보충강의 데이터

day : 2023.10.09

In [1]:
```
import pandas as pd
```

In [2]:
```
x = pd.Series(['banana', 42])
x
```

Out[2]:
```
0    banana
1        42
dtype: object
```

In [3]:
```
type(x)
# 시리즈 : 하나의 변수에 대해, 하나 이상의 레코드를 나열해 둔 것!
```

Out[3]:
```
pandas.core.series.Series
```

In [4]:
```
s = pd.Series(['Layla', 'focalors'], index=['god_of_study', 'god of water'])
s
# 시리즈에도 index속성을 부여할 수 있다!
```

Out[4]:
```
god_of_study       Layla
god of water    focalors
dtype: object
```

In [5]:
```
type(s)
```

Out[5]:
```
pandas.core.series.Series
```

In [7]:
```
from numpy import NaN
```

In [8]:
```
# 데이터프레임 생성
df = pd.DataFrame({
    'Name' : ['Layla', 'arisia'],
    'Occupation' : ['Owner', 'CEO'],
    'Born' : ['2004-10-05', '2004-10-05'],
    'Died' : [NaN, NaN],
    'Age' : [20, 20]
})
```

In [9]:
```
df
```

Out[9]:

	Name	Occupation	Born	Died	Age
0	Layla	Owner	2004-10-05	NaN	20
1	arisia	CEO	2004-10-05	NaN	20

In [11]:
```
# 인덱스 부여
df = pd.DataFrame(data = {
    'Name' : ['Layla', 'arisia'],
    'Occupation' : ['Owner', 'CEO'],
    'Born' : ['2004-10-05', '2004-10-05'],
    'Died' : [NaN, NaN],
    'Age' : [20, 20]
}, index=['cute', 'genius'], columns=['Name', 'Born', 'Doed', 'Age'])
df
```

Out[11]:

	Name	Born	Doed	Age
cute	Layla	2004-10-05	NaN	20
genius	arisia	2004-10-05	NaN	20

In [13]:
```
# 순서가 보장된 데이터프레임
# 주의! : 데이터프레임 생성 시 ' , '를 구분할 때 사용한다!
from collections import OrderedDict

df = pd.DataFrame(OrderedDict([
    ('Name', ['layla', 'arisia']),
    ('Occupation', ['Owner', 'arisia']),
    ('Born', ['2004-10-05', '2004-10-05']),
    ('Died', [NaN, NaN]),
    ('Age', [20, 20])
]))

df
```

Out[13]:

	Name	Occupation	Born	Died	Age
0	layla	Owner	2004-10-05	NaN	20
1	arisia	arisia	2004-10-05	NaN	20

In [16]:
```
# 시리즈 다루기 : 기초
df = pd.DataFrame({
    'Name' : ['Layla', 'arisia'],
    'Occupation' : ['Owner', 'CEO'],
    'Born' : ['2004-10-05', '2004-10-05'],
    'Died' : [NaN, NaN],
    'Age' : [20, 20]
}, index=['Layla', 'Arisia'])
df
```

Out[16]:

	Name	Occupation	Born	Died	Age
Layla	Layla	Owner	2004-10-05	NaN	20
Arisia	arisia	CEO	2004-10-05	NaN	20

In [17]:
```
first_row = df.loc['Layla']
first_row
```

Out[17]:
```
Name               Layla
Occupation         Owner
Born          2004-10-05
Died                 NaN
Age                   20
Name: Layla, dtype: object
```

In [18]:
```
# 시리즈 속성과 메서드
first_row.index
```

Out[18]:
```
Index(['Name', 'Occupation', 'Born', 'Died', 'Age'], dtype='object')
```

In [19]:
```
first_row.keys()
# 인덱스와 동일함
```

Out[19]:
```
Index(['Name', 'Occupation', 'Born', 'Died', 'Age'], dtype='object')
```

In [21]:
```
first_row.values
# 열 데이터 출력
```

Out[21]:
```
array(['Layla', 'Owner', '2004-10-05', nan, 20], dtype=object)
```

In [23]:
```
ages = df['Age']
ages
```

Out[23]:
```
Layla     20
Arisia    20
Name: Age, dtype: int64
```

In [30]:
```
print(str(df))
```

```
          Name Occupation        Born  Died  Age
Layla    Layla      Owner  2004-10-05   NaN   20
Arisia  arisia        CEO  2004-10-05   NaN   20
```

In [31]:
```
ages.mean()
# 평균 구하기
```

Out[31]:
```
20.0
```

In [32]:
```
ages.min()
# 최소값 구하기
```

Out[32]:
```
20
```

In [35]:
```
subset = df[['Age', 'Born']]
subset
# 여러개를 가져올 때는, 데이터에 2개 괄호
```

Out[35]:

	Age	Born
Layla	20	2004-10-05
Arisia	20	2004-10-05

In [38]:
```
print(ages.describe())
```

```
count     2.0
mean     20.0
std       0.0
min      20.0
25%      20.0
50%      20.0
75%      20.0
max      20.0
Name: Age, dtype: float64
```

In [40]:
```
ages.std()
# 표준편차 구하기`
```

Out[40]:
```
0.0
```

In [41]:
```
# 불린 추출
df = pd.read_csv('../data/scientists.csv')
df
```

Out[41]:

	Name	Born	Died	Age	Occupation
0	Rosaline Franklin	1920-07-25	1958-04-16	37	Chemist
1	William Gosset	1876-06-13	1937-10-16	61	Statistician
2	Florence Nightingale	1820-05-12	1910-08-13	90	Nurse
3	Marie Curie	1867-11-07	1934-07-04	66	Chemist
4	Rachel Carson	1907-05-27	1964-04-14	56	Biologist
5	John Snow	1813-03-15	1858-06-16	45	Physician
6	Alan Turing	1912-06-23	1954-06-07	41	Computer Scientist
7	Johann Gauss	1777-04-30	1855-02-23	77	Mathematician

In [43]:
```
ages = df['Age']
ages
```

Out[43]:
```
0    37
1    61
2    90
3    66
4    56
5    45
6    41
7    77
Name: Age, dtype: int64
```

In [44]:
```
ages.mean()
```

Out[44]:
```
59.125
```

In [46]:
```
data = df[df['Age'] > df['Age'].mean()]
data
# 평균보다 큰 데이터 추출
```

Out[46]:

	Name	Born	Died	Age	Occupation
1	William Gosset	1876-06-13	1937-10-16	61	Statistician
2	Florence Nightingale	1820-05-12	1910-08-13	90	Nurse
3	Marie Curie	1867-11-07	1934-07-04	66	Chemist
7	Johann Gauss	1777-04-30	1855-02-23	77	Mathematician

In [45]:
```
data = df[df['Age'] > 59.125]
data
# 평균보다 큰 데이터 추출
```

Out[45]:

	Name	Born	Died	Age	Occupation
1	William Gosset	1876-06-13	1937-10-16	61	Statistician
2	Florence Nightingale	1820-05-12	1910-08-13	90	Nurse
3	Marie Curie	1867-11-07	1934-07-04	66	Chemist
7	Johann Gauss	1777-04-30	1855-02-23	77	Mathematician

In [47]:
```
print(ages > ages.mean())
# 불린 추출 방식으로 추출
```

```
0    False
1     True
2     True
3     True
4    False
5    False
6    False
7     True
Name: Age, dtype: bool
```

In [48]:
```
type(ages > ages.mean())
# 불린 추출의 결과값은 시리즈!
```

Out[48]:
```
pandas.core.series.Series
```

In [50]:
```
manual_bool_values = [True, True, False, False, True, True, True, True]
ages[manual_bool_values]
# 수동 추출도 가능!, true false의 반환 원리 이용
```

Out[50]:
```
0    37
1    61
4    56
5    45
6    41
7    77
Name: Age, dtype: int64
```

브로드캐스팅

시리즈와 데이터 프레임에 있는 모든 데이터에 대해 한번에 연산 처리하는 것

In [51]:
```
ages + ages
```

Out[51]:
```
0     74
1    122
2    180
3    132
4    112
5     90
6     82
7    154
Name: Age, dtype: int64
```

In [52]:
```
ages * ages
```

Out[52]:
```
0    1369
1    3721
2    8100
3    4356
4    3136
5    2025
6    1681
7    5929
Name: Age, dtype: int64
```

In [53]:
```
ages + 100
```

Out[53]:
```
0    137
1    161
2    190
3    166
4    156
5    145
6    141
7    177
Name: Age, dtype: int64
```

In [54]:
```
ages
# 위에서 한 것은 저장하지 않아서, 따로 원본 데이터가 변경된 것은 아님!
```

Out[54]:
```
0    37
1    61
2    90
3    66
4    56
5    45
6    41
7    77
Name: Age, dtype: int64
```

In [55]:
```
rev_ages = ages.sort_index(ascending=False)
rev_ages
# 인덱스를 기준으로 위에서 아래로 설정!
```

Out[55]:
```
7    77
6    41
5    45
4    56
3    66
2    90
1    61
0    37
Name: Age, dtype: int64
```

In [58]:
```
ages + pd.Series([1, 100])
# 데이터가 없어 연산할 수 없기 때문에 NaN 결측치로 데이터 출력됨!
```

Out[58]:
```
0     38.0
1    161.0
2      NaN
3      NaN
4      NaN
5      NaN
6      NaN
7      NaN
dtype: float64
```

In [59]:
```
# 데이터프레임 브로드캐스팅
df * 2
```

Out[59]:

	Name	Born	Died	Age	Occupation
0	Rosaline FranklinRosaline Franklin	1920-07-251920-07-25	1958-04-161958-04-16	74	ChemistChemist
1	William GossetWilliam Gosset	1876-06-131876-06-13	1937-10-161937-10-16	122	StatisticianStatistician
2	Florence NightingaleFlorence Nightingale	1820-05-121820-05-12	1910-08-131910-08-13	180	NurseNurse
3	Marie CurieMarie Curie	1867-11-071867-11-07	1934-07-041934-07-04	132	ChemistChemist
4	Rachel CarsonRachel Carson	1907-05-271907-05-27	1964-04-141964-04-14	112	BiologistBiologist
5	John SnowJohn Snow	1813-03-151813-03-15	1858-06-161858-06-16	90	PhysicianPhysician
6	Alan TuringAlan Turing	1912-06-231912-06-23	1954-06-071954-06-07	82	Computer ScientistComputer Scientist
7	Johann GaussJohann Gauss	1777-04-301777-04-30	1855-02-231855-02-23	154	MathematicianMathematician

In [66]:
```
print(df['Born'].dtypes)
```

```
object
```

In [69]:
```
print(df['Died'].dtypes)
```

```
object
```

In [60]:
```
# 열의 자료형 바꾸기
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8 entries, 0 to 7
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   Name        8 non-null      object
 1   Born        8 non-null      object
 2   Died        8 non-null      object
 3   Age         8 non-null      int64 
 4   Occupation  8 non-null      object
dtypes: int64(1), object(4)
memory usage: 452.0+ bytes
```

In [72]:
```
changed_Born = pd.to_datetime(df['Born'], format='%Y-%m-%d')
print(changed_Born.dtypes)
```

```
datetime64[ns]
```

In [75]:
```
changed_Died = pd.to_datetime(df['Died'], format='%Y-%m-%d')
print(changed_Died.dtypes)
```

```
datetime64[ns]
```

In [76]:
```
df['New_Born'], df['New_Died'] = changed_Born, changed_Died
df
```

Out[76]:

	Name	Born	Died	Age	Occupation	New_Born	New_Died
0	Rosaline Franklin	1920-07-25	1958-04-16	37	Chemist	1920-07-25	1958-04-16
1	William Gosset	1876-06-13	1937-10-16	61	Statistician	1876-06-13	1937-10-16
2	Florence Nightingale	1820-05-12	1910-08-13	90	Nurse	1820-05-12	1910-08-13
3	Marie Curie	1867-11-07	1934-07-04	66	Chemist	1867-11-07	1934-07-04
4	Rachel Carson	1907-05-27	1964-04-14	56	Biologist	1907-05-27	1964-04-14
5	John Snow	1813-03-15	1858-06-16	45	Physician	1813-03-15	1858-06-16
6	Alan Turing	1912-06-23	1954-06-07	41	Computer Scientist	1912-06-23	1954-06-07
7	Johann Gauss	1777-04-30	1855-02-23	77	Mathematician	1777-04-30	1855-02-23

In [77]:
```
df.shape
# 8행 7열`
```

Out[77]:
```
(8, 7)
```

In [81]:
```
# 나이 구하기
df['LiveDays'] = (df['New_Died'] - df['New_Born'])
df
```

Out[81]:

	Name	Born	Died	Age	Occupation	New_Born	New_Died	NAge	LiveDays
0	Rosaline Franklin	1920-07-25	1958-04-16	37	Chemist	1920-07-25	1958-04-16	38 days 06:36:00	13779 days
1	William Gosset	1876-06-13	1937-10-16	61	Statistician	1876-06-13	1937-10-16	62 days 05:36:00	22404 days
2	Florence Nightingale	1820-05-12	1910-08-13	90	Nurse	1820-05-12	1910-08-13	91 days 13:36:00	32964 days
3	Marie Curie	1867-11-07	1934-07-04	66	Chemist	1867-11-07	1934-07-04	67 days 15:00:00	24345 days
4	Rachel Carson	1907-05-27	1964-04-14	56	Biologist	1907-05-27	1964-04-14	57 days 17:08:00	20777 days
5	John Snow	1813-03-15	1858-06-16	45	Physician	1813-03-15	1858-06-16	45 days 21:56:00	16529 days
6	Alan Turing	1912-06-23	1954-06-07	41	Computer Scientist	1912-06-23	1954-06-07	42 days 13:36:00	15324 days
7	Johann Gauss	1777-04-30	1855-02-23	77	Mathematician	1777-04-30	1855-02-23	78 days 22:48:00	28422 days

In [84]:
```
df['LiveDays_new'] = df['LiveDays'] / 360
print(df)
# 이거 왜 결과값에서 나이가 다른지 질문!( 1차이 )
```

```
                 Name        Born        Died  Age          Occupation  W
0     Rosaline Franklin  1920-07-25  1958-04-16   37             Chemist
1        William Gosset  1876-06-13  1937-10-16   61         Statistician
2  Florence Nightingale  1820-05-12  1910-08-13   90                Nurse
3           Marie Curie  1867-11-07  1934-07-04   66             Chemist
4         Rachel Carson  1907-05-27  1964-04-14   56            Biologist
5             John Snow  1813-03-15  1858-06-16   45            Physician
6          Alan Turing  1912-06-23  1954-06-07   41   Computer Scientist
7         Johann Gauss  1777-04-30  1855-02-23   77       Mathematician

    New_Born   New_Died             NAge   LiveDays     LiveDays_new
0 1920-07-25 1958-04-16 38 days 06:36:00 13779 days 38 days 06:36:00
1 1876-06-13 1937-10-16 62 days 05:36:00 22404 days 62 days 05:36:00
2 1820-05-12 1910-08-13 91 days 13:36:00 32964 days 91 days 13:36:00
3 1867-11-07 1934-07-04 67 days 15:00:00 24345 days 67 days 15:00:00
4 1907-05-27 1964-04-14 57 days 17:08:00 20777 days 57 days 17:08:00
5 1813-03-15 1858-06-16 45 days 21:56:00 16529 days 45 days 21:56:00
6 1912-06-23 1954-06-07 42 days 13:36:00 15324 days 42 days 13:36:00
7 1777-04-30 1855-02-23 78 days 22:48:00 28422 days 78 days 22:48:00
```

In [85]:
```
df.columns
```

Out[85]:
```
Index(['Name', 'Born', 'Died', 'Age', 'Occupation', 'New_Born', 'New_Died',
       'NAge', 'LiveDays', 'LiveDays_new'],
      dtype='object')
```

In [88]:
```
df.index
```

Out[88]:
```
RangeIndex(start=0, stop=8, step=1)
```

In [87]:
```
df.keys()
# 이거 질문! index와 동일한 역할이라고 했는데 왜..?
```

Out[87]:
```
Index(['Name', 'Born', 'Died', 'Age', 'Occupation', 'New_Born', 'New_Died',
       'NAge', 'LiveDays', 'LiveDays_new'],
      dtype='object')
```

In [90]:
```
names = df['Name']
```

In [92]:
```
import os
os.listdir('./')
```

Out[92]:
```
['.ipynb_checkpoints', 'HomeStudy01.ipynb', 'pickle.pickle', 'Untitled.ipynb']
```

In [91]:
```
names.to_pickle('./pickle.pickle')
```

In [94]:
```
# 읽어오기
names = pd.read_pickle('./pickle.pickle')
names
```

Out[94]:
```
0       Rosaline Franklin
1         William Gosset
2    Florence Nightingale
3            Marie Curie
4          Rachel Carson
5              John Snow
6            Alan Turing
7           Johann Gauss
Name: Name, dtype: object
```

In [95]:
```
df.to_pickle('./df.pickle')
```

In [96]:
```
import os
os.listdir('./')
```

Out[96]:
```
['.ipynb_checkpoints',
 'df.pickle',
 'HomeStudy01.ipynb',
 'pickle.pickle',
 'Untitled.ipynb']
```

In [97]:
```
df = pd.read_pickle('./df.pickle')
df
```

Out[97]:

	Name	Born	Died	Age	Occupation	New_Born	New_Died	NAge	LiveDays	LiveDays_new
0	Rosaline Franklin	1920-07-25	1958-04-16	37	Chemist	1920-07-25	1958-04-16	38 days 06:36:00	13779 days	38 days 06:36:00
1	William Gosset	1876-06-13	1937-10-16	61	Statistician	1876-06-13	1937-10-16	62 days 05:36:00	22404 days	62 days 05:36:00
2	Florence Nightingale	1820-05-12	1910-08-13	90	Nurse	1820-05-12	1910-08-13	91 days 13:36:00	32964 days	91 days 13:36:00
3	Marie Curie	1867-11-07	1934-07-04	66	Chemist	1867-11-07	1934-07-04	67 days 15:00:00	24345 days	67 days 15:00:00
4	Rachel Carson	1907-05-27	1964-04-14	56	Biologist	1907-05-27	1964-04-14	57 days 17:08:00	20777 days	57 days 17:08:00
5	John Snow	1813-03-15	1858-06-16	45	Physician	1813-03-15	1858-06-16	45 days 21:56:00	16529 days	45 days 21:56:00
6	Alan Turing	1912-06-23	1954-06-07	41	Computer Scientist	1912-06-23	1954-06-07	42 days 13:36:00	15324 days	42 days 13:36:00
7	Johann Gauss	1777-04-30	1855-02-23	77	Mathematician	1777-04-30	1855-02-23	78 days 22:48:00	28422 days	78 days 22:48:00

In [98]:
```
# csv, tsv 저장하기
df.to_csv('./df.csv') # csv
df.to_csv('./df.tsv', sep='Wt') # tsv
```

In [99]:
```
import os
os.listdir('./')
```

Out[99]:
```
['.ipynb_checkpoints',
 'df.csv',
 'df.pickle',
 'df.tsv',
 'HomeStudy01.ipynb',
 'pickle.pickle',
 'Untitled.ipynb']
```

In [100…
```
# 인덱스 삭제 후 저장
df.to_csv('./2.csv', index=False)
```

In [101…
```
# excel 저장
import xlwt
import openpyxl
# 중요!

df.to_excel('./main.xls')
df.to_excel('./maun2.xlsx')
```

```
C:WUsersWstarlWAppDataWLocalWTempWipykernel_17588W797460593.py:5: FutureWarning: As the xlwt package is no longer maintained, the xlwt engine will be remove
d in a future version of pandas. This is the only engine in pandas that supports writing in the xls format. Install openpyxl and write to an xlsx file inste
ad. You can set the option io.excel.xls.writer to 'xlwt' to silence this warning. While this option is deprecated and will also raise a warning, it can be g
lobally set and the warning suppressed.
  df.to_excel('./main.xls')
```

In [102…
```
import os
os.listdir('./')
```

Out[102]:
```
['.ipynb_checkpoints',
 '2.csv',
 'df.csv',
 'df.pickle',
 'df.tsv',
 'HomeStudy01.ipynb',
 'main.xls',
 'maun2.xlsx',
 'pickle.pickle',
 'Untitled.ipynb']
```