

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

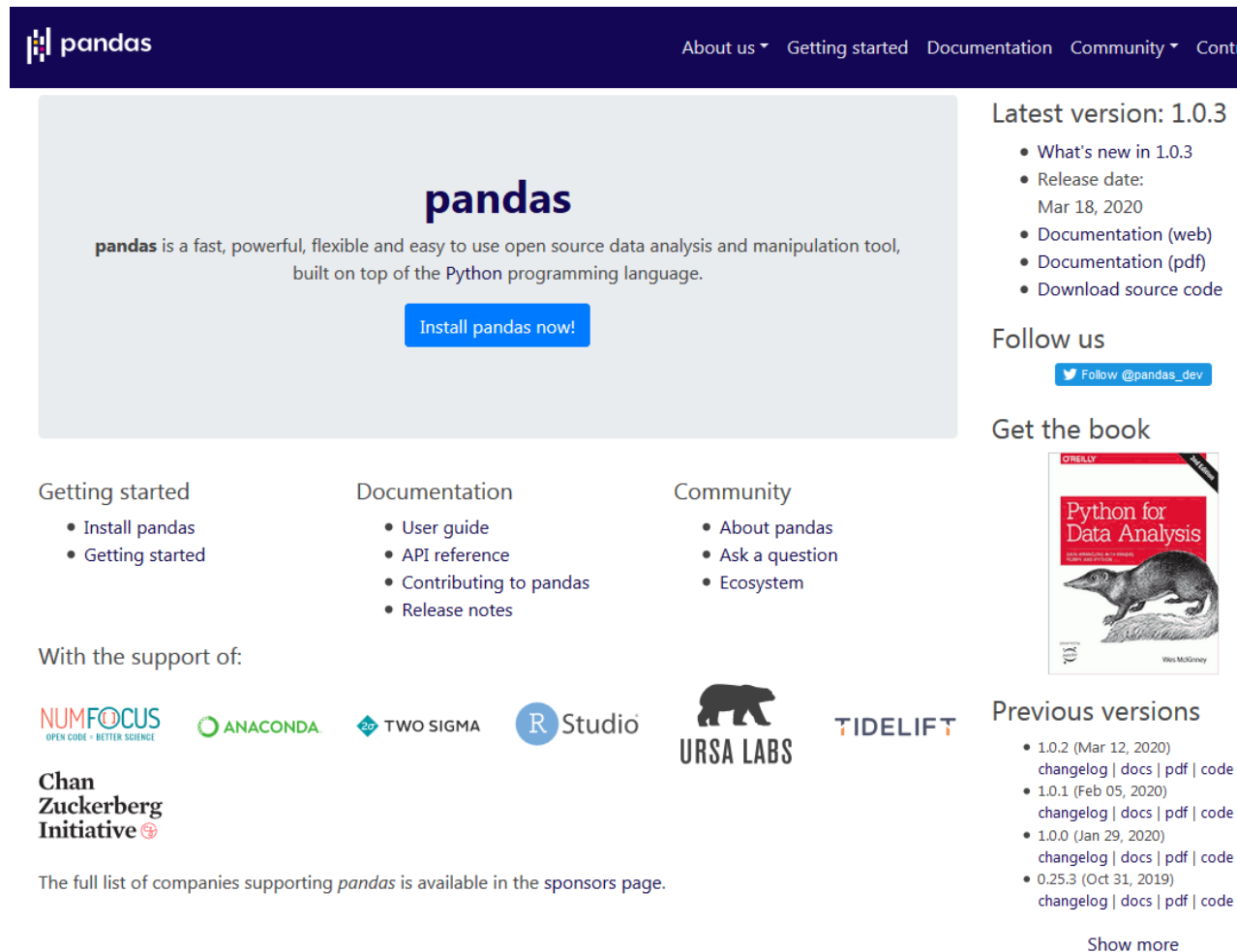
- 통합 인덱싱을 활용한 데이터 조작을 가능하게 하는 데이터프레임(DataFrame) 오브젝트
- 인메모리(in-memory) 데이터 구조와 다양한 파일 포맷들 간의 데이터 읽기/쓰기 환경 지원
- 데이터 결측치의 정렬 및 처리
- 데이터셋의 재구조화 및 피보팅(pivoting)
- 레이블 기반의 슬라이싱, 잘 지원된 인덱싱, 대용량 데이터셋에 대한 서브셋 지원
- 데이터 구조의 칼럼 추가 및 삭제
- 데이터셋의 분할-적용-병합을 통한 GroupBy 엔진 지원
- 데이터셋 병합(merging) 및 조인(joining) 지원
- 저차원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

Library Highlights

- A fast and efficient **DataFrame** object for data manipulation with integrated indexing;
- Tools for **reading and writing data** between in-memory data structures and different formats: CSV and text files, Microsoft Excel, SQL databases, and the fast HDF5 format;
- Intelligent **data alignment** and integrated handling of **missing data**: gain automatic label-based alignment in computations and easily manipulate messy data into an orderly form;
- Flexible **reshaping** and pivoting of data sets;
- Intelligent label-based **slicing, fancy indexing**, and **subsetting** of large data sets;
- Columns can be inserted and deleted from data structures for **size mutability**;
- Aggregating or transforming data with a powerful **group by** engine allowing split-apply-combine operations on data sets;
- High performance **merging and joining** of data sets;
- **Hierarchical axis indexing** provides an intuitive way of working with high-dimensional data in a lower-dimensional data structure;
- **Time series**-functionality: date range generation and frequency conversion, moving window statistics, date shifting and lagging. Even create domain-specific time offsets and join time series without losing data;
- Highly **optimized for performance**, with critical code paths written in Cython or C.

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 Contr. 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 links for "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 "@pandas_dev". Further down is a "Get the book" section featuring the cover of "Python for Data Analysis" by Wes McKinney. At the bottom, there's a "Previous versions" section listing versions 1.0.2, 1.0.1, 1.0.0, and 0.25.3, each with links for changelog, docs, pdf, and code. A "Show more" link is at the bottom right. On the left side of the bottom section, there are logos for sponsors: NUMFOCUS, ANACONDA, TWO SIGMA, R Studio, URSA LABS, and TIDELIFT. Below these is the Chan Zuckerberg Initiative logo and a note that the full list of companies supporting pandas is available in the sponsors page.

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!](#)

Getting started

- Install pandas
- Getting started

Documentation

- User guide
- API reference
- Contributing to pandas
- Release notes

Community

- About pandas
- Ask a question
- Ecosystem

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.

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

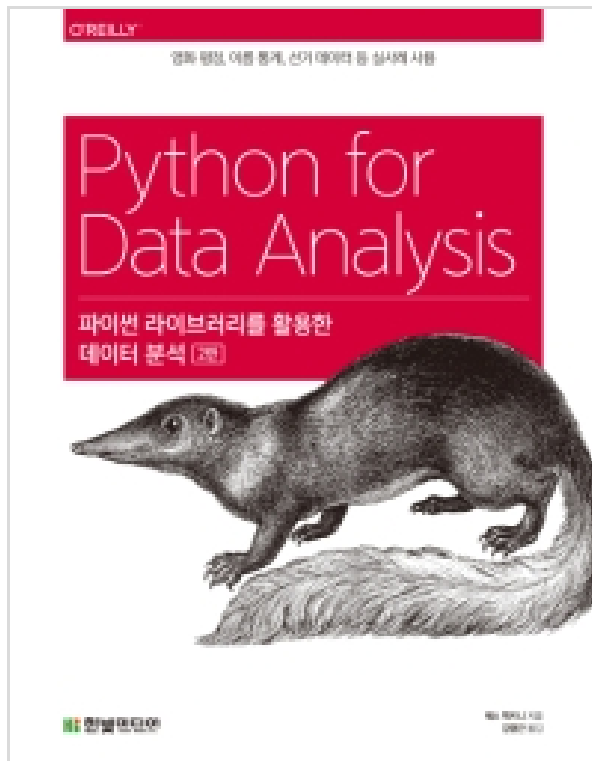
Python for Data Analysis

Wes McKinney

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](#) | [차레 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

이번 수업에서는

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

Pandas를 사용하려면

➤ import

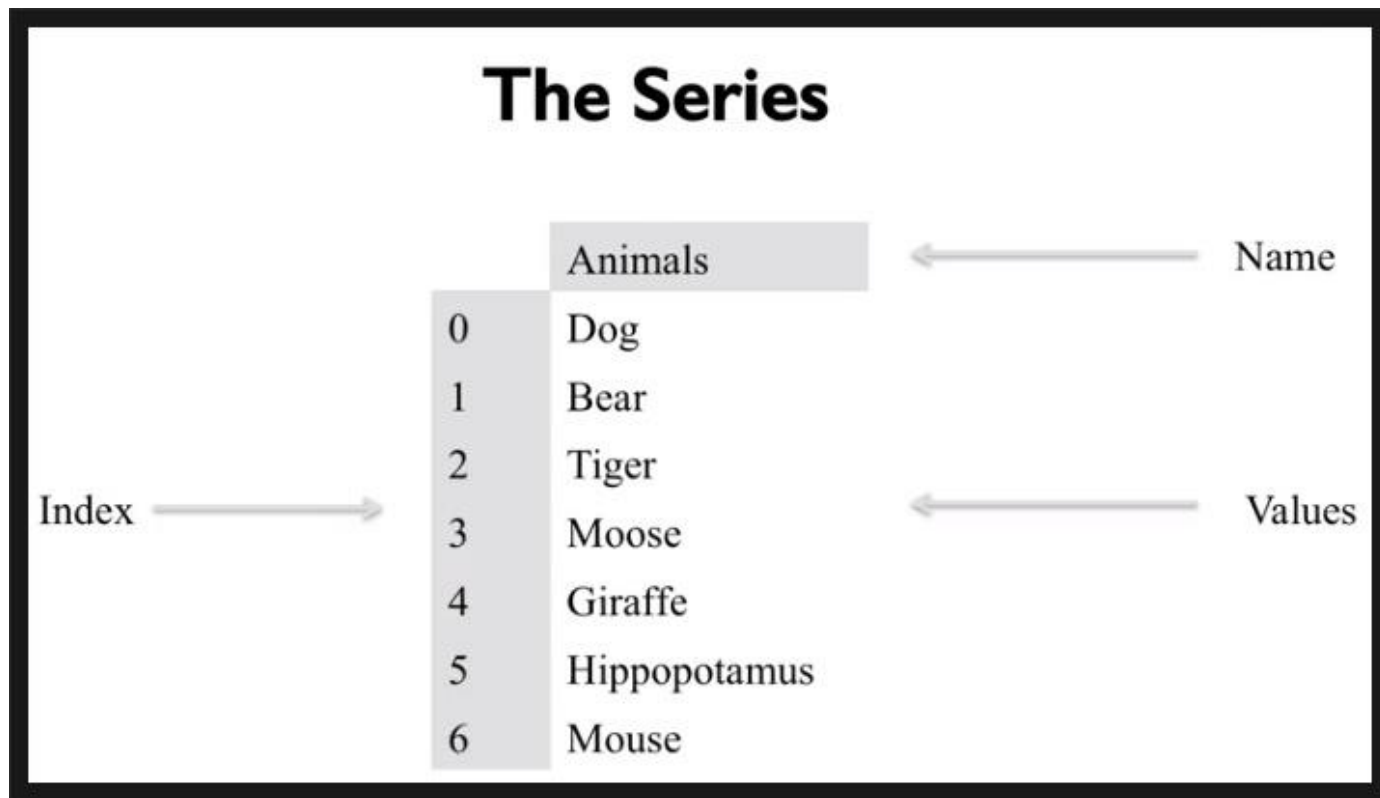
```
import pandas as pd
```

- 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]

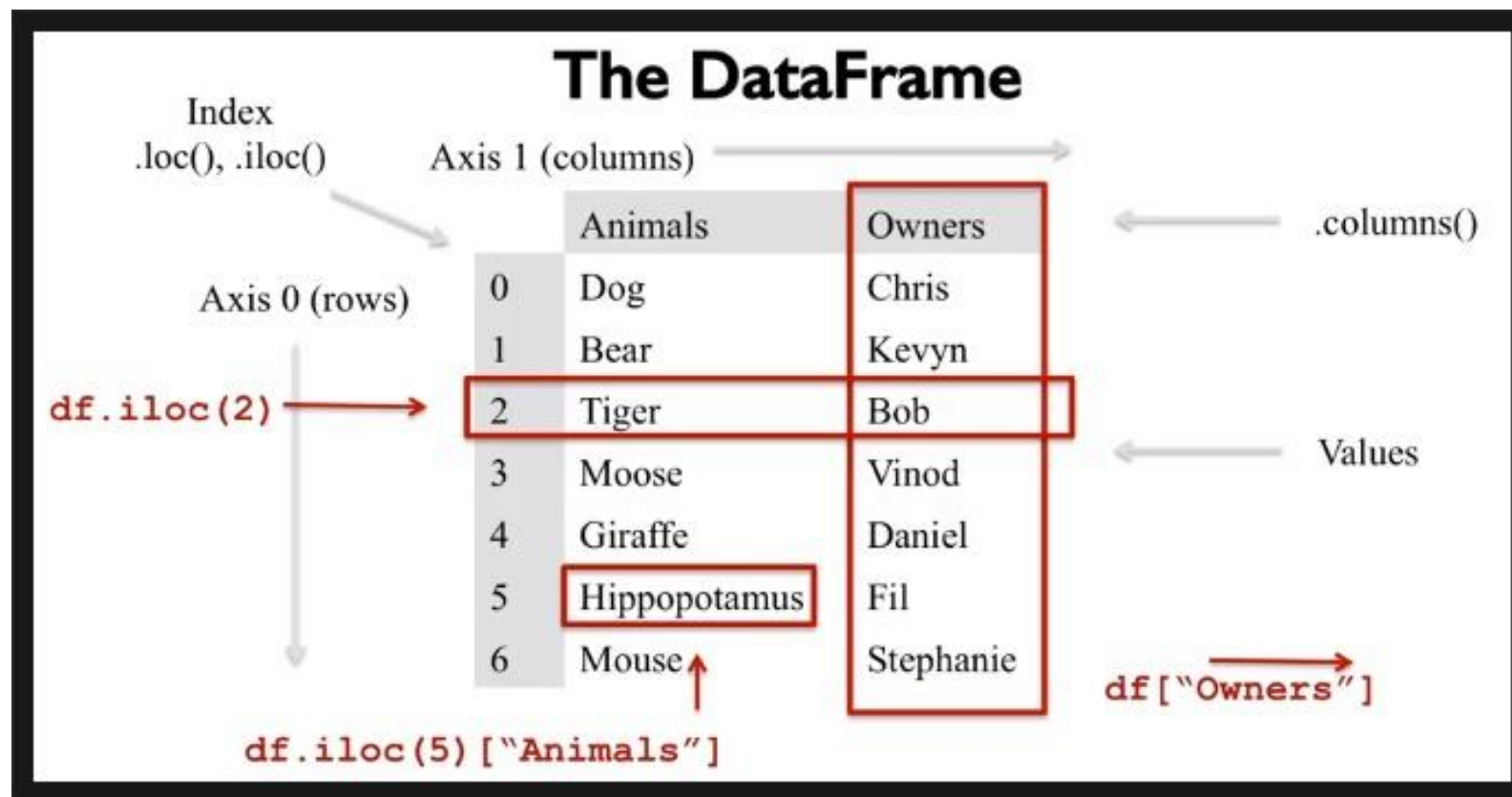
선수 = {"김연아":"피겨스케이팅", "류현진":"야구", "손흥민":"축구"}

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
1	5
2	6
3	7

United Kingdom	London
India	New Delhi
United States	Washington
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]
```

```
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	음악	체육	서준	90
우현	80	85	100	우현	80
인아	70	95	90	인아	70
		100	90		

행/열 이름바꾸기

```
객체.index = 새로운 행 이름 리스트  
객체.columns = 새로운 열 이름 객체.열이름 리스트
```

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

```
df  
df.rename(index = {"서준": "준서", "우현": "현우"})  
  
df.rename(columns = {"수학": "math", "영어": "English"})
```

	수학	영어	음악	체육		수학	영어	음악	체육		math	English	음악	체육
서준	90	98	85	100	준서	90	98	85	100	서준	90	98	85	100
우현	80	89	95	90	현우	80	89	95	90	우현	80	89	95	90
인아	70	95	100	90	인아	70	95	100	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 )
```