Series/DataFrame 만들기

In [10]:
```python
import pandas as pd
import numpy as np
```

Series

```
data=list or arraylist,
index=None,
dtype: Dtype | None = None,
name=None,
```

list

In [12]:
```python
mylist=[1,2,3,'A',np.nan]
s=pd.Series(data=mylist, index=range(5),name="seq")
s
```

Out[12]:
```
0      1
1      2
2      3
3      A
4    NaN
Name: seq, dtype: object
```

array

In [15]:
```python
#array는 리스트를 만들어 형변환을 시키는 것
mylist=[1,2,3,'A',np.nan]
myarr=np.array(mylist)
myarr #array(['1', '2', '3', 'A', 'nan'], dtype='<U32')
print(myarr) #['1' '2' '3' 'A' 'nan']
print(myarr.shape) #(5,): 1행 5열 ** / .shape : 모양 확인
```

```
['1' '2' '3' 'A' 'nan']
(5,)
```

In [24]:
```python
mylist=[1,2,3,'A',np.nan]
myarr=np.array(mylist)
s=pd.Series(data=myarr, index=range(5),name="seq")
print(s.shape) #(5,) : array로 series 만들었기때문에 array 개수를 사용
print(len(s)) #5 : row 개수
```

```
(5,)
5
```

1. shape() - 함수 X

2. mylist.shape - 리스트는 shape 없음

list + list

In [43]:
```python
mylist=[[1,2,3],[10,20,30]]
print(mylist) #[[1, 2, 3], [10, 20, 30]]
#print(mylist.shape) : 리스트는 shape 없음
print(len(mylist), len(mylist[0]))

myarr=np.array(mylist)
print(myarr) #[[ 1  2  3]
             #[10 20 30]]
print(myarr.shape)
```

```
[[1, 2, 3], [10, 20, 30]]
2 3
[[ 1  2  3]
 [10 20 30]]
(2, 3)
```

DataFrame

```
- head(), tail(), shape, describe()

-       data=None,
        index: Axes | None = None,
        columns: Axes | None = None,
        dtype: Dtype | None = None,
```

list or array

In [31]:
```python
mylist=[[1,2,3],[10,20,30]]
myarr=np.array(mylist)

#df=pd.DataFrame(data=myarr)
df=pd.DataFrame(data=mylist, columns=['a','b','c'], index = [10,20])
print(df.shape) #(2, 3): 2행 3열
print(len(df)) #row 개수
df
```

```
(2, 3)
2
```

Out[31]:

	a	b	c
10	1	2	3
20	10	20	30

Series

In [52]:
```python
s = pd.Series([1,2,3,4])
df =pd.DataFrame(['a','b','c','d'], columns=['code'])
df['seq']=s
df['sal']=[100,200,300,400]
df[['id','pw']] = np.array([['kim',11]]*4)
df['tel']='000' #모든 값이 000으로 삽입
                #없는 컬럼이면 열 추가, 있는 컬럼이면 값 변경
df
```

Out[52]:

	code	seq	sal	id	pw	tel
0	a	1	100	kim	11	000
1	b	2	200	kim	11	000
2	c	3	300	kim	11	000
3	d	4	400	kim	11	000

list + dic [{}]

- dict에 사용 된 키 값이 dataframe의 컬럼명이 된다

In [67]:
```python
listdic = [{"uid":"kim","upw":111},
      {"uid":"kim","upw":111}]
df=pd.DataFrame(listdic) #딕셔너리에 사용된 키 값이 컬럼 이름이 됨
df
```

Out[67]:

	uid	upw
0	kim	111
1	kim	111

list + list

In [71]:
```python
listlist = [["kim",111],
      ["kim",111]]
df=pd.DataFrame(listlist, columns=['a','b'])
df
```

Out[71]:

	a	b
0	kim	111
1	kim	111

DataFrame 구조 보기

index

In [56]:
```python
df =pd.DataFrame(['a','b','c','d'], columns=['code'], index=[10,20,30,40])
df.index #인덱스가 안보이게 하는 방법은 없음
print(df.index) #Int64Index([10, 20, 30, 40], dtype='int64')
print(df.index.values) # 인덱스의 값만 보이기 :[10 20 30 40]
print(df.index.values.tolist()) #리스트 형태로 변환 : [10, 20, 30, 40]
```

```
Int64Index([10, 20, 30, 40], dtype='int64')
[10 20 30 40]
[10, 20, 30, 40]
```

In [57]:
```python
df =pd.DataFrame(['a','b','c','d'], columns=['code'])
df.index = [10,20,30,40] # 인덱스 별도 지정
df
```

Out[57]:

	code
10	a
20	b
30	c
40	d

columns

In [60]:
```python
s = pd.Series([1,2,3,4])
df =pd.DataFrame(['a','b','c','d'], columns=['code'])
df['seq']=s
df['sal']=[100,200,300,400]
print(df.columns)
print(df.columns.values)
print(df.columns.values.tolist())
```

```
Index(['code', 'seq', 'sal'], dtype='object')
['code' 'seq' 'sal']
['code', 'seq', 'sal']
```

values

In [62]:
```python
s = pd.Series([1,2,3,4])
df =pd.DataFrame(['a','b','c','d'], columns=['code'])
df['seq']=s
df['sal']=[100,200,300,400]


print(df.values)  #df의 값만 빼오기
print(df.values.tolist()) #array --> list
```

```
[['a' 1 100]
 ['b' 2 200]
 ['c' 3 300]
 ['d' 4 400]]
[['a', 1, 100], ['b', 2, 200], ['c', 3, 300], ['d', 4, 400]]
```

info()

In [32]:
```python
df.info() #데이터프레임의 사이즈, 인덱스, 컬럼 등의 정보 출력
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2 entries, 10 to 20
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   a       2 non-null      int64
 1   b       2 non-null      int64
 2   c       2 non-null      int64
dtypes: int64(3)
memory usage: 64.0 bytes
```

head()

In [36]:
```python
#df.head(5)
df.head(1) # 앞에서부터 1행 출력
```

Out[36]:

	a	b	c
10	1	2	3

tail()

In [37]:
```python
df.tail(1) # 뒤에서부터 1행 출력
```

Out[37]:

	a	b	c
20	10	20	30

describe

In [39]:
```python
df.describe()
```

Out[39]:

	a	b	c
count	2.000000	2.000000	2.000000
mean	5.500000	11.000000	16.500000
std	6.363961	12.727922	19.091883
min	1.000000	2.000000	3.000000
25%	3.250000	6.500000	9.750000
50%	5.500000	11.000000	16.500000
75%	7.750000	15.500000	23.250000
max	10.000000	20.000000	30.000000

SELECT in DataFrame

- loc[줄,칸] : 값으로 꺼내기
- iloc[줄,칸] : 인덱스로 꺼내기
- [줄,칸]
  - 단일값
  - 리스트: [단일값, 단일값, 단일값]
  - 슬라이싱[:]

파일꺼내오기

In [76]:
```python
df=pd.read_csv("./lec08_emp.csv", sep=",", parse_dates=['HIREDATE']) #parse_dates : 날짜 타입으로 변환
df.head(14)
```

Out[76]:

	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
0	7369	SMITH	CLERK	7902.0	1980-12-17	800	NaN	20
1	7499	ALLEN	SALESMAN	7698.0	1981-02-20	1600	300.0	30
2	7521	WARD	SALESMAN	7698.0	1981-02-22	1250	500.0	30
3	7566	JONES	MANAGER	7839.0	1981-04-02	2975	NaN	20
4	7654	MARTIN	SALESMAN	7698.0	1981-09-28	1250	1400.0	30
5	7698	BLAKE	MANAGER	7839.0	1981-05-01	2850	NaN	30
6	7782	CCC	MANAGER	7839.0	1981-06-09	2450	NaN	10
7	7788	SCOTT	ANALYST	7566.0	1987-07-13	3000	NaN	20
8	7839	KING	PRESIDENT	NaN	1981-11-17	5000	NaN	10
9	7844	TURNER	SALESMAN	7698.0	1981-09-08	1500	0.0	30
10	7876	ADAMS	CLERK	7788.0	1987-07-13	1100	NaN	20
11	7900	JAMES	CLERK	7698.0	1981-12-03	950	NaN	30
12	7902	FORD	ANALYST	7566.0	1981-12-03	3000	NaN	20
13	7934	MILLER	CLERK	7782.0	1982-01-23	1300	NaN	10

*** 숫자값에 nan이 있으면 타입은 float(소수점)**

In [75]:
```python
df.info()
#RangeIndex: 14 entries : 총 레코드 개수
#4 non-null : 4개 제외한 나머지 결측
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14 entries, 0 to 13
Data columns (total 8 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   EMPNO     14 non-null     int64
 1   ENAME     14 non-null     object
 2   JOB       14 non-null     object
 3   MGR       13 non-null     float64
 4   HIREDATE  14 non-null     datetime64[ns]
 5   SAL       14 non-null     int64
 6   COMM      4 non-null      float64
 7   DEPTNO    14 non-null     int64
dtypes: datetime64[ns](1), float64(2), int64(3), object(2)
memory usage: 1.0+ KB
```

loc

In [ ]:

In [ ]:

In [ ]:

iloc

In [ ]:

In [ ]:

In [ ]:

결측(missing value)처리

- np.nan
- df.dropna(axis=1 또는 axis=0)
- df.fillna(값)
- pd.isna(df)

df.dropna

- axis=0 :행
- axis=1 :열

In [97]:
```python
dfcp=df.copy()
#dfcp= dfcp.dropna(axis=0)  # drop하고 난 후 dfcp에 적용시켜라
                            # dfcp.dropna(axis=0) 만 하면 메모리에서만 지워지는 것
dfcp.dropna(axis=0,inplace=True) #inplace=True : 덮어쓰기
```

In [95]:
```python
dfcp
```

Out[95]:

	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
1	7499	ALLEN	SALESMAN	7698.0	1981-02-20	1600	300.0	30
2	7521	WARD	SALESMAN	7698.0	1981-02-22	1250	500.0	30
4	7654	MARTIN	SALESMAN	7698.0	1981-09-28	1250	1400.0	30
9	7844	TURNER	SALESMAN	7698.0	1981-09-08	1500	0.0	30

In [98]:
```python
dfcp=df.copy()
dfcp.dropna(axis=1,inplace=True)
dfcp
```

Out[98]:

	EMPNO	ENAME	JOB	HIREDATE	SAL	DEPTNO
0	7369	SMITH	CLERK	1980-12-17	800	20
1	7499	ALLEN	SALESMAN	1981-02-20	1600	30
2	7521	WARD	SALESMAN	1981-02-22	1250	30
3	7566	JONES	MANAGER	1981-04-02	2975	20
4	7654	MARTIN	SALESMAN	1981-09-28	1250	30
5	7698	BLAKE	MANAGER	1981-05-01	2850	30
6	7782	CCC	MANAGER	1981-06-09	2450	10
7	7788	SCOTT	ANALYST	1987-07-13	3000	20
8	7839	KING	PRESIDENT	1981-11-17	5000	10
9	7844	TURNER	SALESMAN	1981-09-08	1500	30
10	7876	ADAMS	CLERK	1987-07-13	1100	20
11	7900	JAMES	CLERK	1981-12-03	950	30
12	7902	FORD	ANALYST	1981-12-03	3000	20
13	7934	MILLER	CLERK	1982-01-23	1300	10

df.fillna(inplace=True)

- 결측 채우기 후 일반적으로 형변환을 한다.

- df['컬럼명']=df['컬럼명'].astype('타입')

In [100…
```python
dfcp=df.copy()
dfcp=dfcp.fillna(99999999)
#dfcp.fillna(99999999,inplace=True)
dfcp
```

Out[100]:

	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
0	7369	SMITH	CLERK	7902.0	1980-12-17	800	99999999.0	20
1	7499	ALLEN	SALESMAN	7698.0	1981-02-20	1600	300.0	30
2	7521	WARD	SALESMAN	7698.0	1981-02-22	1250	500.0	30
3	7566	JONES	MANAGER	7839.0	1981-04-02	2975	99999999.0	20
4	7654	MARTIN	SALESMAN	7698.0	1981-09-28	1250	1400.0	30
5	7698	BLAKE	MANAGER	7839.0	1981-05-01	2850	99999999.0	30
6	7782	CCC	MANAGER	7839.0	1981-06-09	2450	99999999.0	10
7	7788	SCOTT	ANALYST	7566.0	1987-07-13	3000	99999999.0	20
8	7839	KING	PRESIDENT	99999999.0	1981-11-17	5000	99999999.0	10
9	7844	TURNER	SALESMAN	7698.0	1981-09-08	1500	0.0	30
10	7876	ADAMS	CLERK	7788.0	1987-07-13	1100	99999999.0	20
11	7900	JAMES	CLERK	7698.0	1981-12-03	950	99999999.0	30
12	7902	FORD	ANALYST	7566.0	1981-12-03	3000	99999999.0	20
13	7934	MILLER	CLERK	7782.0	1982-01-23	1300	99999999.0	10

In [107…
```python
dfcp=df.copy()
#dfcp=dfcp.fillna(99999999)
# dfcp['COMM']=dfcp['COMM'].fillna(8888)
# dfcp['MGR']=dfcp['MGR'].fillna(77)

dfcp[['COMM','MGR']]=dfcp[['COMM','MGR']].fillna(99999)
dfcp[['COMM','MGR']]=dfcp[['COMM','MGR']].fillna(99999).astype('int') #astype('int'): int 타입으로 변환
dfcp.head(3)
```

Out[107]:

	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
0	7369	SMITH	CLERK	7902	1980-12-17	800	99999	20
1	7499	ALLEN	SALESMAN	7698	1981-02-20	1600	300	30
2	7521	WARD	SALESMAN	7698	1981-02-22	1250	500	30

df.isna / pd.isna(df1)

- df에 결측이 있으면 true

In [110…
```python
dfcp=df.copy()
#pandas 것
#pd.isna(dfcp)
#dfcp.isna()
#dfcp.notna() # nan이 아닌 것 true

#dfcp.isnull() #numpy 것
#dfcp.notnull() #numpy 것
```

Out[110]:

	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
0	True	True	True	True	True	True	False	True
1	True	True	True	True	True	True	True	True
2	True	True	True	True	True	True	True	True
3	True	True	True	True	True	True	False	True
4	True	True	True	True	True	True	True	True
5	True	True	True	True	True	True	False	True
6	True	True	True	True	True	True	False	True
7	True	True	True	True	True	True	False	True
8	True	True	True	False	True	True	False	True
9	True	True	True	True	True	True	True	True
10	True	True	True	True	True	True	False	True
11	True	True	True	True	True	True	False	True

L0	True	True	True	True	True	True	False	True
L1	True	True	True	True	True	True	False	True