required argument 'year' (pos 1)
```

datetime은 연, 월, 일까지 필수로 지정해 주어야 합니다.

In [ ]:

datetime 모듈 실습

◎ 시계열 데이터 타입(datetime)

✔ datetime 객체 생성

Run | Code

```
ndas as pd
as import Series, DataFrame
```

1. 시계열 데이터 기초

1.1 시계열 데이터 타입(datetime)

```
In [1]: import datetime as dt

In [3]: dt.datetime(2021, 3, 2)
Out[3]: datetime.datetime(2021, 3, 2, 0, 0)

In [ ]:

In [ ]:
```

1.2 timedelta 를 활용한 시간 계산

datetime 모듈 실습

◎ 시계열 데이터 타입(datetime)

✔ now 함수실행

Run Code

1 시계열 데이터 기초

1.1 시계열 데이터 타입(datetime)

```
In [1]: import datetime as dt
```

now 현재 시간 return 해 주는 함수

```
In [4]: dt.datetime.now()
Out[4]: datetime.datetime(2021, 2, 26, 18, 41, 44, 927283)
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

datetime 모듈 실습

◎ 시계열 데이터 타입(datetime)

✔ timezone 지정

▶ Run ■ C ▶▶ Code

1.1 시계열 데이터 타입(datetime)

In [1]:
```
import datetime as dt
```

In [3]:
```
dt.datetime(2021, 3, 2)
```

Out[3]: datetime.datetime(2021, 3, 2, 0, 0)

timezone 현재 위치에 해당하는 시간대(timezone)로 기본설정

In [6]:
```
now
```

Out[6]: datetime.datetime(2021, 2, 26, 18, 41, 54, 468656)

In [ ]:
```

```

In [ ]:
```

```

1.2 timedelta 를 활용한 시간 계산

datetime 모듈 실습

◎ 시계열 데이터 타입(datetime)

✔ timezone 지정

1.1 시계열 데이터 타입(datetime)

```
In [1]: import datetime as dt

In [3]: dt.datetime(2021, 3, 2)
Out[3]: datetime.datetime(2021, 3, 2, 0, 0)
```

timezone

```
In [ ]: w = dt.datetime.now()

In [6]: now
Out[6]: datetime.datetime(2021, 2, 26, 18, 41, 54, 468656)

In [ ]: # 타임존 지정해서 날짜 생성
        dt.datetime(2021, 3, 2, 13, 50, tzinfo = dt.timezone.utc)

In [ ]:
```

datetime 모듈 실습

◎ 시계열 데이터 타입(datetime)

✔ timezone 지정

Run | Markdown

```
etime.datetime(2021, 3, 2, 0, 0)
```

```
In [5]: now = dt.datetime.now()

In [6]: now
Out[6]: datetime.datetime(2021, 2, 26, 18, 41, 54, 468656)

In [7]: # 타임존 지정해서 날짜 생성
        dt.datetime(2021, 3, 2, 13, 50, tzinfo = dt.timezone.utc)
Out[7]: datetime.datetime(2021, 3, 2, 13, 50, tzinfo=datetime.timezone.utc)
```

astimezone 만들어져 있는 datetime 객체의 시간대도 timezone에 맞춰 변경

```
In [10]: now.astimezone(dt.timezone.utc)
Out[10]: datetime.datetime(2021, 2, 26, 9, 41, 54, 468656, tzinfo=datetime.timezone.utc)
```

datetime 모듈 실습

◎ timedelta를 활용한 시간 계산

Run Code

In [7]:
```
# 타임존 지정해서 날짜 생성
dt.datetime(2021, 3, 2, 13, 50, tzinfo = dt.timezone.utc)
```

Out[7]:
```
datetime.datetime(2021, 3, 2, 13, 50, tzinfo=datetime.timezone.utc)
```

In [8]:
```
dt.datetime.now(tz = dt.timezone.utc)
```

Out[8]:
```
datetime.datetime(2021, 2, 26, 9, 43, 18, 647117, tzinfo=datetime.timezone.utc)
```

In [10]:
```
now.astimezone(dt.timezone.utc)
```

Out[10]:
```
datetime.datetime(2021, 2, 26, 9, 41, 54, 468656, tzinfo=datetime.timezone.utc)
```

timedelta 시간 계산을 편하게 해 주는 객체

In [11]:
```
from datetime import timedelta
```

In [ ]:

In [ ]:

datetime 모듈 실습

◎ timedelta를 활용한 시간 계산

✔ 현재 시간 기준으로 100일 뒤 날짜를 알고 싶은 경우

```
                                                    ne.utc)
```

Out[7]:
```
datetime.datetime(2021, 3, 2, 13, 50, tzinfo=datetime.timezone.utc)
```

In [8]:
```
dt.datetime.now(tz = dt.timezone.utc)
```

Out[8]:
```
datetime.datetime(2021, 2, 26, 9, 43, 18, 647117, tzinfo=datetime.timezone.utc)
```

In [10]:
```
now.astimezone(dt.timezone.utc)
```

Out[10]:
```
datetime.datetime(2021, 2, 26, 9, 41, 54, 468656, tzinfo=datetime.timezone.utc)
```

1.2 timedelta 를 활용한 시간 계산

In [11]:
```
from datetime import timedelta
```

In [12]:
```
# 현재 시간 기준으로 100일 뒤의 날짜
now + timedelta(days = 100)
```

Out[12]:
```
datetime.datetime(2021, 2, 26, 18, 41, 54, 468656)
```

datetime 모듈 실습

◎ timedelta를 활용한 시간 계산

✔ 현재 시간 기준으로 1시간 25분 후의 시간 구하기

Out[10]: datetime.datetime(2021, 2, 26, 9, 41, 54, 468656, tzinfo=datetime.timezone.utc)

1.2 timedelta 를 활용한 시간 계산

In [11]:
```
from datetime import timedelta
```

In [13]:
```
# 현재 시간 기준으로 100일 뒤의 날짜
now + timedelta(days = 100)
```

Out[13]: datetime.datetime(2021, 6, 6, 18, 41, 54, 468656)

In [14]:
```
# 현재 시간 기준으로 1시간 25분 후의 시간
now + timedelta(hours = 1, minutes = 25)
```

Out[14]: datetime.datetime(2021, 2, 26, 20, 6, 54, 468656)

현재 있는 datetime에 어떤 날짜와 관련된 값을 가감할 때 timedelta를 사용합니다.

datetime 모듈 실습

◎ timedelta를 활용한 시간 계산

✔ 두 날짜 간의 차이 계산

Run | Code

etime(2021, 2, 26, 9, 41, 54, 468656, tzinfo=datetime.timezone.utc)

1.2 timedelta 를 활용한 시간 계산

In [11]:
```
from datetime import timedelta
```

In [13]:
```
# 현재 시간 기준으로 100일 뒤의 날짜
now + timedelta(days = 100)
```

Out[13]: datetime.datetime(2021, 6, 6, 18, 41, 54, 468656)

In [14]:
```
# 현재 시간 기준으로 1시간 25분 후의 시간
now + timedelta(hours = 1, minutes = 25)
```

Out[14]: datetime.datetime(2021, 2, 26, 20, 6, 54, 468656)

In [ ]:
```
# 두 날짜 간의 차이 계산
dt1 = dt.datetime(2021, 3, 2, 9, 30)
dt2 = dt.datetime(2023, 5, 18, 13:15)
```

In [ ]:

datetime 모듈 실습

◎ timedelta를 활용한 시간 계산

✔ 두 날짜 간의 차이 계산

```
In [14]: ...ta(hours = 1, minutes = 25)
Out[14]: datetime.datetime(2021, 2, 26, 20, 6, 54, 468656)

In [15]: # 두 날짜 간의 차이 계산
         dt1 = dt.datetime(2021, 3, 2, 9, 30)
         dt2 = dt.datetime(2023, 5, 18, 13, 15)

In [17]: diff = dt2 - dt1

In [18]: diff
Out[18]: datetime.timedelta(days=807, seconds=13500)

In [ ]:

In [ ]:
```

timedelta는 초 단위를 기준으로 하기 때문에
시간이나 분 단위를 알기 위해서는 추가로 계산해 주어야 합니다.

02

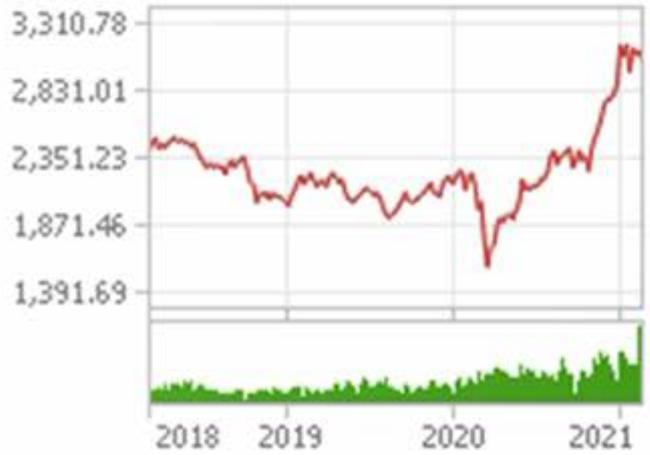
시계열 데이터(Time-Series) 소개 및 처리

시계열 데이터 (Time-Series) 소개 및 처리

② 시계열 데이터(Time-Series) 소개 및 처리

시계열 데이터 (Time-Series)

✓ 일정 시간 간격으로 배치된 데이터들의 배열



날짜	체결가	전일비	등락률	거래량(천주)	거래대금(백만)
2021.02.26	3,020.61	▼ 79.08	-2.55%	544,939	7,622,863
2021.02.25	3,099.69	▲ 104.71	+3.50%	1,280,086	17,213,950
2021.02.24	2,994.98	▼ 75.11	-2.45%	1,560,930	21,519,850
2021.02.23	3,070.09	▼ 9.66	-0.31%	2,357,758	17,661,157
2021.02.22	3,079.75	▼ 27.87	-0.90%	1,831,955	18,224,657
2021.02.19	3,107.62	▲ 20.96	+0.68%	3,455,504	19,720,469
2021.02.18	3,086.66	▼ 47.07	-1.50%	1,860,566	18,141,179
2021.02.17	3,133.73	▼ 29.52	-0.93%	1,784,126	16,846,120
2021.02.16	3,163.25	▲ 16.25	+0.52%	1,944,395	17,171,320
2021.02.15	3,147.00	▲ 46.42	+1.50%	1,604,653	16,944,151
2021.02.10	3,100.58	▲ 15.91	+0.52%	2,152,916	18,333,691
2021.02.09	3,084.67	▼ 6.57	-0.21%	2,145,388	18,389,342

② 시계열 데이터(Time-Series) 소개 및 처리

시계열 데이터 (Time-Series)

- ✓ 시간 정보를 가진 데이터들이 순차적으로 나열되어 있음을 의미
- ✓ 벡터 : 여러 개 값의 데이터들이 존재
- ✓ Pandas에 있는 Series나 DataFrame 같은 자료형으로 처리 가능
- ✓ Series와 DataFrame은 숫자형, 문자열 뿐만 아니라 datetime 자료형을 포함한 모든 자료형을 값으로 가질 수 있음

시계열 데이터 처리 실습

◎ 시계열 데이터 sample 생성

1.3 시계열 데이터 처리

In [24]:
```
import random
import pandas as pd
from pandas import DataFrame, Series
sample = DataFrame()
sample['date'] = pd.date_range(start = '20210101', periods=500).tolist()
sample['count'] = random.sample(range(1,1000), 500)
```

In [25]:

sample 날짜와 어떤 count값을 가지고 있는 간단한 DataFrame

Out[25]:

	date	count
0	2021-01-01	757
1	2021-01-02	899
2	2021-01-03	259
3	2021-01-04	56
4	2021-01-05	601
...	...	...

시계열 데이터 처리 실습

◎ 시계열 데이터 sample 생성

✔ describe 함수실행

Run | Code

```
andas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   date    500 non-null    datetime64[ns]
 1   count   500 non-null    int64
dtypes: datetime64[ns](1), int64(1)
memory usage: 7.9 KB
```

In [27]:
```
sample.describe()
```

Out[27]:

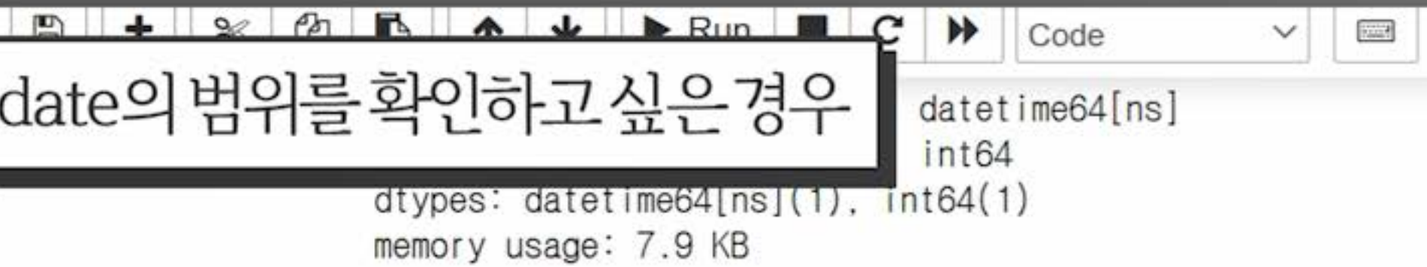
	count
count	500.000000
mean	504.216000
std	289.607781
min	2.000000
50%	499.000000

describe 함수는 숫자형 컬럼에 대해서만 통계 정보를 제공합니다.

시계열 데이터 처리 실습

◎ 시계열 데이터 sample 생성

✔ date의 범위를 확인하고 싶은 경우



시계열 데이터 처리 실습

◎ 시계열 데이터 sample 생성

✔ 7월 데이터만 조회하기

① 조건 색인 활용

```
dtypes: datetime64[ns](1), int64(1)
memory usage: 7.9 KB
```

In [31]:
```
print(sample.date.min(), sample.date.max())
```

```
2021-01-01 00:00:00 2022-05-15 00:00:00
```

In [33]:
```
# 7월달 데이터만 조회 #1
# 조건 색인 활용
samp(sample.date >= '2021-07-01') & (sample.date <= '2021-07-31')
```

Out[33]:
```
0      False
1      False
2      False
3      False
4      False
       ...
495    False
496    False
497    False
498    False
499    False
```

시계열 데이터 처리 실습

◎ 시계열 데이터 sample 생성

✔ 7월 데이터만 조회하기

② isin() 함수 활용

In [34]:
```
# 7월달 데이터만 조회 #1
# 조건 색인 활용
sample[(sample.date >= '2021-07-01') & (sample.date <= '2021-07-31')]
```

...

In [36]:
```
# 7월달 데이터만 조회 #2
# isin() 함수 활용
sample[sample.date.isin(pd.date_range())]
```

date_range() 함수 실행

Out[36]:
```
0     True
1    False
2     True
3    False
4     True
dtype: bool
```

In [15]:
```
# 7월달 데이터마 조회 #3
# datetimeIndex 활용
```

시계열 데이터 처리 실습

◎ 시계열 데이터 sample 생성

✔ 7월 데이터만 조회하기

③ Datetimeindex 활용

```
                                                    '2021-07-27', '2021-07-28',
                                                    '2021-07-31'],
              dtype='datetime64[ns]', freq='D')

In [40]: # 7월달 데이터만 조회 #2
         # isin() 함수 활용
         sample[sample.date.isin(pd.date_range(start = '2021-07-01',end = '2021-07-31'))]

                                    ...

In [42]: # 7월달 데이터마 조회 #3
         # datetimeIndex 활용
         sample.set_index('date')

         <class 'pandas.core.frame.DataFrame'>
         RangeIndex: 500 entries, 0 to 499
         Data columns (total 2 columns):
          #   Column  Non-Null Count  Dtype
         ---  ------  --------------  -----
          0   date    500 non-null    datetime64[ns]
          1   count   500 non-null    int64
```

날짜 컬럼의 자료형이 datetime인 것을 반드시 확인해야 합니다.

시계열 데이터 처리 실습

◎ 시계열 데이터 sample 생성

✔ 7월 데이터만 조회하기

③ Datetimeindex 활용

```
07-01',end = '2021-07-31'))]
```

...

In [45]:
```
# 7월달 데이터마 조회 #3
# datetimeIndex 활용
sample.set_index('date')
```

Out[45]:
```
DatetimeIndex(['2021-01-01', '2021-01-02', '2021-01-03', '2021-01-04',
               '2021-01-05', '2021-01-06', '2021-01-07', '2021-01-08',
               '2021-01-09', '2021-01-10',
               ...
               '2022-05-06', '2022-05-07', '2022-05-08', '2022-05-09',
               '2022-05-10', '2022-05-11', '2022-05-12', '2022-05-13',
               '2022-05-14', '2022-05-15'],
              dtype='datetime64[ns]', name='date', length=500, freq=None)
```

- datetimeIndex 색인

DatetimeIndex로 row index가 구성되어 있는 경우 색인을 쉽게 할 수 있습니다.

시계열 데이터 처리 실습

◎ 시계열 데이터 sample 생성

✔ 데이터 조회 응용실습

Run | Code

columns

In [52]:
```
# 2021년 7월 데이터만 선택
#sample2['2021-07']
sample2['2021-07-01':'2021-07-31']
```

Out[52]:

date	count
2021-07-01	635
2021-07-02	581
2021-07-03	500
2021-07-04	766
2021-07-05	628
2021-07-06	913
2021-07-07	482
2021-07-08	23
2021-07-09	883

시계열 데이터 처리 실습

◎ 시계열 데이터 sample 생성

✔ 데이터 조회 응용실습

✔ 2021년 4월 28일 데이터만 선택하기

```
----> 2 sample2['2021-04-28']

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\frame.py in __getitem__(self, key)
   2900             if self.columns.nlevels > 1:
   2901                 return self._getitem_multilevel(key)
-> 2902             indexer = self.columns.get_loc(key)
   2903             if is_integer(indexer):
   2904                 indexer = [indexer]

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\indexes\base.py in get_loc(self, key, method, tolerance)
   2895                 return self._engine.get_loc(key)
   2896             except KeyError as e:
-> 2897                 raise KeyError(key) from e
   2898
   2899         if tolerance is not None:

KeyError: '2021-04-28'
```

Error 발생

```
In [ ]:
```

시계열 데이터 처리 실습

◎ 시계열 데이터 sample 생성

✔ 데이터 조회 응용 실습

✔ 2021년 4월 28일 데이터만 선택하기

```
sample2['2021-07-01':'2021-07-31']
```

...

In [51]:
```
# 2021년 3월 2일부터 2021년 3월 28일까지 데이터 선택
sample2['2021-03-02':'2021-03-28']
```

...

In [53]:
```
# 2021년 4월 28일 데이터 선택
sample2['2021-04-28']
```

```
---------------------------------------------------------------------------
KeyError                                  T
C:WProgramDataWAnaconda3WlibWsite-packages
elf, key, method, tolerance)
   2894                 try:
-> 2895                     return self._engine
   2896                 except KeyError as err

pandasW_libsWindex.pyx in pandas._libs.ind

pandasW_libsWindex.pyx in pandas._libs.index.IndexEngine.get_loc()
```

_loc(s

KeyError 발생 원인

범위 설정 데이터는 슬라이싱 색인

특정 날짜 데이터는 슬라이싱 색인X

시계열 데이터 처리 실습

◎ 시계열 데이터 sample 생성

✔ 데이터 조회 응용 실습

✔ 2021년 4월 28일 데이터만 선택하기

```
sample2['2021-07-01':'2021-07-31']
```

...

In [51]:
```
# 2021년 3월 2일부터 2021년 3월 28일까지 데이터 선택
sample2['2021-03-02':'2021-03-28']
```

...

In [54]:
```
# 2021년 4월 28일 데이터 선택
sample2.loc['2021-04-28']
```

.loc 삽입

Out[54]:
```
count    92
Name: 2021-04-28 00:00:00, dtype: int64
```

In [ ]:

1-4 strptime()으로 문자열을 날짜 타입으로 변환하기

In [19]:
```
문자열 = '21/08/11'
```

시계열 데이터 처리 실습

◎ 시계열 데이터 sample 생성

✔ 데이터 조회 응용 실습

✔ 2021년 4월 28일 데이터만 선택하기

365 rows × 1 columns

In [52]:
```
# 2021년 7월 데이터만 선택
#sample2['2021-07']
sample2['2021-07-01':'2021-07-31']
```

...

In [51]:
```
# 2021년 3월 2일부터 2021년 3월 28일까지 데이터 선택
sample2['2021-03-02':'2021-03-28']
```

...

In [54]:
```
# 2021년 4월 28일 데이터 선택
sample2.loc['2021-04-28']
```

Out[54]:
```
count    92
Name: 2021-04-28 00:00:00, dtype: int64
```

슬라이싱 색인의 경우에만 loc를 생략할 수 있습니다.

시계열 데이터

학습완료

1/ 날짜와 시간 데이터 처리를 위한 datetime 모듈

2/ 시계열 데이터(Time-Series) 소개 및 처리

수고하셨습니다!