

이번 수업에서는

1. 행/ 열 추가
2. 원소 선택
3. 원소 변경

행 추가

- 객체.loc["새로운 행의 이름"] = 데이터값들 (리스트 또는 배열)

```
df.loc["상기"] = [95, 100, 80, 95]
```

	수학	영어	음악	체육
서준	90	98	85	100
우현	80	89	95	90
인아	70	95	100	90
상기	95	100	80	95

열 추가

- 객체["새로운 열의 이름"] = 데이터값들 (리스트 또는 배열)

```
df["미술"] = [80, 90, 95, 100]
```

	수학	영어	음악	체육	미술
서준	90	98	85	100	80
우현	80	89	95	90	90
인아	70	95	100	90	95
상기	95	100	80	95	100

열 추가

- 객체["새로운 열의 이름"] = 데이터값들 (리스트 또는 배열)

```
df["과학"] = [80]
```

```
df["과학"] = 80
```

ValueError: Length of values does not match length of index

	수학	영어	음악	체육	미술	과학
서준	90	98	85	100	80	80
우현	80	89	95	90	90	80
인아	70	95	100	90	95	80
상기	95	100	80	95	100	80

원소 선택(조회)

➤ 객체.iloc[행번호, 열번호]

```
df[2,3]          # pandas는 numpy 가 아니다
df.iloc[2,3]
df.iloc[2][3]
```

KeyError

[/usr/local/lib/python3.6/dist-package](#)

```
2896          try:
-> 2897             return self._
2898          except KeyError:
```

```
df.iloc[2,3]
```

90

```
df.iloc[2][3]
```

90

원소 선택

- 객체.loc["행이름", "열이름"]

```
df.loc["인아", "체육"]
```

```
df.loc["인아", "체육"]
```

90

```
df.loc["인아", ["체육", "영어"]] # df.iloc[2, [3, 1]]
```

```
df.loc["인아", ["체육", "영어"] ]
```

체육 90

영어 95

Name: 인아, dtype: int64

원소 값 바꾸기

➤ 원소 선택 = 새로운 값

```
# 인아의 체육점수를 95점으로  
df.loc["인아", "체육"] = 95
```

	수학	영어	음악	체육	미술
서준	90	98	85	100	80
우현	80	89	95	90	90
인아	70	95	100	95	95
상기	95	100	80	95	100

```
df.iloc[2, 3] = 90  
df
```

원소 값 바꾸기

- Extract a diagonal or construct a diagonal

➤ 여러 개를 바꾸려면

```
df.loc["인아", ["체육", "영어"]] = 80, 90
```

	수학	영어	음악	체육	미술
서준	90	98	85	100	80
우현	80	89	95	90	90
인아	70	90	100	80	95
상기	95	100	80	95	100

이번 수업에서는

1. 파일 읽기. csv, Excel 쓰기
2. 데이터 프레임 살펴보기
3. 기본 통계함수 적용해 보기

csv 파일 읽기

```
df = pd.read_csv("https://raw.githubusercontent.com/plotly/datasets/master/auto-mpg.csv")  
df
```

	mpg	cylinders	displacement	horsepower	weight	acceleration	model-year
0	18.0	8	307.0	130.0	3504	12.0	70
1	15.0	8	350.0	165.0	3693	11.5	70
2	18.0	8	318.0	150.0	3436	11.0	70
3	16.0	8	304.0	150.0	3433	12.0	70

```
df.to_csv("mpg.csv")
```

excel 파일 읽기

```
df = pd.read_excel('http://qrc.depaul.edu/Excel_Files/Presidents.xls')
df
```

	President	Years in office	Year first inaugurated	Age at inauguration	State elected from	# of electoral votes
0	George Washington	8.0	1789	57	Virginia	69
1	John Adams	4.0	1797	61	Massachusetts	132
2	Thomas Jefferson	8.0	1801	57	Virginia	73

```
df.to_excel("President.xlsx")
```

데이터 프레임 살펴보기

- `head()`
- `tail()`
- `describe()`

- `shape`
- `dtypes`

데이터프레임 살펴보기

```
df = pd.read_csv("https://raw.githubusercontent.com/plotly/datasets/master/auto-mpg.csv")  
df
```

	mpg	cylinders	displacement	horsepower	weight	acceleration	model-year
0	18.0	8	307.0	130.0	3504	12.0	70
1	15.0	8	350.0	165.0	3693	11.5	70
2	18.0	8	318.0	150.0	3436	11.0	70
3	16.0	8	304.0	150.0	3433	12.0	70

데이터프레임 살펴보기

```
df.head()
```

	mpg	cylinders	displacement	horsepower	weight	acceleration	model-year
0	18.0	8	307.0	130.0	3504	12.0	70
1	15.0	8	350.0	165.0	3693	11.5	70
2	18.0	8	318.0	150.0	3436	11.0	70
3	16.0	8	304.0	150.0	3433	12.0	70
4	17.0	8	302.0	140.0	3449	10.5	70

데이터프레임 살펴보기

```
df.tail()
```

	mpg	cylinders	displacement	horsepower	weight	acceleration	model-year
393	27.0	4	140.0	86.0	2790	15.6	82
394	44.0	4	97.0	52.0	2130	24.6	82
395	32.0	4	135.0	84.0	2295	11.6	82
396	28.0	4	120.0	79.0	2625	18.6	82
397	31.0	4	119.0	82.0	2720	19.4	82

```
df.tail(2)
```

	mpg	cylinders	displacement	horsepower	weight	acceleration	model-year
396	28.0	4	120.0	79.0	2625	18.6	82
397	31.0	4	119.0	82.0	2720	19.4	82

데이터프레임 살펴보기

`df.info()`

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 398 entries, 0 to 397
Data columns (total 7 columns):
mpg                398 non-null float64
cylinders          398 non-null int64
displacement       398 non-null float64
horsepower         396 non-null float64
weight             398 non-null int64
acceleration       398 non-null float64
model-year         398 non-null int64
dtypes: float64(4), int64(3)
memory usage: 21.9 KB
```

`df.shape`

(398, 7)

`df.dtypes`

```
mpg                float64
cylinders          int64
displacement       float64
horsepower         float64
weight             int64
acceleration       float64
model-year         int64
dtype: object
```

`df.size`

2786

데이터프레임 살펴보기

➤ describe() 기초통계량

```
df.describe()
```

	mpg	cylinders	displacement	horsepower	weight	acceleration
count	398.000000	398.000000	398.000000	396.000000	398.000000	398.000000
mean	23.514573	5.454774	193.425879	104.189394	2970.424623	15.568090
std	7.815984	1.701004	104.269838	38.402030	846.841774	2.757689
min	9.000000	3.000000	68.000000	46.000000	1613.000000	8.000000
25%	17.500000	4.000000	104.250000	75.000000	2223.750000	13.825000
50%	23.000000	4.000000	148.500000	92.000000	2803.500000	15.500000
75%	29.000000	8.000000	262.000000	125.000000	3608.000000	17.175000
max	46.600000	8.000000	455.000000	230.000000	5140.000000	24.800000

데이터프레임 살펴보기

➤ `count()` 빈도수

```
df.count()
```

```
mpg          398
cylinders    398
displacement 398
horsepower   396
weight       398
acceleration 398
model-year   398
dtype: int64
```

➤ `value_counts()` 중복제거한 unique 한 개수

```
df['cylinders'].value_counts()
```

```
4      204
8      103
6       84
3        4
5         3
Name: cylinders, dtype: int64
```

기초통계량 직접 계산하기

df.mean()

```
mpg          23.514573
cylinders     5.454774
displacement 193.425879
horsepower   104.189394
weight      2970.424623
acceleration  15.568090
model-year   76.010050
dtype: float64
```

df.std()

```
mpg          7.8159
cylinders     1.7010
displacement 104.2698
horsepower    38.4020
weight      846.8417
acceleration   2.7570
model-year    3.6970
dtype: float64
```

df.median()

```
mpg          23.0
cylinders     4.0
displacement 148.5
horsepower    92.0
weight      2803.5
acceleration  15.5
model-year    76.0
dtype: float64
```

df.var()

```
mpg          61.0
cylinders     2.8
displacement 10872.0
horsepower   1474.0
weight      717140.0
acceleration   7.6
model-year   13.6
dtype: float64
```

df.max()

```
mpg          46.6
cylinders     8.0
displacement 455.0
horsepower   230.0
weight      5140.0
acceleration  24.8
model-year    82.0
dtype: float64
```

df['mpg'].mean()

```
23.514572864321615
```

상관계수 구하기

df.corr()

	mpg	cylinders	displacement	horsepower	weight	acceleration	model-year
mpg	1.000000	-0.775396	-0.804203	-0.777575	-0.831741	0.420289	0.579267
cylinders	-0.775396	1.000000	0.950721	0.843751	0.896017	-0.505419	-0.348746
displacement	-0.804203	0.950721	1.000000	0.897787	0.932824	-0.543684	-0.370164
horsepower	-0.777575	0.843751	0.897787	1.000000	0.864350	-0.687241	-0.420697
weight	-0.831741	0.896017	0.932824	0.864350	1.000000	-0.417457	-0.306564
acceleration	0.420289	-0.505419	-0.543684	-0.687241	-0.417457	1.000000	0.288137
model-year	0.579267	-0.348746	-0.370164	-0.420697	-0.306564	0.288137	1.000000

상관계수 구하기

```
df[["mpg", "cylinders", "displacement"]].corr()
```

	mpg	cylinders	displacement
mpg	1.000000	-0.775396	-0.804203
cylinders	-0.775396	1.000000	0.950721
displacement	-0.804203	0.950721	1.000000

이번 수업에서는

결측치

결측치란

- 결측치란

```
import seaborn as sns
import pandas as pd
import numpy as np

df = sns.load_dataset('titanic')
df.head()
```

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 891 entries, 0 to 890  
Data columns (total 15 columns):  
survived      891 non-null int64  
pclass        891 non-null int64  
sex           891 non-null object  
age           714 non-null float64  
sibsp         891 non-null int64  
parch         891 non-null int64  
fare          891 non-null float64  
embarked      889 non-null object  
class         891 non-null category  
who           891 non-null object  
adult_male    891 non-null bool  
deck         203 non-null category  
embark_town   889 non-null object  
alive         891 non-null object  
alone         891 non-null bool  
dtypes: bool(2), category(2), float64(2), int64(4), object(5)  
memory usage: 80.6+ KB
```


결측치 존재

```
df.isnull()
```

	survived	pclass	sex	age	s
0	False	False	False	False	
1	False	False	False	False	
2	False	False	False	False	
3	False	False	False	False	
4	False	False	False	False	
...

```
df.isnull().sum()
```

survived	0
pclass	0
sex	0
age	177
sibsp	0
parch	0
fare	0
embarked	2
class	0
who	0
adult_male	0
deck	688
embark_town	2
alive	0
alone	0
dtype:	int64

결측치 삭제

```
df1 = df.copy()
```

```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 15 columns):
survived      891 non-null int64
pclass        891 non-null int64
sex           891 non-null object
age           714 non-null float64
sibsp         891 non-null int64
parch         891 non-null int64
fare          891 non-null float64
embarked      889 non-null object
class         891 non-null category
who           891 non-null object
adult_male    891 non-null bool
deck         203 non-null category
```

```
df1.dropna().info()  # 행 삭제
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 182 entries, 1 to 889
Data columns (total 15 columns):
survived      182 non-null int64
pclass        182 non-null int64
sex           182 non-null object
age           182 non-null float64
sibsp         182 non-null int64
parch         182 non-null int64
fare          182 non-null float64
embarked      182 non-null object
class         182 non-null category
who           182 non-null object
adult_male    182 non-null bool
deck          182 non-null category
```

결측치 삭제

```
df1.dropna(axis =1).info() # 열 삭제
```

```
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 891 entries, 0 to 890  
Data columns (total 11 columns):  
survived      891 non-null int64  
pclass        891 non-null int64  
sex           891 non-null object  
sibsp         891 non-null int64  
parch         891 non-null int64  
fare          891 non-null float64  
class         891 non-null category  
who           891 non-null object  
adult_male    891 non-null bool  
alive         891 non-null object  
alone         891 non-null bool  
dtypes: bool(2), category(1), float64(1), int64(4), object(3)  
memory usage: 58.5+ KB
```

결측치 대체

```
# 평균값으로 대체하기
```

```
mean_age = df2['age'].mean()
```

```
df2['age'].fillna(mean_age, inplace= True)
```

```
df2['age'].isnull()
```

```
0      False
```

```
1      False
```

```
2      False
```

```
3      False
```

```
4      False
```

```
...
```

```
886     False
```

```
887     False
```

```
888     False
```

```
889     False
```

```
890     False
```

```
Name: age, Length: 891, dtype: bool
```

```
df2['age'].isnull().sum()
```

```
0
```

중복데이터

```
df = pd.DataFrame({'A' : ['a', 'a', 'b', 'a', 'b'],  
                  'B' : [1, 1, 1, 2, 2],  
                  'C' : [1, 1, 2, 2, 2]})
```

df

	A	B	C
0	a	1	1
1	a	1	1
2	b	1	2
3	a	2	2
4	b	2	2

df.duplicated()

- 중복된 행의 개수

	A	B	C
0	a	1	1
1	a	1	1
2	b	1	2
3	a	2	2
4	b	2	2

```
df.duplicated()
```

```
0    False
```

```
1     True
```

```
2    False
```

```
3    False
```

```
4    False
```

```
dtype: bool
```

```
] df['A'].duplicated() # 한개 열(vector)에도 적용
```

```
0    False
```

```
1     True
```

```
2    False
```

```
3     True
```

```
4     True
```

```
Name: A, dtype: bool
```

중복행 제거

	A	B	C
0	a	1	1
1	a	1	1
2	b	1	2
3	a	2	2
4	b	2	2

```
df.drop_duplicates()
```

	A	B	C
0	a	1	1
2	b	1	2
3	a	2	2
4	b	2	2

- 저장하려면 `inplace = True`

이번 수업에서는

데이터프레임 합치기 (concat)

concat

- Series + Series

```
E = pd.Series(['e0', 'e1', 'e2', 'e3'], name = 'e')  
F = pd.Series(['f0', 'f1', 'f2'], name = 'f', index = [3, 4, 5])  
G = pd.Series(['g0', 'g1', 'g2', 'g3'], name = 'g')
```

E

0	e0
1	e1
2	e2
3	e3

Name: e,

F

3	f0
4	f1
5	f2

Name: f,

G

0	g0
1	g1
2	g2
3	g3

Name: g,

- axis = 0 이 기본값이다

```
pd.concat([E, F])
```

0	e0
1	e1
2	e2
3	e3
3	f0
4	f1
5	f2

```
pd.concat([E, G])
```

0	e0
1	e1
2	e2
3	e3
0	g0
1	g1
2	g2
3	g3

- index에 맞게 병합

```
pd.concat([E, G], axis = 1 )
```

	e	g
0	e0	g0
1	e1	g1
2	e2	g2
3	e3	g3

```
pd.concat([E, F], axis = 1 )
```

	e	f
0	e0	NaN
1	e1	NaN
2	e2	NaN
3	e3	f0
4	NaN	f1
5	NaN	f2

- 결과물은 Series 이거나 DataFrame 이다

```
type(pd.concat([E, G], axis = 0 ) )
```

```
pandas.core.series.Series
```

```
type(pd.concat([E, G], axis = 1 ))
```

```
pandas.core.frame.DataFrame
```

concat

- DataFrame + DataFrame

```
df1 = pd.DataFrame({'a': ['a0', 'a1', 'a2'],  
                    'b': ['b0', 'b1', 'b2'],  
                    'c': ['c0', 'c1', 'c2']},  
                    index = [0, 1, 2])
```

```
df2 = pd.DataFrame({'b': ['b2', 'b3', 'b4'],  
                    'c': ['c2', 'c3', 'c4'],  
                    'd': ['d2', 'd3', 'd4']},  
                    index = [1, 2, 3])
```

df1

	a	b	c
0	a0	b0	c0
1	a1	b1	c1
2	a2	b2	c2

df2

	b	c	d
1	b2	c2	d2
2	b3	c3	d3
3	b4	c4	d4

```
pd.concat([df1,df2])
```

	a	b	c	d
0	a0	b0	c0	NaN
1	a1	b1	c1	NaN
2	a2	b2	c2	NaN
1	NaN	b2	c2	d2
2	NaN	b3	c3	d3
3	NaN	b4	c4	d4

df1

	a	b	c
0	a0	b0	c0
1	a1	b1	c1
2	a2	b2	c2

df2

	b	c	d
1	b2	c2	d2
2	b3	c3	d3
3	b4	c4	d4

```
pd.concat([df1,df2], ignore_index=True)
```

	a	b	c	d
0	a0	b0	c0	NaN
1	a1	b1	c1	NaN
2	a2	b2	c2	NaN
3	NaN	b2	c2	d2
4	NaN	b3	c3	d3
5	NaN	b4	c4	d4

```
pd.concat([df1,df2], axis = 1)
```

	a	b	c	b	c	d
0	a0	b0	c0	NaN	NaN	NaN
1	a1	b1	c1	b2	c2	d2
2	a2	b2	c2	b3	c3	d3
3	NaN	NaN	NaN	b4	c4	d4

df1

	a	b	c
0	a0	b0	c0
1	a1	b1	c1
2	a2	b2	c2

df2

	b	c	d
1	b2	c2	d2
2	b3	c3	d3
3	b4	c4	d4


```
pd.concat([df1,df2], axis = 1, join = 'inner')
```

	a	b	c	b	c	d
1	a1	b1	c1	b2	c2	d2
2	a2	b2	c2	b3	c3	d3