

Pandas 개요

이번 수업에서는

1. Pandas란
2. Pandas의 특징과 장점

- "panel datas(패널자료)"

Pandas library provide high-performance, easy-to-use data structures and data analysis tools

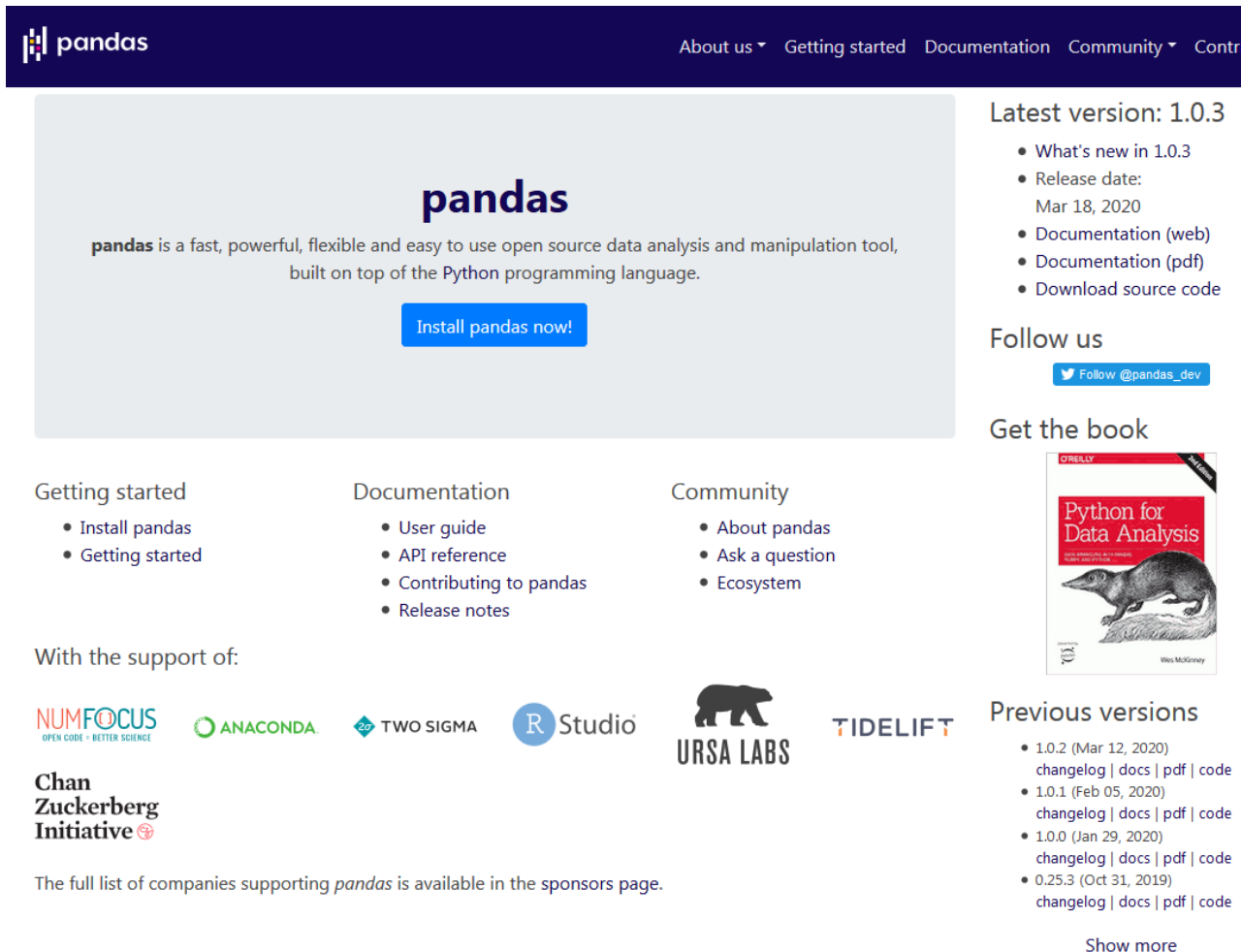
- DataFrame

Pandas 의 특징과 장점

- 통합 인덱싱을 활용한 데이터 조작을 가능하게 하는 데이터프레임(DataFrame) 오브젝트
- 인메모리(in-memory) 데이터 구조와 다양한 파일 포맷들 간의 데이터 읽기/쓰기 환경 지원
- 데이터 결측치의 정렬 및 처리
- 데이터셋의 재구조화 및 피보팅(pivoting)
- 레이블 기반의 슬라이싱, 잘 지원된 인덱싱, 대용량 데이터셋에 대한 서브셋 지원
- 데이터 구조의 칼럼 추가 및 삭제
- 데이터셋의 분할-적용-병합을 통한 GroupBy 엔진 지원
- 데이터셋 병합(merging) 및 조인(joining) 지원
- 저차원 데이터에서의 고차원 데이터 처리를 위한 계층적 축 인덱싱 지원

Official website

➤ <https://pandas.pydata.org/>



The screenshot shows the pandas official website. The header is dark blue with the pandas logo and navigation links: About us, Getting started, Documentation, Community, and Contributing. The main content area has a light blue background with the pandas logo and a description: "pandas is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language." Below this is a blue button that says "Install pandas now!". To the right, there's a section for the "Latest version: 1.0.3" with a list of links: What's new in 1.0.3, Release date: Mar 18, 2020, Documentation (web), Documentation (pdf), and Download source code. Below that is a "Follow us" section with a Twitter link to @pandas_dev. Further down is a "Get the book" section featuring the cover of the book "Python for Data Analysis" by Wes McKinney. At the bottom, there's a "With the support of:" section listing various sponsors: NUMFOCUS, ANACONDA, TWO SIGMA, R Studio, URSA LABS, TIDELIFT, and Chan Zuckerberg Initiative. A link to "Show more" is also present.

pandas

pandas is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language.

[Install pandas now!](#)

Latest version: 1.0.3

- What's new in 1.0.3
- Release date: Mar 18, 2020
- Documentation (web)
- Documentation (pdf)
- Download source code

Follow us

[Follow @pandas_dev](#)

Get the book

With the support of:

NUMFOCUS
OPEN CODE - BETTER SCIENCE

ANACONDA

TWO SIGMA

R Studio

URSA LABS

TIDELIFT

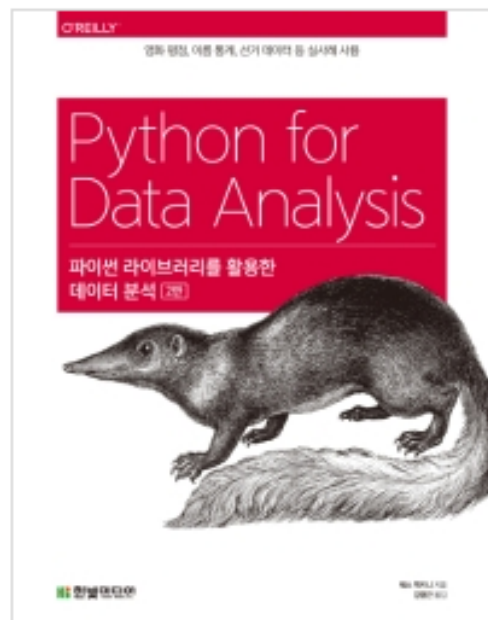
Chan Zuckerberg Initiative

The full list of companies supporting pandas is available in the sponsors page.

Previous versions

- 1.0.2 (Mar 12, 2020)
[changelog](#) | [docs](#) | [pdf](#) | [code](#)
- 1.0.1 (Feb 05, 2020)
[changelog](#) | [docs](#) | [pdf](#) | [code](#)
- 1.0.0 (Jan 29, 2020)
[changelog](#) | [docs](#) | [pdf](#) | [code](#)
- 0.25.3 (Oct 31, 2019)
[changelog](#) | [docs](#) | [pdf](#) | [code](#)

[Show more](#)



크게 보기 | 미리보기

검색

매장 재고 · 위치 >

무료배송 | 이벤트 | 사은품 | 소득공제

파이썬 라이브러리를 활용한 데이터 분석

사례 사용 2판

웨스 맥키니 지음 | 김영근 옮김 | 한빛미디어 | 2019년 05월 20일 출간



9.3(8) | ☆☆☆☆☆ 리뷰 0개

[리뷰쓰기](#)

정가 : 35,000원

판매가 : **31,500원** [10%↓ 3,500원 할인]

통합포인트 : [기본적립] 1,750원 적립 [5% 적립]

[추가적립] 5만원 이상 구매 시 2천원 추가적립 [안내](#)

[회원혜택] 실버등급 이상, 3만원 이상 구매 시 2~4% 추가적립 [안내](#)

추가혜택 : [포인트 안내](#) [도서소득공제 안내](#) [추가혜택 더보기](#)



 확대 보기 |  차례 보기

파이썬 머신러닝 판다스 데이터 분석

저자: 오승환

역자:

구분: 국내서

발행일: 2019년 06월 15일

정가: 25,000원

페이지: 392 페이지

ISBN: 978-89-5674-833-7

출판사: 정보문화사

판형: 187×235

난이도:



초급

초·중급

중급

Youtube

- 허민석 : Pandas 팬더스 강의 기초 실습 14회
- 머신러닝 입문 강좌 | TEAMLAB X

이번 수업에서는

1. Pandas의 1차원 자료. Series
2. Pandas의 2차원 자료. DataFrame

Pandas를 사용하려면

➤ import

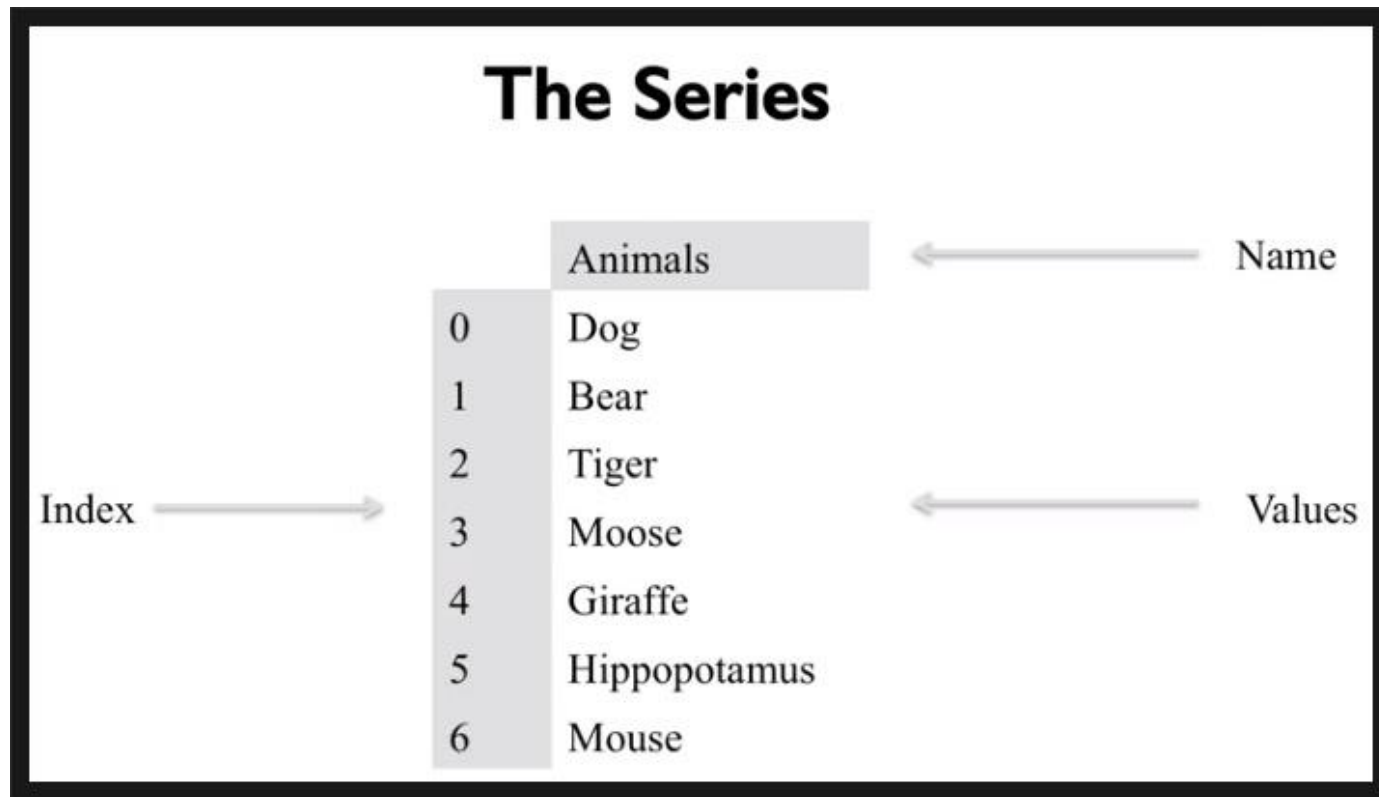
```
import pandas as pd
```

- 모듈(라이브러리)을 호출하여 속성과 메서드를 사용한다

Series 란

Pandas 의 1차원 자료구조.

인덱스(index)와 값(value)로 쌍을 이룸. 딕셔너리와 비슷



Series 만들기

리스트 -> 시리즈

```
import pandas as pd

animals = ['Tigers', 'Bears', 'Moose']
p = pd.Series(animals)
print(p)
```

```
0    Tigers
1     Bears
2     Moose
dtype: object
```

```
numbers = [1,2,3]
nums = pd.Series(numbers)
print(nums)
```

➤ 결과는 어떻게 될까요?

```
0    1
1    2
2    3
dtype: int64
```

pd.Series([1,2,3]) 으로 해도 좋습니다

딕셔너리 -> 시리즈

```
diction = {'a': 1, 'b':2, "c":3}  
print(diction)  # key와 value의 쌍
```

```
d_1 = pd.Series(diction)  
print(d_1)
```

```
a      1  
b      2  
c      3  
dtype: int64
```

딕셔너리란

사전 형태

이름 = {Key1:Value1, Key2:Value2, Key3:Value3, ...}

사전 찾는 법: 이름[key]

사전 = {1:'나', 2:'너', 3:'우리'}

사전[3]

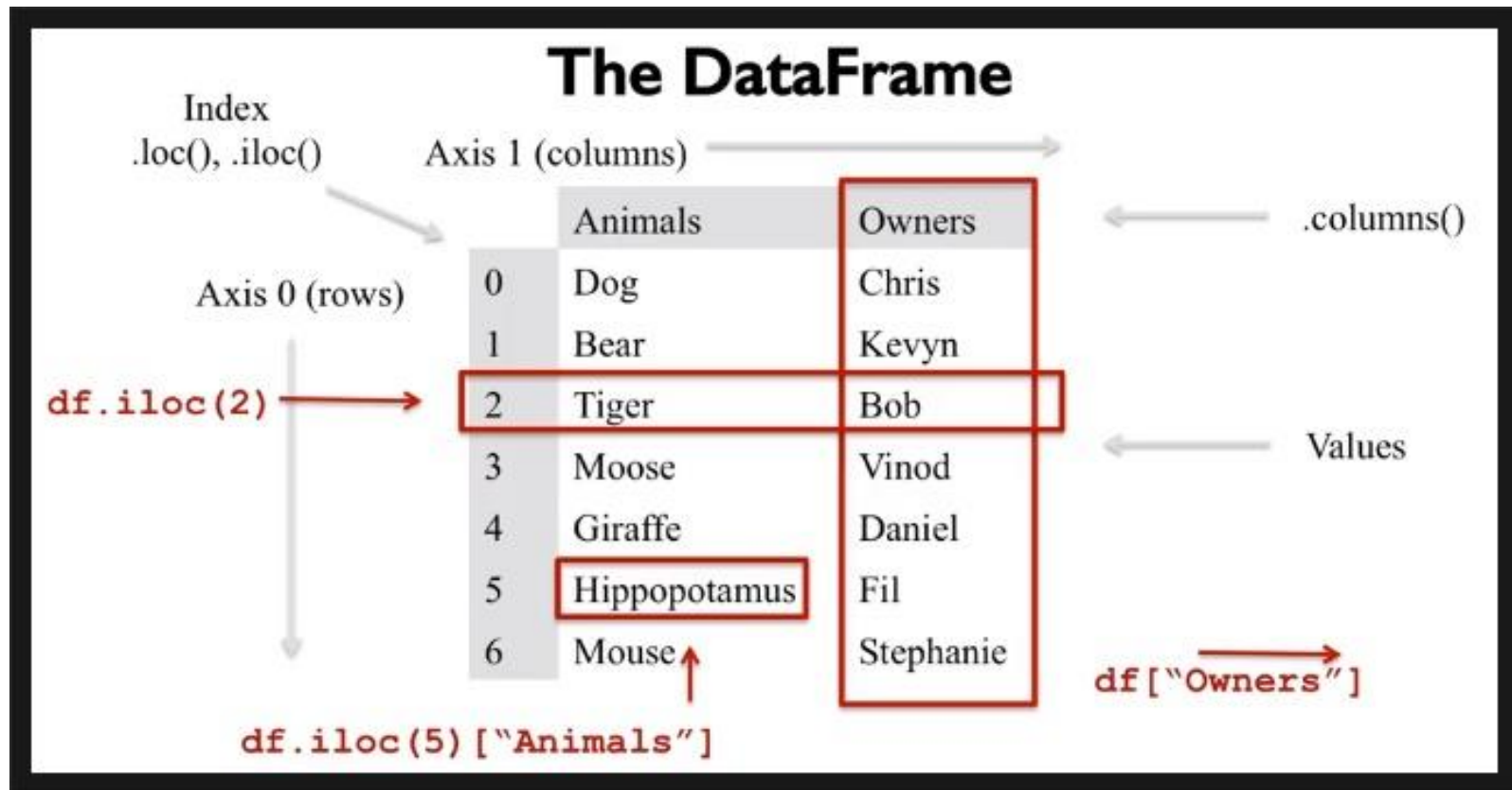
선수 = {"김연아":"피겨스케이팅", "류현진":"야구", "손흥민":"축구"}

선수['김연아']

DataFrame 이란

Pandas 의 2차원 자료구조.

행과 열로 이루어져 있다. 엑셀과 같다. 행렬은 Numpy



DataFrame 만들기

데이터프레임을 만드는 4가지 방법

```
import numpy as np
# 1. Take a 2D array as input to your DataFrame
my_2d_array = np.array([[1, 2, 3], [4, 5, 6]])
pd.DataFrame(my_2d_array)
```

```
# 2. Take a dictionary
my_dict = {"a": ['1', '3'], "b": ['1', '2'], "c": ['2', '4']}
pd.DataFrame(my_dict)
```

	0	1	2
0	1	2	3
1	4	5	6

	a	b	c
0	1	1	2
1	3	2	4

DataFrame 만들기

데이터프레임을 만드는 4가지 방법

3. Take a DataFrame

```
my_df = pd.DataFrame(data=[4,5,6,7], index=range(0,4), columns=['A'])  
pd.DataFrame(my_df)
```

4. Take a Series

```
my_series = pd.Series({"United Kingdom":"London", "India":"New Delhi",  
                        "United States":"Washington", "Belgium":"Brussels"})  
pd.DataFrame(my_series)
```

	A		
0	4	United Kingdom	London
1	5	India	New Delhi
2	6	United States	Washington
3	7	Belgium	Brussels

DataFrame 만들기

- 딕셔너리 -> 데이터 프레임으로

```
pandas.DataFrame(딕셔너리 객체)
```

```
dict_data = {'c0':[1,2,3], 'c1':[4,5,6], 'c2':[7,8,9],  
'c3':[10,11,12], 'c4':[13,14,15]}
```

```
df = pd.DataFrame(dict_data)  
print(type(df))  
print(df)
```

```
<class 'pandas.core.frame.DataFrame'>
```

	c0	c1	c2	c3	c4
0	1	4	7	10	13
1	2	5	8	11	14
2	3	6	9	12	15

DataFrame 만들기

- 2차원배열 -> 데이터 프레임으로: 행과 열이름 정하기

```
pandas.DataFrame(2차원 배열,  
                  index = 행 이름들  
                  columns = 열 이름들)
```

```
df = pd.DataFrame([[15, '남', '남중'], [17, '여', '여중']],  
                  index=['철수', '영희'],  
                  columns=['나이', '성별', '학교'])  
print(df)
```

	나이	성별	학교
철수	15	남	남중
영희	17	여	여중

행과 열 이름 보기

```
df
df.index      #행 인덱스
df.columns    #열 이름
```

```
In [39]: df
```

```
Out[39]:
```

	나이	성별	학교
철수	1	2	3
영희	4	5	6

```
In [40]: df.index
```

```
Out[40]: Index(['철수', '영희'], dtype='object')
```

```
In [41]: df.columns
```

```
Out[41]: Index(['나이', '성별', '학교'], dtype='object')
```

이번 수업에서는

1. 데이터프레임의 행과 열 다루기
 - 삭제 (drop)
 - 선택 (indexing)
 - 이름바꾸기 (rename)

DataFrame 만들기

- 2차원배열 -> 데이터 프레임으로: 행과 열이름 정하기

```
pandas.DataFrame( 2차원 배열,  
                  index = 행 이름들  
                  columns = 열 이름들)
```

```
df = pd.DataFrame([[15, '남', '남중'], [17, '여', '여중']],  
                  index=['철수', '영희'],  
                  columns=['나이', '성별', '학교'])  
print(df)
```

	나이	성별	학교
철수	15	남	남중
영희	17	여	여중

행 삭제하기

```
객체.drop(행이름)  
객체.drop([행1, 행2])  
객체.drop(행이름, axis = 0, inplace = True)
```

```
df.drop('철수')  
df.drop(['철수', '영희'])
```

```
In [55]: df.drop('철수')
```

```
Out[55]:
```

	나이	성별	학교
영희	17	여	여중

```
In [56]: df.drop(['철수', '영희'])
```

```
Out[56]:
```

```
Empty DataFrame
```

```
Columns: [나이, 성별, 학교]
```

```
Index: []
```


열 삭제하기

```
객체.drop( 열이름, axis = 1 )  
객체.drop([열1, 열2], axis = 1 )  
객체.drop( 열이름, axis = 1 , inplace = True)
```

```
df2 = df  
df2.drop('나이', axis = 1)  
df2.drop(['나이', '성별'], axis = 1)  
  
df2.drop(['나이', '성별'], axis = 1, inplace = True)
```

	성별	학교
철수	남	남중
영희	여	여중

	학교
철수	남중
영희	여중

➤ 아래와 같은 데이터프레임을 만들어 봅시다

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

```
exam = {'수학' : [ 90, 80, 70], '영어' : [ 98, 89, 95],  
        '음악' : [ 85, 95, 100], '체육' : [ 100, 90, 90]}
```

```
df = pd.DataFrame(exam, index=['서준', '우현', '인아'])
```

```
print(df)
```

- 4개의 칼럼을 만들고 하나로 합치는 과정

행 선택

```
객체.loc['행이름']    또는  객체.loc[['행이름']]
객체.loc[['행1', '행2']]
객체.iloc[행숫자, inplace = True)
```

```
df.loc['서준']        # loc 인덱서 활용
df.loc[['서준', '우현']]
```

```
df.iloc[0]           # iloc 인덱서 활용
df.iloc[0:2]
```

수학 90
영어 98
음악 85
체육 100

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

수학 90
영어 98
음악 85
체육 100

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

행 선택

객체.iloc[start:stop:step] #slicing

```
df.iloc[0:2]
df.iloc[0:3:2]
df.iloc[ : :2]
df.iloc[ : : -1]
```

수학 영어 음악 체육

서준 90 98 85 100

우현 80 89 95 90

인아 70 95 100 90

수학 영어 음악 체육

서준 90 98 85 100

우현 80 89 95 90

수학 영어 음악 체육

서준 90 98 85 100

인아 70 95 100 90

수학 영어 음악 체육

서준 90 98 85 100

인아 70 95 100 90

수학 영어 음악 체육

인아 70 95 100 90

우현 80 89 95 90

서준 90 98 85 100

열 선택

객체.['열이름'] 또는 객체[['열이름']]

객체[['열1', '열2']]

객체.열이름

```
df['수학']
```

```
df[['음악', '체육']]
```

```
df.수학
```

서준	90
우현	80
인아	70

	음악	체육
서준	85	100
우현	95	90
인아	100	90

서준	90
우현	80
인아	70

행/열 이름바꾸기

객체.index = 새로운 행 이름 리스트
객체.columns = 새로운 열 이름 객체.열이름리스트

```
df
df.index = [ "준", "현", "아"]

df.columns = ["수","영", "음", "체"]
```

수학 영어 음악 체육

서준	90	98	85	100
우현	80	89	95	90
인아	70	95	100	90

수학 영어 음악 체육

준	90	98	85	100
현	80	89	95	90
아	70	95	100	90

수 영 음 체

준	90	98	85	100
현	80	89	95	90
아	70	95	100	90

일부만 이름바꾸기

```
객체.rename(index = {기준이름: 새로운 이름, 기준이름: 새로운 이름, ...})  
객체.rename(columns = {기준이름: 새로운 이름, 기준이름: 새로운 이름, ...})
```

```
df
```

```
df.rename(index = {"서준": "준서", "우현": "현우"})
```

```
df.rename(columns = {"수학": "math", "영어": "English"})
```

	수학	영어	음악	체육
--	----	----	----	----

서준	90	98	85	100
----	----	----	----	-----

우현	80	89	95	90
----	----	----	----	----

인아	70	95	100	90
----	----	----	-----	----

	수학	영어	음악	체육
--	----	----	----	----

준서	90	98	85	100
----	----	----	----	-----

현우	80	89	95	90
----	----	----	----	----

인아	70	95	100	90
----	----	----	-----	----

	math	English	음악	체육
--	------	---------	----	----

서준	90	98	85	100
----	----	----	----	-----

우현	80	89	95	90
----	----	----	----	----

인아	70	95	100	90
----	----	----	-----	----

연습

```
import plotly.express as px
df = px.data.gapminder()
df.head()
df.shape
```

	country	continent	year	lifeExp	pop	gdpPercap	iso_alpha	iso_num
0	Afghanistan	Asia	1952	28.801	8425333	779.445314	AFG	4
1	Afghanistan	Asia	1957	30.332	9240934	820.853030	AFG	4
2	Afghanistan	Asia	1962	31.997	10267083	853.100710	AFG	4
3	Afghanistan	Asia	1967	34.020	11537966	836.197138	AFG	4
4	Afghanistan	Asia	1972	36.088	13079460	739.981106	AFG	4

```
df.shape
```

```
(1704, 8)
```


연습

- df 에서 각행이 100, 200, 300, 400, 500 번째 데이터만 뽑아서 df라는 이름으로 저장해 주세요.

	country	continent	year	lifeExp	pop	gdpPercap	iso_alpha	iso_num
100	Bangladesh	Asia	1972	45.252	70759295	630.233627	BGD	50
200	Burkina Faso	Africa	1992	50.260	8878303	931.752773	BFA	854
300	Colombia	Americas	1952	50.643	12350771	2144.115096	COL	170
400	Czech Republic	Europe	1972	70.290	9862158	13108.453600	CZE	203
500	Eritrea	Africa	1992	49.991	3668440	582.858510	ERI	232

```
df = df.iloc[[100, 200, 300, 400, 500]]
df
```

연습

- df 에서 gdpPercap , iso_alpha, iso_num 을 삭제해 주세요

	country	continent	year	lifeExp	pop
100	Bangladesh	Asia	1972	45.252	70759295
200	Burkina Faso	Africa	1992	50.260	8878303
300	Colombia	Americas	1952	50.643	12350771
400	Czech Republic	Europe	1972	70.290	9862158
500	Eritrea	Africa	1992	49.991	3668440

```
df.drop(["gdpPercap", "iso_alpha", "iso_num"], axis =1, inplace= True)
```

연습

- 행의 이름을 각각 A, B, C, D, E 로 바꾸어 봅시다.

	country	continent	year	lifeExp	pop
A	Bangladesh	Asia	1972	45.252	70759295
B	Burkina Faso	Africa	1992	50.260	8878303
C	Colombia	Americas	1952	50.643	12350771
D	Czech Republic	Europe	1972	70.290	9862158
E	Eritrea	Africa	1992	49.991	3668440

```
df.index = ["A", "B", "C", "D", "E"]  
df
```

연습

- 열의 이름 중에서 continent 는 conti 로, lifeExp 는 life 로 각각 바꾸어 주세요.

	country	conti	year	life	pop
A	Bangladesh	Asia	1972	45.252	70759295
B	Burkina Faso	Africa	1992	50.260	8878303
C	Colombia	Americas	1952	50.643	12350771
D	Czech Republic	Europe	1972	70.290	9862158
E	Eritrea	Africa	1992	49.991	3668440

```
df.rename(columns = {"continent":"conti", "lifeExp":"life"}, inplace = True )
```

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 numpy가 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          7.815959
cylinders     1.701131
displacement 104.269811
horsepower    38.402014
weight      846.841719
acceleration   2.757678
model-year     3.697656
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          61.082965
cylinders     2.893851
displacement 10872.234567
horsepower   1474.115557
weight      717140.896208
acceleration   7.603789
model-year    13.664792
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