

# Pandas 기초

발표자 : 최강규



# CONTENTS

- 01 데이터와 Pandas
- 02 시리즈
- 03 데이터프레임
  - 생성, 추가, 삭제 등
- 04 데이터프레임 활용
  - 여러가지 함수



열: 세로,  
데이터의 성격, 속성



행: 가로,  
데이터

ID	Name	AGE	SEX
01	KIM	32	M
02	LEE	26	F
03	PARK	72	F
04	CHOI	15	M

01

## 데이터

비정형 데이터: 형태가 정의되지 않았다는 것을 의미  
정형데이터로 만들어야 모델 생성할 때 사용 가능

정형 데이터: 형태가 정의되어 정해진 데이터로 존재



01

## Pandas란

관계형 또는 레이블이 된 데이터(예를 들어 엑셀 데이터)로 쉽고 직관적으로 작업할 수 있도록 설계되었고 빠르고, 유연한 데이터 구조를 제공하는 Python 라이브러리

# 01 라이브러리



import: 불러오기

`import pandas as pd`

: pd라는 짧은 이름으로 pandas라는 라이브러리를 불러오기하겠다는 뜻

라이브러리 종류

`import numpy as np`

: 다차원 배열을 쉽고 효율적으로 처리하게 해주며 수학 계산을 위한 라이브러리

`import matplotlib.pyplot as plt`

: 시각화를 위한 라이브러리

## 02 시리즈



pd.Series()

```
numbers=[1,2,3]
```

```
pd.Series(numbers)
```

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

1차원 구조

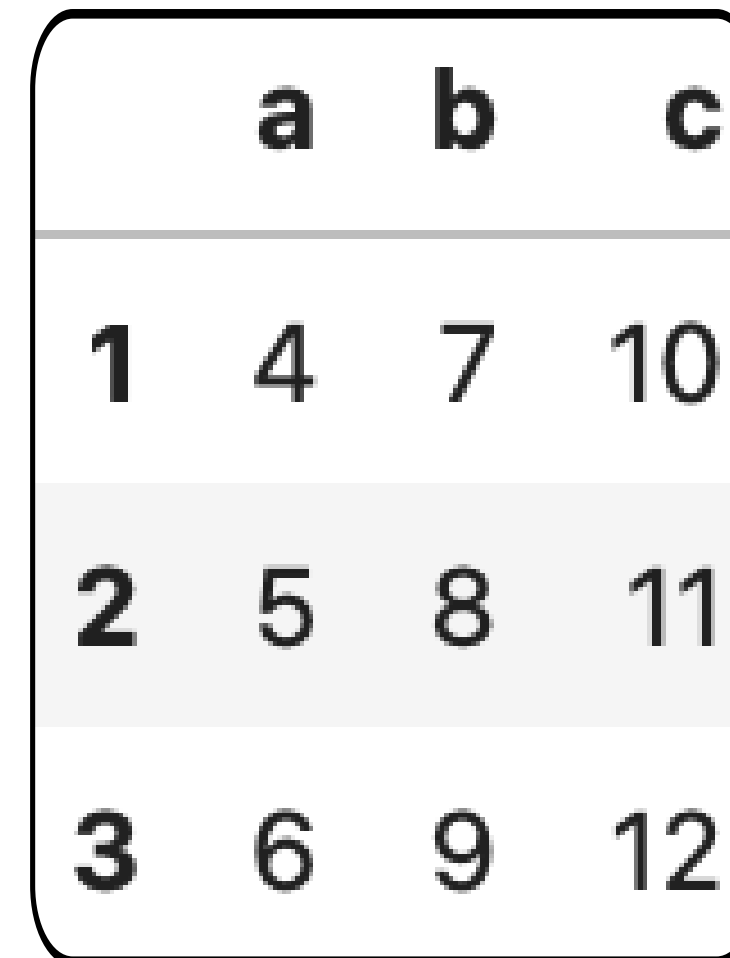
### 03 데이터프레임

pd.DataFrame()

```
df = pd.DataFrame(  
    {"a" : [4, 5, 6],  
     "b" : [7, 8, 9],  
     "c" : [10, 11, 12]},  
    index = [1, 2, 3])
```

index: 책의 목차와 같은 역할, 변경 가능

인덱스



	a	b	c
1	4	7	10
2	5	8	11
3	6	9	12

2차원 구조

## 03 데이터프레임



### 열 추출

`df['a']`

대괄호 하나 사용

1차원 구조의 Series

```
1    4
2    5
3    6
Name: a
```

`df[['a']]`

대괄호를 하나 더 사용하면

데이터프레임 형태

```
   a
1  4
2  5
3  6
```



## 03 데이터프레임



시리즈 -> 데이터프레임

```
numbers=[1,2,3]  
a=pd.Series(numbers)  
pd.DataFrame(a)
```

	0
0	1
1	2
2	3

### 03 데이터프레임

```
df=pd.DataFrame([{'Name': '홍길동',  
                  'Phone_no': '01011112222',  
                  '1Q_point': 90},  
                 {'Name': '김수진',  
                  'Phone_no': '01012341234',  
                  '1Q_point': 70},  
                 {'Name': '홍길동',  
                  'Phone_no': '01099998888',  
                  '1Q_point': 22.5}],  
                 index=['10001','10002','10003'])
```

```
customer_1=pd.Series({'Name': '홍길동',  
                      'Phone_no': '01011112222',  
                      '1Q_point': 90})
```

```
customer_2=pd.Series({'Name': '김수진',  
                      'Phone_no': '01012341234',  
                      '1Q_point': 70})
```

```
customer_3=pd.Series({'Name': '홍길동',  
                      'Phone_no': '01099998888',  
                      '1Q_point': 22.5})
```

```
df=pd.DataFrame([customer_1, customer_2, c  
customer_3], index=['10001','10002','10003'])
```

시리즈 여러개를 []를 사용해 한꺼번에 데이터프레임화

### 03 데이터프레임

#### 인덱스를 활용한 추출

`df.iloc[0]`

iloc: 인덱스 순서 이용

`df.loc['10001']`

loc: 인덱스 번호 이용

	Name	Phone_no	1Q_point
10001	홍길동	01011112222	90.0
10002	김수진	01012341234	70.0
10003	홍길동	01099998888	22.5

```
Name                홍길동
Phone_no            01011112222
1Q_point            90.0
Name: 10001, dtype: object
```

## 03 데이터프레임



### 인덱스를 활용한 추출

`df.iloc[0:2]`

a:b -> 인덱스 순서 a부터 b-1 출력

	Name	Phone_no	1Q_point
10001	홍길동	01011112222	90.0
10002	김수진	01012341234	70.0

`df.iloc[:,:]`

[행,열], 숫자를 안적으면 전체 출력

	Name	Phone_no	1Q_point
10001	홍길동	01011112222	90.0
10002	김수진	01012341234	70.0
10003	홍길동	01099998888	22.5

## 03 데이터프레임



### 조건에 맞는 데이터 추출

```
df['Name']=='홍길동'
```

Name 열에 있는 홍길동 유무에 따른 인덱스별 참 거짓 출력

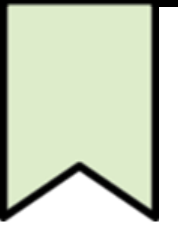
10001	True
10002	False
10003	True

```
df[df['Name']=='홍길동']
```

참거짓 판단된 데이터를 [] 안에 넣으면 참인 데이터만 출력

	Name	Phone_no	1Q_point
10001	홍길동	01011112222	90.0
10003	홍길동	01099998888	22.5

## 03 데이터프레임



### 조건에 맞는 데이터 추출

```
df[df['1Q_point']>50]
```

```
df[(df['Name']=='홍길동')&(df['1Q_point']>50)]
```

논리연산자 활용 가능

	Name	Phone_no	1Q_point
10001	홍길동	01011112222	90.0
10002	김수진	01012341234	70.0

	Name	Phone_no	1Q_point
10001	홍길동	01011112222	90.0

## 03 데이터프레임

삭제: 아래 코드는 저장이 되는 것X, 유지하기 위해서는 삭제한 데이터를 따로 저장하거나 `inplace=True` 사용

```
df.drop('10001')
```

인덱스를 이용한 행 삭제

```
df.drop(['10001','10002'])
```

대괄호로 묶어서 여러개의 행이나 열 삭제 가능

```
df.drop(columns='Name')
```

columns를 이용한 열 삭제

	Name	Phone_no	1Q_point
10002	김수진	01012341234	70.0
10003	홍길동	01099998888	22.5

	Name	Phone_no	1Q_point
10003	홍길동	01099998888	22.5

	Phone_no	1Q_point
10001	01011112222	90.0
10002	01012341234	70.0
10003	01099998888	22.5

### 03 데이터프레임



열 이름 변경: 마찬가지로 저장X

```
df=df.rename(columns={'1Q_point':'point'})
```

1Q\_point가 point로 변경됨

	Name	Phone_no	point
10001	홍길동	01011112222	90.0
10002	김수진	01012341234	70.0
10003	홍길동	01099998888	22.5



### 03 데이터프레임

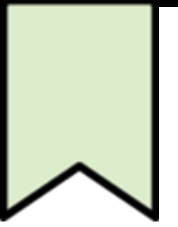


열 추가: `df['새로운 열 이름']=[내용]`

```
df['2Q_point']=[80,40,100]
```

	Name	Phone_no	1Q_point	point2	2Q_point
10001	홍길동	01011112222	90.0	80	80
10002	김수진	01012341234	70.0	40	40
10003	홍길동	01099998888	22.5	100	100

## 03 데이터프레임



### 데이터프레임 합치기

데이터 프레임 2개 생성

```
df_customer = pd.DataFrame([{'Name': '홍길동', 'Phone_no': '01011112222'},  
                             {'Name': '김수진', 'Phone_no': '01012341234'},  
                             {'Name': '안은영', 'Phone_no': '01099998888'}],  
                             index=['20190001', '20190002', '20190003'])  
  
df_point = pd.DataFrame([{'Name': '김개똥', '1Q_point': 90, 'final_point': 80},  
                          {'Name': '홍길동', '1Q_point': 70, 'final_point': 55},  
                          {'Name': '이가은', '1Q_point': 22.5, 'final_point': 95.5}],  
                          index=['20190002', '20190001', '20193333'])
```

## 03 데이터프레임



### 데이터프레임 합치기

1. 합집합: 모든 데이터를 출력, 해당 데이터가 없으면 NaN

```
pd.merge(df_customer, df_point, how='outer', left_index=True, right_index=True)
```

2. 교집합: 겹치는 데이터 출력

```
pd.merge(df_customer, df_point, how='inner', left_index=True, right_index=True)
```

3. 왼쪽(먼저)에 입력한 프레임 기준

```
pd.merge(df_customer, df_point, how='left', left_index=True, right_index=True)
```

4. 오른쪽 기준

```
pd.merge(df_customer, df_point, how='right', left_index=True, right_index=True)
```

## 03 데이터프레임



### 데이터프레임 합치기

#### 1. 합집합

	Name_x	Phone_no	Name_y	1Q_point	final_point
20190001	홍길동	01011112222	홍길동	70.0	55.0
20190002	김수진	01012341234	김개똥	90.0	80.0
20190003	안은영	01099998888	NaN	NaN	NaN
20193333	NaN	NaN	이가은	22.5	95.5

#### 2. 교집합

	Name_x	Phone_no	Name_y	1Q_point	final_point
20190001	홍길동	01011112222	홍길동	70.0	55.0
20190002	김수진	01012341234	김개똥	90.0	80.0

#### 3. 왼쪽 기준

	Name_x	Phone_no	Name_y	1Q_point	final_point
20190001	홍길동	01011112222	홍길동	70.0	55.0
20190002	김수진	01012341234	김개똥	90.0	80.0
20190003	안은영	01099998888	NaN	NaN	NaN

#### 4. 오른쪽 기준

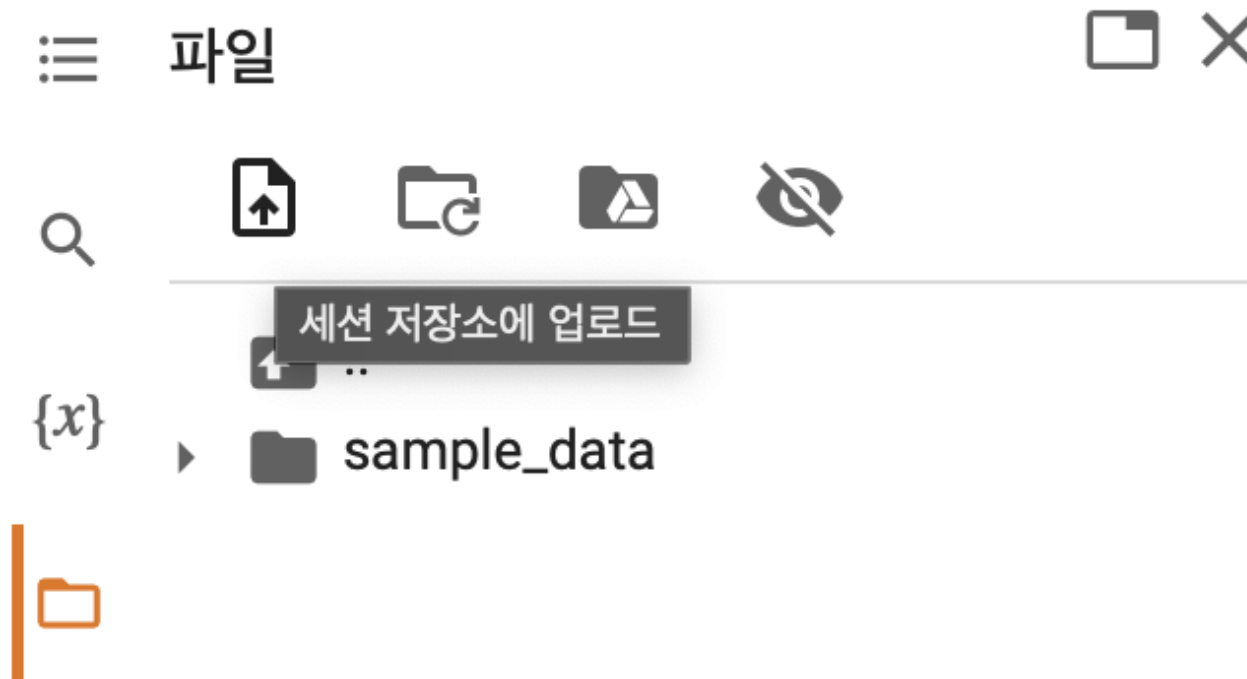
	Name_x	Phone_no	Name_y	1Q_point	final_point
20190002	김수진	01012341234	김개똥	90.0	80.0
20190001	홍길동	01011112222	홍길동	70.0	55.0
20193333	NaN	NaN	이가은	22.5	95.5

## 04 데이터프레임 활용

### 파일 불러오기

엑셀파일 불러올 때, `pd.read_excel()` 사용

csv 파일 불러올 때, `pd.read_csv()` 사용



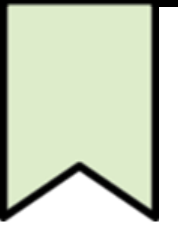
우측 메뉴 중 파일 클릭 -> 첫번째 메뉴(문서에 화살표 아이콘) 클릭 -> 불러올 파일 선택

업로드 후 해당 파일 우클릭 -> 경로 복사

`pd.read_csv(복사한 경로 붙여넣기)`

```
df=pd.read_excel("olympics_organized-1.xlsx")
```

## 04 데이터프레임 활용



### 기본 정보 확인

`df.shape()`

행, 열 개수 확인

`df.info()`

데이터 타입과 같은 정보 확인

`df.describe()`

수치 데이터 범위, 사분위값, 평균 확인

## 04 데이터프레임 활용



결측치: 데이터에 값이 없는 것(NaN, Na, Null, None)

`df.isnull()`

결측값 여부 출력

`df.isnull().sum()`

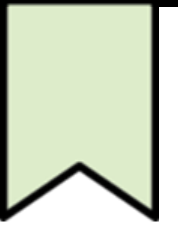
sum()과 함께 써 결측값 개수를 한번에 파악할 수 있음

sum: 합산, true=1이므로 true인 데이터의 개수 파악 가능

`df.dropna()` or `df.dropna(axis=0)`: 결측값이 있는 행 제거

`df.dropna(axis=1)`: 열 제거

## 04 데이터프레임 활용



### 여러 함수

`df.unique()`

특정 칼럼의 유일한 값 확인

`df.sort_values()`

특정 칼럼의 값을 오름차순으로 정렬

`ascending=False`을 괄호 안에 넣으면 내림차순 정렬

`df.value_counts()`

카테고리별 데이터 값 세기



## + 실습 진행

