-Data추출이나 결손값의 취급 방법 등 pandas를 사용한 data 처리의 기본을 설명한다.

```
import pandas as pd
```

1. 논리값으로 data 추출하기
   1)Series에 대해 비교 연산자를 사용하여 loc로 label을 지정한다.

```
df = pd.read_csv('pandas_data/sungjuk_utf8.csv', header = None,
        names = ['학번', '이름', '국어', '영어', '수학', '전산'])

df.loc[df['국어'] > 90].head()
```

	학번	이름	국어	영어	수학	전산
5	1106	튼튼이	98	97	93	88
7	1108	더크게	98	67	93	78
10	1111	한산섬	98	89	73	78

```
df.query('학번 == 1104')
```

	학번	이름	국어	영어	수학	전산
3	1104	고아라	83	57	88	73

2. where method로 data 추출하기
   1)위의 경우에서는 label을 사용하여 검색하였는데, 검색 조건에 맞지 않은 행은 제외되었다.
   2)DataFrame.where() 는 해당되지 않는 data를 NaN으로 채운 DataFrame을 반환한다.

```
df.where(df['영어'] < 70)
```

	학번	이름	국어	영어	수학	전산
0	NaN	NaN	NaN	NaN	NaN	NaN
1	NaN	NaN	NaN	NaN	NaN	NaN
2	1103.0	그리운	76.0	56.0	87.0	78.0
3	1104.0	고아라	83.0	57.0	88.0	73.0
4	NaN	NaN	NaN	NaN	NaN	NaN
5	NaN	NaN	NaN	NaN	NaN	NaN
6	1107.0	한아름	68.0	67.0	83.0	89.0
7	1108.0	더크게	98.0	67.0	93.0	78.0
8	NaN	NaN	NaN	NaN	NaN	NaN
9	NaN	NaN	NaN	NaN	NaN	NaN
10	NaN	NaN	NaN	NaN	NaN	NaN
11	NaN	NaN	NaN	NaN	NaN	NaN

3. 값 변경하기
   1)DataFrame의 lable을 지정하고 값을 대입하면 지정된 범위의 값이 바뀐다.

```
df.head(3)
```

	학번	이름	국어	영어	수학	전산
0	1101	한송이	78	87	83	78
1	1102	정다워	88	83	57	98
2	1103	그리운	76	56	87	78

```
import numpy as np

df.loc[1, '전산'] = np.nan

df.loc[1, '전산']
```

```
nan
```

```
df.head(3)
```

	학번	이름	국어	영어	수학	전산
0	1101	한송이	78	87	83	78.0
1	1102	정다워	88	83	57	NaN
2	1103	그리운	76	56	87	78.0

   2)복수의 값 변경

```
df.loc[df['학번'] > 1110, '수학'] = np.nan

df.tail(2)
-----------------------------------------------
     학번      이름    국어  영어  수학     전산
10   1111      한산섬  98   89   NaN    78.0
11   1112      하나로  89   97   NaN    88.0
```

4. 결손값 제외하기
   1)NaN은 결손값 혹은 결측치로 취급된다.
   2)isnull()는 결손값일 경우 True를 반환한다.

```
df.loc[df['수학'].isnull()]
---------------------------------------
     학번      이름    국어  영어  수학     전산
10   1111      한산섬  98   89   NaN    78.0
11   1112      하나로  89   97   NaN    88.0
```

   3)결손값이 포함되어 있는 data 제외

```
df.dropna().loc[8:]   #8번째 이후 data 중 NaN값이 있는 data제외
-----------------------------------------------
   학번      이름    국어  영어  수학  전산
8  1109      더높이  88   99   53.0 88.0
9  1110      아리랑  68   79   63.0 66.0
```

   4)dropna()는 비파괴적 조작이다.
   5)따라서 df에는 이전 data가 남아있다.
   6)DataFrame의 내용을 파괴적으로 다시 쓰는 경우

```
df.dropna(inplace = True)
```

5. Data 형
   1)Series나 DataFrame은 작성된 시점에 data형이 자동으로 설정된다.
   2)수치 data는 NumPy의 data형이 저장되고, 문자열 등의 data는 object 형으로 취급된다.
   3)Series의 data 형을 확인하는 경우
      -Series의 data 형을 확인할 때에는 dtype을 참조한다.

```
df['국어'].dtype
-----------------------
dtype('int64')
```

   4)DataFrame의 data 형을 확인하는 경우
      -DataFrame의 data 형을 확인할 때는 dtypes를 참조한다.

```
df.dtypes
---------------------------------
학번      int64
이름      object
국어      int64
영어      int64
수학      float64
전산      float64
dtype: object
```

   5)형을 변환하는 경우에는 astype() 을 사용한다.
      -인수에는 type형 또는 NumPy의 data 형을 지정한다.

```
df['학번'].astype(np.str)
-----------------------------------------
0    1101
1    1102
2    1103
3    1104
...
...
10   1111
```

```
    11    1112
    Name: 학번, dtype: object
```

6)복수열의 형을 변경하는 경우
-인수에 사전을 지정한다.

```
df.astype({'영어':np.float64, '수학':np.str})

df.dtypes
-----------------------------------------------------
학번      int64
이름      object
국어      int64
영어      int64
수학      float64          #변경되지 않음. 비파괴적이어서...
전산      float64
dtype: object
```

7)DataFrame을 다시 쓰는 경우

```
df['수학'] = df['수학'].astype(np.str)
df.dtypes
-----------------------------------------------------
학번      int64
이름      object
국어      int64
영어      int64
수학      object        #변경됐음.
전산      float64
dtype: object
```

6. Sort 하기
1)Data의 순서를 바꿀 때는 sort_values()를 사용한다.
2)대상이 DataFrame일 경우에는 인수에 열 이름을 지정한다.
3)초기 설정에서는 오름차순으로 정렬한다.
4)내림차순으로 정렬하는 경우에는 keyword 인수 ascending에 False를 지정한다.

```
del df
df = pd.read_csv('pandas_data/sungjuk_utf8.csv', header = None,
          names = ['학번', '이름', '국어', '영어', '수학', '전산'])

df.sort_values('국어', ascending=False)
-------------------------------------------------------------
      학번      이름    국어  영어  수학  전산
5     1106      튼튼이  98   97   93   88
7     1108      더크게  98   67   93   78
10    1111      한산섬  98   89   73   78
...
...
```

5)sort_values() 역시 비파괴적 조작이다.
6)덮어쓰려면 inplace에 True를 할당한다.

7. 함수 적용하기
1)Series나 DataFrame에서 임의의 함수를 적용할 수 있다.
2)적용할 함수의 이름은 다음과 같다.

method	통용대상	반환값
map	Series(값별)	Series
apply	DataFrame(열 또는 행별)	Series
applymap	DataFrame(값별)	DataFrame

3)map()에 의한 함수 적용

```
emp_list = [
    {'Name':'john', 'Age':25},
    {'Name':'smith', 'Age':35},
    {'Name':'jenny', 'Age':45},
```

```
    ]

    df = pd.DataFrame.from_records(emp_list, columns=['Name', 'Age'])
    df
    ------------------------------------------------
        Name  Age
    0    john   25
    1    smith  35
    2    jenny  45

    df['Name'].map(str.capitalize)
    ------------------------------------------------------
    0    John
    1   Smith
    2   Jenny
    Name: Name, dtype: object
```

4)apply()에 의한 함수 적용

```
    df['Name'].apply(len)
    -----------------------------------
    0   4
    1   5
    2   5
    Name: Name, dtype: int64
```

5)apply()를 이용한 행에 대한 함수 적용

-각각의 행에 함수를 적용하는 경우에는, keyword 인수로 axis에 1을 지정한다.

```
        def makeConcate(row):
            return row[0] + '(' + str(row[1]) + '세)'

        df.apply(makeConcate, axis=1)
        ------------------------------------------------------
        0    john(25세)
        1   smith(35세)
        2   jenny(45세)
        dtype: object
```

6)apply()에 넘겨준 함수의 인수에 label 지정

-apply()에 넘겨진 함수의 인수는 Series형이다.

-다음 예제는 이름의 길이와 나이의 자릿수를 더한 값을 출력하는 예제이다.

```
        df.apply(lambda x: len(x['Name']) + len(str(x['Age'])), axis=1)
        ------------------------------------------------------------------------------
        0   6
        1   7
        2   7
        dtype: int64
```

7)applymap()에 의한 함수 적용

-먼저 Age의 data형을 문자열로 변환해서 Name과 Age의 문자열의 길이를 한번에 구해보자.

```
        df.dtypes
        ---------------------------------
        Name     object
        Age      int64
        dtype: object

        df['Age'] = df['Age'].astype(np.str)

        df.dtypes
        ---------------------------------
        Name     object
        Age      object
        dtype: object

        df[['Name', 'Age']].applymap(len)
        ------------------------------------------------------
            Name  Age
```

```
   0   4   2
   1   5   2
   2   5   2
```

8. 통계량 산출

1)Series나 DataFrame에는 일반적인 수학적, 통계학적 계산을 실행하는 method를 사용할 수 있다.

```
df
------------------------------------------------------------
   학번    이름   국어  영어  수학  전산
0  1101    한송이  78   87   83   78
1  1102    정다워  88   83   57   98
...
...

df.loc[:, '국어':'전산'].mean()
------------------------------------------------------------
국어   84.916667
영어   80.416667
수학   75.333333
전산   79.750000
dtype: float64
```

2)Series에서도 같은 method를 사용할 수 있다.

```
df['국어'].sum()
------------------------
1019
```

3)합계나 평균값 외에 여러 가지 통계적 method가 준비되어 있다.

```
통계적인 method의 예
------------------------------------------------------------
count     결손값을 제외한 data 수
sum       합계값
mean      평균값
median    중앙값
min       최소값
max       최대값
std       표준편차
var       분산
skew      왜도(skewness)
kurt      첨도(kurtosis)
quantile  사분위수(percentile)
cov       공분산(covariance)
corr      상관(correlation)
```

4)자세한 내용는 pandas documentation API Reference를 참조한다.

-Series Computations / Descriptive Stats :
http://pandas.pydata.org/pandas-docs/stable/api.html#computations-descriptive-stats

-DataFrame Computations / Descriptive Stats :
http://pandas.pydata.org/pandas-docs/stable/api.html#api-dataframe-stats

5)기본 통계량 산출하기

-describe()를 사용하여 기본 통계량을 산출할 수 있다.

-percent 표시는 백위수 값(전체를 100으로 작은 쪽부터세어서 몇 번째가 되는지 나타내는 수치이다. 50 백분위수가 중앙값이다)이다.

```
df.describe().round(1)    #round() 반올림함수
------------------------------------------------------------
        국어   영어   수학   전산
count  12.0   12.0   12.0   12.0
mean   84.9   80.4   75.3   79.8
std    10.7   15.3   15.2   11.6
min    68.0   56.0   53.0   55.0
25%    77.5   67.0   61.5   76.8
50%    87.5   85.0   80.5   78.0
75%    91.2   91.0   87.2   88.0
max    98.0   99.0   93.0   98.0
```

-백분위수 값을 변경할 경우에는 keyword 인수 percentiles의 list 요소에 1이상의 소수 값을 지정한다.

```
df.describe(percentiles = [0.1, 0.9]).round(1)
```

	국어	영어	수학	전산
count	12.0	12.0	12.0	12.0
mean	84.9	80.4	75.3	79.8
std	10.7	15.3	15.2	11.6
min	68.0	56.0	53.0	55.0
10%	68.8	58.0	53.4	66.7
50%	87.5	85.0	80.5	78.0
90%	98.0	97.0	92.5	88.9
max	98.0	99.0	93.0	98.0

-위의 예에서는 2개의 값을 지정하고 있지만 3개 이상 지정하는 것도 가능하다.
-DataFrame에 대해서 통계적인 연산을 하는 method를 실행한 경우에는 수치형의 열이 대상이 된다.
-비수치열에 대해 describe()를 사용하는 경우에는 다음과 같은 기본 통계량이 산출된다.
--count : 결손값을 제외한 data 수
--unique : unique한 data 수
--top : data의 수가 가장 많은 값
--freq : top의 data 수

```
df[['학번', '이름']].describe()
```

	학번	이름
count	12	12
unique	12	12
top	1102	한산섬
freq	1	1

9. Cross 집계
1)여러 개의 변수의 인과관계를 교차해서 집계하는 분석 기법
2)groupby method로 grouping하기
-지정된 열에서 grouping된 object인 DataFrameGroupBy를 반환한다.

```
df
```

	학번	이름	국어	영어	수학	전산
0	1101	한송이	78	87	83	78
1	1102	정다워	88	83	57	98
2	1103	그리운	76	56	87	78
...						
...						

```
df['전공'] = ["Computer Science", "Psychology", "Economics",
  "Physics", "Computer Science", "Economics",
  "Psychology", "Computer Science", "Physics",
  "Economics", "Physics", "Psychology"]

df
```

	학번	이름	국어	영어	수학	전산	전공
0	1101	한송이	78	87	83	78	Computer Science
1	1102	정다워	88	83	57	98	Psychology
2	1103	그리운	76	56	87	78	Economics
3	1104	고아라	83	57	88	73	Physics
4	1105	사랑해	87	87	53	55	Computer Science
5	1106	튼튼이	98	97	93	88	Economics
6	1107	한아름	68	67	83	89	Psychology
7	1108	더크게	98	67	93	78	Computer Science
8	1109	더높이	88	99	53	88	Physics
9	1110	아리랑	68	79	63	66	Economics
10	1111	한산섬	98	89	73	78	Physics
11	1112	하나로	89	97	78	88	Psychology

```
df['학번'] = df['학번'].astype(np.str)
```

-전공별 grouping

```
grouped = df.groupby('전공')
type(grouped)
---------------------------------------------------------------
pandas.core.groupby.groupby.DataFrameGroupBy
```

-여기서도 통계적, 수학적 method를 사용할 수 있다.
-자세한 사항은 http://pandas.pydata.org/pandas-docs/stable/api.html#groupby를 참조한다.

```
grouped.mean().round(1)
---------------------------------------------------------------
```

전공	국어	영어	수학	전산
Computer Science	87.7	80.3	76.3	70.3
Economics	80.7	77.3	81.0	77.3
Physics	89.7	81.7	71.3	79.7
Psychology	81.7	82.3	72.7	91.7

-복수의 요소로 grouping 하려면, groupby() 인수에 list를 넘겨주면 된다.
-아래의 예제는 억지로 전공과 국어별로 grouping을 하는 예제이다.

```
df.groupby(['전공', '국어']).mean().round(1)
---------------------------------------------------------------------
```

전공	국어	영어	수학	전산
Computer Science	78	87	83	78
	87	87	53	55
	98	67	93	78
Economics	68	79	63	66
	76	56	87	78
	98	97	93	88
Physics	83	57	88	73
	88	99	53	88
	98	89	73	78
Psychology	68	67	83	89
	88	83	57	98
	89	97	78	88

2)pivot_table method로 집약하기
-groupby()와 같은 처리를 pivot_table()에서도 실행할 수 있다.
-keyword 인수 index에 grouping 대상의 열이름, aggfunc에 집계한 함수를 지정한다.
-아래의 예제는 위의 groupby와 동일한 예제이다.

```
df.pivot_table(index='전공', aggfunc=np.mean)
---------------------------------------------------------------
```

전공	국어	수학	영어	전산
Computer Science	87.666667	76.333333	80.333333	70.333333
Economics	80.666667	81.000000	77.333333	77.333333
Physics	89.666667	71.333333	81.666667	79.666667
Psychology	81.666667	72.666667	82.333333	91.666667

```
df.pivot_table(index=['전공', '국어'], aggfunc=np.mean)
---------------------------------------------------------------------
```

전공	국어	수학	영어	전산
Computer Science	78	83	87	78
	87	53	87	55
	98	93	67	78
Economics	68	63	79	66
	76	87	56	78
	98	93	97	88
Physics	83	88	57	73
	88	53	99	88
	98	73	89	78
Psychology	68	83	67	89
	88	57	83	98
	89	78	97	88

10. 시계열 Data의 처리

1)Data준비
-예제를 위해서 Oracle Database의 emp table을 사용한다.

```
import cx_Oracle

conn = cx_Oracle.connect('scott', 'tiger', 'localhost:1521/XE')

cursor = conn.cursor()

cursor.execute("SELECThiredate, empno, ename, job, sal, comm, deptno FROM emp")

emp_list = []

for hiredate, empno, ename, job, sal, comm, deptno in cursor:
    emp_list.append([hiredate, empno, ename, job, sal, comm, deptno])

df = pd.DataFrame.from_records(emp_list, index = 'hiredate', columns=['hiredate', 'empno', 'ename', 'job', 'sal',
'comm', 'deptno'])

df
```

hiredate	empno	ename	job	sal	comm	deptno
1980-12-17	7369	SMITH	CLERK	800.0	NaN	20
1981-02-20	7499	ALLEN	SALESMAN	1600.0	300.0	30
1981-02-22	7521	WARD	SALESMAN	1250.0	500.0	30
1981-04-02	7566	JONES	MANAGER	2975.0	NaN	20
1981-09-28	7654	MARTIN	SALESMAN	1250.0	1400.0	30
1981-05-01	7698	BLAKE	MANAGER	2850.0	NaN	30
1981-06-09	7782	CLARK	MANAGER	2450.0	NaN	10
1987-07-13	7788	SCOTT	ANALYST	3000.0	NaN	20
1981-11-17	7839	KING	PRESIDENT	5000.0	NaN	10
1981-09-08	7844	TURNER	SALESMAN	1500.0	0.0	30
1987-07-13	7876	ADAMS	CLERK	1100.0	NaN	20
1981-12-03	7900	JAMES	CLERK	950.0	NaN	30
1981-12-03	7902	FORD	ANALYST	3000.0	NaN	20
1982-01-23	7934	MILLER	CLERK	1300.0	NaN	10

```
df = df.drop(df.index[[12]])  #중복된 입사날짜 행 제거
```

1)DatetimeIndex
-pandas의 DatetimeIndex는 datetime 형에 특화된 처리가 가능한 Index이다.
-pandas.date_range()는 지정한 주기(표준설정 1일)의 DatetimeIndex를 작성한다.
-date_range() 인수
--start : DatetimeIndex 시작일시를 문자열 또는 Datetime형으로 지정
--end : DatetimeIndex 종료일시를 문자열 또는 Datetime형으로 지정
--periods : 길이를 정수값으로 지정
--freq : 주기를 문자열로 지정(Offset Aliases,
https://pandas.pydata.org/pandas-docs/stable/timeseries.html#offset-aliases)
--tz : timezone을 문자열로 지정
--normalize : True을 지정해서 시각부분을 둥글게 한다.
--name : DatetimeIndex 이름을 문자열로 지정
--closed
---지정한 쪽의 기간을 닫는다.
---left를 지정해서 마지막 일시가 제외된 DatetimeIndex가 작성된다.
---right를 지정해서 최초 일시가 제외된 DatetimeIndex가 작성된다.

-아래 예제에서는 2019년 1월 1일부터 2019년 2월 1일까지의 기간에 대하여 1시간 단위로 DatetimeIndex를 작성하고 있다.

```
ix = pd.date_range('2019-01', '2019-02', freq='1H')
ix
```

```
DatetimeIndex(['2019-01-01 00:00:00', '2019-01-01 01:00:00',
               '2019-01-01 02:00:00', '2019-01-01 03:00:00',
               '2019-01-01 04:00:00', '2019-01-01 05:00:00',
               '2019-01-01 06:00:00', '2019-01-01 07:00:00',
               '2019-01-01 08:00:00', '2019-01-01 09:00:00',
               ...
               '2019-01-31 15:00:00', '2019-01-31 16:00:00',
               '2019-01-31 17:00:00', '2019-01-31 18:00:00',
```

```
                                '2019-01-31 19:00:00', '2019-01-31 20:00:00',
                                '2019-01-31 21:00:00', '2019-01-31 22:00:00',
                                '2019-01-31 23:00:00', '2019-02-01 00:00:00'],
                               dtype='datetime64[ns]', length=745, freq='H')
```

-Series의 index로 사용하는 경우

--DatetimeIndex는 Series나 DataFrame의 index로 사용할 수 있다.

```
time_series = pd.Series(np.arange(len(ix)), index=ix)
time_series
-------------------------------------------------------
2019-01-01 00:00:00      0
2019-01-01 01:00:00      1
2019-01-01 02:00:00      2
2019-01-01 03:00:00      3
2019-01-01 04:00:00      4
                        ...
2019-01-31 20:00:00    740
2019-01-31 21:00:00    741
2019-01-31 22:00:00    742
2019-01-31 23:00:00    743
2019-02-01 00:00:00    744
Freq: H, Length: 745, dtype: int32
```

2)시계열 Data를 추출하기

-DatetimeIndex의 index에는 datetime 형과 문자열형 양쪽을 지정할 수 있다.

-다음의 예에서는 index에 datetime형 값 지정

```
df.info()
-------------------------------------------------------
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 12 entries, 1980-12-17 to 1982-01-23
Data columns (total 6 columns):
empno     12 non-null int64
ename     12 non-null object
job       12 non-null object
sal       12 non-null float64
comm      4 non-null float64
deptno    12 non-null int64
dtypes: float64(2), int64(2), object(2)
memory usage: 672.0+ bytes

from datetime import datetime

df.loc[datetime(1981, 9, 8)]
-------------------------------------------
empno           7844
ename           TURNER
job             SALESMAN
sal             1500
comm            0
deptno          30
Name: 1981-09-08 00:00:00, dtype: object
```

-index에 문자열을 지정하는 경우

```
df.loc['1981-09-08']
-------------------------------------------
                empno     ename     job          sal      comm     deptno
hiredate
1981-09-08 7844         TURNER    SALESMAN    1500.0    0.0         30
```

-다음도 가능

```
df.loc['Sep-08-1981']
-------------------------------------------
                empno     ename     job          sal      comm     deptno
hiredate
1981-09-08 7844         TURNER    SALESMAN    1500.0    0.0         30
```

-특정 년도나 월의 data만 추출하는 경우
-index에 년, 월을 지정해서 특정 년, 월의 data만 추출할 수 있다.
-현재 hiredate는 문자열이기 때문에 error 발생할 것임.

```
print(df.loc['1981'])  #년도를 지정한 추출
```

hiredate	empno	ename	job	sal	comm	deptno
1981-02-20	7499	ALLEN	SALESMAN	1600.0	300.0	30
1981-02-22	7521	WARD	SALESMAN	1250.0	500.0	30
1981-04-02	7566	JONES	MANAGER	2975.0	NaN	20
1981-09-28	7654	MARTIN	SALESMAN	1250.0	1400.0	30
1981-05-01	7698	BLAKE	MANAGER	2850.0	NaN	30
1981-06-09	7782	CLARK	MANAGER	2450.0	NaN	10
1981-11-17	7839	KING	PRESIDENT	5000.0	NaN	10
1981-09-08	7844	TURNER	SALESMAN	1500.0	0.0	30

```
print(df.loc['1981-09'])  #월을 지정한 추출
```

hiredate	empno	ename	job	sal	comm	deptno
1981-09-28	7654	MARTIN	SALESMAN	1250.0	1400.0	30
1981-09-08	7844	TURNER	SALESMAN	1500.0	0.0	30

```
print(df.loc['1981-09':'1982-12'])  #년, 월등을 지정해서 slicing하는 경우
```

hiredate	empno	ename	job	sal	comm	deptno
1981-09-28	7654	MARTIN	SALESMAN	1250.0	1400.0	30
1981-11-17	7839	KING	PRESIDENT	5000.0	NaN	10
1981-09-08	7844	TURNER	SALESMAN	1500.0	0.0	30
1982-01-23	7934	MILLER	CLERK	1300.0	NaN	10

-지정한 시각만의 data를 추출하는 경우
--datetime.time 형으로 지정한 시각만의 data를 추출할 수 있다.

```
from datetime import time

time_series.loc[time(9,0)]
```

2019-01-01 09:00:00	9
2019-01-02 09:00:00	33
2019-01-03 09:00:00	57
2019-01-04 09:00:00	81
2019-01-05 09:00:00	105
2019-01-06 09:00:00	129
2019-01-07 09:00:00	153
2019-01-08 09:00:00	177
2019-01-09 09:00:00	201
2019-01-10 09:00:00	225
2019-01-11 09:00:00	249
2019-01-12 09:00:00	273
2019-01-13 09:00:00	297
2019-01-14 09:00:00	321
2019-01-15 09:00:00	345
2019-01-16 09:00:00	369
2019-01-17 09:00:00	393
2019-01-18 09:00:00	417
2019-01-19 09:00:00	441
2019-01-20 09:00:00	465
2019-01-21 09:00:00	489
2019-01-22 09:00:00	513
2019-01-23 09:00:00	537
2019-01-24 09:00:00	561
2019-01-25 09:00:00	585
2019-01-26 09:00:00	609
2019-01-27 09:00:00	633
2019-01-28 09:00:00	657
2019-01-29 09:00:00	681
2019-01-30 09:00:00	705

```
2019-01-31 09:00:00    729
Freq: 24H, dtype: int32
```

-지정한 시간대만 추출하는 경우
--between-time() 이용

```
time_series.between_time(time(9,0), time(12,0))
---------------------------------------------------------------------
2019-01-01 09:00:00     9
2019-01-01 10:00:00    10
2019-01-01 11:00:00    11
2019-01-01 12:00:00    12
2019-01-02 09:00:00    33
2019-01-02 10:00:00    34
...
...
2019-01-31 09:00:00   729
2019-01-31 10:00:00   730
2019-01-31 11:00:00   731
2019-01-31 12:00:00   732
Length: 124, dtype: int32
```

3)Resampling
-resample()을 이용하여 시계열 data의 빈도를 변환할 수 있다.
-일별 data를 주별이나 월별 등의 data로 변환할 수 있다.

```
df['sal'].resample('M').mean()
-----------------------------------------
hiredate
1980-12-31        800.0
1981-01-31          NaN
1981-02-28       1425.0
1981-03-31          NaN
1981-04-30       2975.0
1981-05-31       2850.0
1981-06-30       2450.0
1981-07-31          NaN
1981-08-31          NaN
1981-09-30       1375.0
1981-10-31          NaN
1981-11-30       5000.0
1981-12-31          NaN
1982-01-31       1300.0
1982-02-28          NaN
...
...
1987-06-30      NaN
1987-07-31    2050.0
Freq: M, Name: sal, Length: 80, dtype: float64
```

-ohlc()를 사용하여 4개 값(시작값, 마감값, 최고값, 최저값)으로 변환할 수 있다.
-다음 예제는 일별 data에서 주별 4개 값(주봉이라고 불린다)으로 변환한다.

```
df['sal'].resample('W').ohlc().head()
-------------------------------------------------------------
                open        high        low         close
hiredate
1980-12-21    800.0       800.0       800.0       800.0
1980-12-28    NaN         NaN         NaN         NaN
1981-01-04    NaN         NaN         NaN         NaN
1981-01-11    NaN         NaN         NaN         NaN
1981-01-18    NaN         NaN         NaN         NaN
```

4)Data 시각화
-pandas를 사용해서 data를 시각화할 수 있다.
-pandas의 Series 또는 DataFrame의 plot()를 사용하여 쉽게 시각화할 수 있다.
-plot()은 내부에서 Matplotlib를 사용하고 있다.
-Notebook에 graph 표시하기
--Notebook에 graph를 표시하기 위해서는 pyplot.show()를 사용한다.

```
import pandas as pd
import matplotlib.pyplot as plt

ax = pd.Series([1,2,3]).plot()
ax.set_title('Line Chart')
plt.show()
```