

# Data Analysis with Pandas

Pandas 기초 I

Jinwook Seo, Ph.D.

Human-Computer Interaction Lab

Department of Computer Science and Engineering Seoul National  
University

지난 강의 내용 • NumPy 배열 연산 • NumPy 배열의 집계  
함수 • Python을 활용한 DB 연동

# 강의 내용 • Pandas 소개 • Series

- DataFrame
- 인덱싱
- NumPy를 기반으로 만들어진 패키지 • Label이 붙은 데이터를 위한 편리한 인터페이스 제공
- 스프레드시트 프로그램 사용자에게 익숙한 환경

## Pandas



- <https://pandas.pydata.org/>

# pandas

$$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$$



[home](#) // [about](#) // [get pandas](#) // [documentation](#) // [community](#) // [talks](#) // [donate](#)

## Python Data Analysis Library

*pandas* is an open source, BSD-licensed library providing high-performance, easy-to-use data structures and data analysis tools for the [Python](#) programming language.

*pandas* is a [NumFOCUS](#) sponsored project. This will help ensure the success of development of *pandas* as a world-class open-source project, and makes it possible to [donate](#) to the project.

### VERSIONS

#### Release

0.22.0 - December 2017

[download](#) // [docs](#) // [pdf](#)

#### Development

0.23.0 - 2018

[github](#) // [docs](#)

(실습) Pandas 버전 확인 • Pandas를 import 후

version으로 확인

- as 문을 사용해서 축약어 활용 가능

```
In [1]: import pandas as pd  
pd.__version__
```

```
Out[1]: '0.22.0'
```

- NumPy의 ndarray의 한계 • 데이터에 레이블을 붙이려면  
번거로움
- 누락된 데이터로 작업하기 어려움 • 요소 단위의 브로드캐스팅을  
벗어나는 연산이 어려움 • 그룹화, 피벗 등

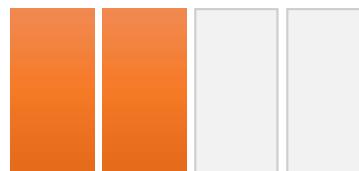
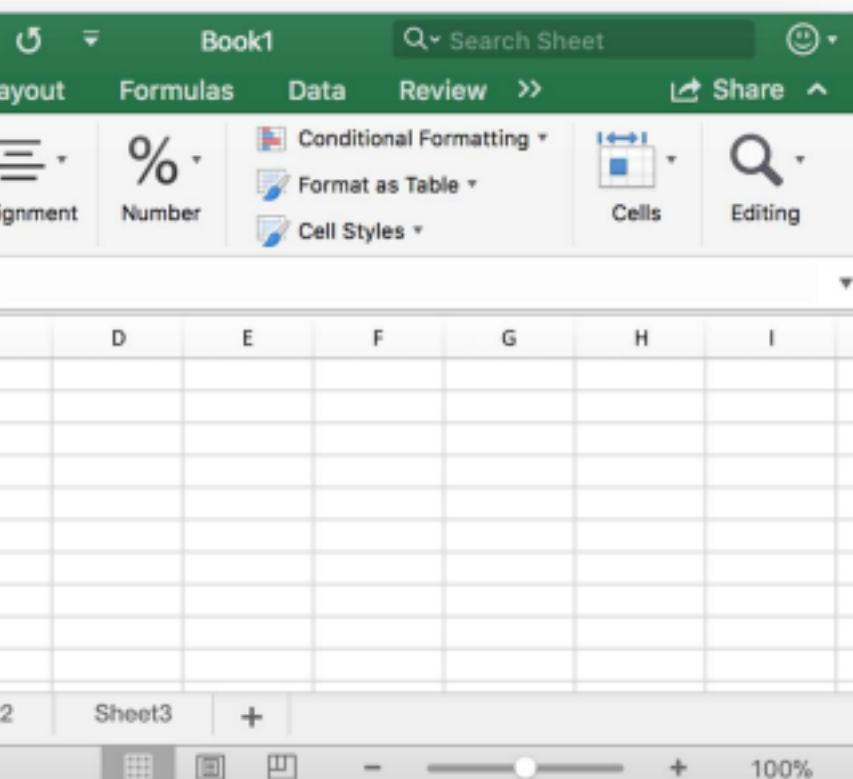
## Pandas

- Pandas의 Series와 DataFrame을 사용하여 극복 가능

# Pandas와 Excel의 비교

hci lab  
HUMAN COMPUTER INTERACTION

## Panel



Series



DataFrame

- NumPy의 구조화된 배열의 확장 • Series의 선언

## (실습) Series



# Series의 구조

- Series 내부의 데이터는 NumPy 배열을 사용해서 처리
- Index는 별도의 형태로 관리



Series의 구조 • 1차원의 데이터를 관리 • Python의

dict처럼 index와 value가 연계되어 처리 index values

data: 실제 값

index: 데이터를 접근할 정보

dtype: 데이터들의 타입

1 2 3

name: Series 인스턴스의 이름

## Series의 생성

- Series를 생성하면서 data, index, name 등을 지정 가능
- 아래의 예시에서는 Index를 지정
- Series의 dimension과 shape은 NumPy와 동일한 형식

index values

'a'

'b'

'c'

- 연속하지 않은 값들도 사용 가능 • 인덱스를 사용해서 저장된 값에 접근 가능

## Series의 Index





- Python의 Dictionary로부터 Series 생성 가능
- Dictionary의 key들이 index로 사용 됨

# Dictionary와 Series



- 값을 가져오는 연산의 경우 Dictionary와 Series 모두 동일•



그러나 Series의 경우

슬라이싱 연산 지원

Dictionary와 Series





## Series의 슬라이싱

- 숫자 기반의 슬라이싱과는 달리 끝 인덱스에 해당하는 값을 포함



## (실습) Series의 슬라이싱

- 문자열로 슬라이싱을 할 경우 인덱스의 순서는 입력된 순서





## (실습) Series의 슬라이싱

- 숫자로 슬라이싱을 할 경우?



- 배열과 유사하게 동작
- 표준 인덱싱 표기법을 사용해서 값이나 슬라이스 처리

## Pandas의 Index

•



그러나



내부의 값을 직접 변경 불가

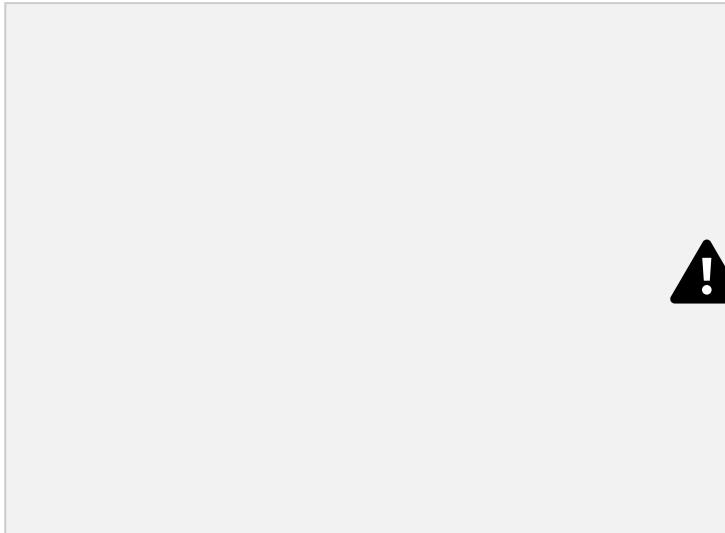


## Pandas Index의 활용

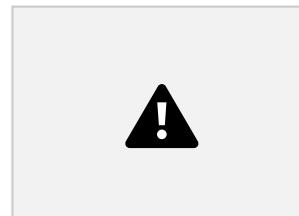
- 집합 연산을 활용하여 필요한 index 도출 가능 • &: 교집합, |:

## 합집합

- ^:



두 집합의 상대 여집합의  
합집합



# Series의 데이터 접근

- Series 이름을 쓸 경우 전체

## 데이터 반환

- `head( )`: 첫 5개의 행을 반환
- `tail( )`: 마지막 5개의 행을 반환





## Series의 데이터 검색

- 데이터의 검색 방법은 Python dictionary의 표현식과 유사
- ‘in’



구문을 사용하여

key들 중에서 찾고자 하는 값이 있는지 확인

Series의 데이터 추가 • 데이터를 추가하는 방법도

dictionary와 유사

- 새로운 key를 지정하고, 해당 키에 값을 저장



## 정수 인덱스를 사용할 경우의 해석

- 정수 인덱스를 사용할 경우 혼동이 야기될 가능성 높음 • 1번을



key로 갖는 value? vs 1번째  
value?

정수 인덱스를 사용할 경우의 해석 명시적  
인덱스

암묵적 인덱스





인덱서(loc, iloc, ix)의 활용 • 정수 인덱싱을 사용할 때 발생할 혼선을 방지하기 위하여 인덱싱 방식을 명시 가능 •

인덱서(indexer) 속성 제공 • loc: 항상 명시적 인덱스를 참조 • iloc: 암묵적인 python 스타일의 인덱스를 참조  
• ix: 위의 두 가지 속성의 하이브리드 형태  
• DataFrame 파트 참고

인덱서(loc, iloc, ix)의 활용

• loc 속성을 사용할 경우 명시적인 key값을 기준으로 함.





슬라이싱을 할 때도 명시적  
key값을 기준으로 처리

## 인덱서(loc, iloc, ix)의 활용

- iloc 속성을 사용할 경우 암묵적인 python 스타일의 인덱스 참조



DataFrame의 구조 • 여러 개의 Series들이

각각 column이

되어 DataFrame에 속하는 형태

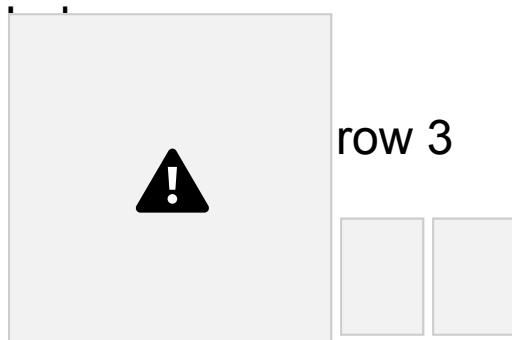
col 1 col 2 col 3

DataFrame

Column

row 1

row 2



row 3

Series

(실습) DataFrame의 생성





DataFrame의 속성

- 각 행에 접근할 수 있는 `index` 속성
- 각 `Series`의 `label`에 접근할 수 있는 `columns` 속성•



위의 속성들을

활용하여 행과 열에 모두 접근 가능

## DataFrame의 자료 접근

- Series의 조회
- 



NumPy와의 차이점





0번째 **행**을 반환

## DataFrame의 다양한 생성 방법

- 단일 Series로부터 구성하기





## DataFrame의 다양한 생성 방법

- Dictionary의 List로부터 생성하기 • 각 dictionary는 DataFrame에서 행이 됨
- Dictionary의 key는 각 열의 이름에 대응

## DataFrame의 다양한 생성 방법

- 일부 key가 누락된 경우의 처리 • NaN: Not a Number

- 누락된 자리에 채워지는 값으로 숫자가 아님을 의미
- 실수 자료형으로 정의 됨 ◉◉ 벡터화된 연산 지원• 산술 연산이 가능

## NaN: Not a Number

- NaN이 포함된 연산의 결과는 모두 NaN



## DataFrame의 데이터 추가

- 새로운 column에 대한 key를 지정하면서 값을 대입



새로운 column에 대한  
key를 지정하면서 값을  
대입



DataFrame: 2차원 배열 • DataFrame에는 2차원 배열

## 형태로 저장

- 배열에서 접했던 많은 유사한 작업을 처리 가능
- Ex) Transpose



## DataFrame: 2차원 배열 • 행에 접근하고 싶을 경우 •

values 배열에 단일 인덱스를 전달



- 인덱스를 전달할 경우 열에 저장된 자료 반환



## DataFrame: 2차원 배열

- 앞서 Series와 유사하게 loc과 iloc 활용 가능



## DataFrame: 2차원 배열

- 인덱싱을 사용하여 값을 설정 혹은 변경 가능



## 계층적 인덱싱

- 각 주별로 2000년도와 2010년도의 자료가 있을 경우



<https://pastebin.com/AytUqvaf>

## 계층적 인덱싱

- 위의 구현에서 2010년의 모든 값을 살펴보고자 할 경우?•



더 좋은

방법은 없을까요? ♦♦ 계층적 인덱싱

## 계층적 인덱싱

- 튜플들로부터 다중 인덱스 생성 가능



- 생성한 다중 인덱스를 활용하여 데이터들을



계층적으로 저장 가능

## (실습) 계층적 인덱싱

- 첫 두 열은 다중 인덱스



# 계층적 인덱싱 • Texas의 2000년도 데이터 확인



- 부분 인덱싱



계층적 인덱싱

- 계층적 인덱싱된 데이터에 접근하기



## 계층적 인덱싱

- 부분 슬라이싱

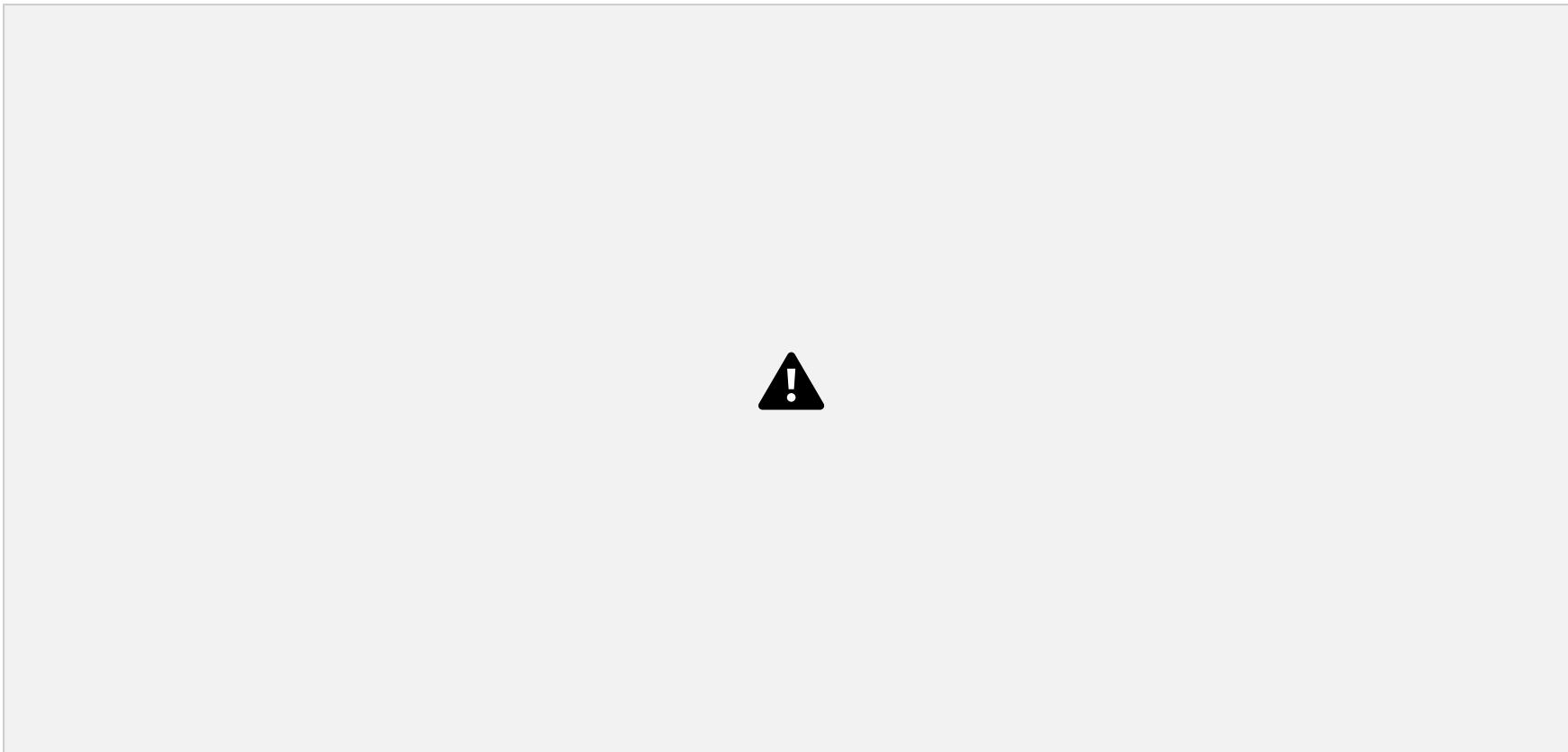


- 정렬되어 있지 않은 인덱스의 경우 .loc로 슬라이싱  
계층적 인덱싱
- Boolean 마스킹



## Boolean 마스킹의 활용

- 인구가 2,000,000 보다 큰 도시의 면적과 인구 밀도



Panel의 구조





Panel

