Chapter. 05

보고 싶은 데이터 찾아보기: 데이터 정렬과 인덱싱

데이터 정렬

FAST CAMPUS
ONLINE
데이터 탐색과 전처리 I

강사. 안길승

I 리스트 및 튜플 정렬

- sort: 리스트 및 튜플 자체를 정렬 (return 값이 없음)
  - ➢ reverse: 내림차순으로 정렬할 것인지 여부를 결정하는 매개변수
  - ➢ Pandas에서 inplace = True로 설정한 것과 같음

- sorted: 정렬된 리스트 및 튜플을 반환
  - ➢ reverse: 내림차순으로 정렬할 것인지 여부를 결정하는 매개변수
  - ➢ key: 정렬 기준 함수 (주로 lambda 함수를 사용)

I Series 정렬

- sort_values를 사용하면 Series를 손쉽게 정렬할 수 있음
- 주요 입력
  - ascending: 오름차순으로 정렬할 것인지 여부
  - key: 정렬 기준 함수 (주로 lambda 함수를 사용)
  - na_position: 결측이 있는 경우 어디에 배치할 것인지 결정 {first, last}

Series 요약 함수

- value_counts: Series의 구성 요소의 빈도를 순서대로 출력
  - ascending: 오름차순으로 정렬할 것인지 여부
  - normalize: 빈도 대신 비율을 출력할 것인지 여부

- unique: Series에 포함된 유니크한 값을 출력
  - 출력 결과의 데이터 타입: ndarray
  - 범주형 변수와 연속형 변수를 판단하는데 사용 가능

DataFrame 정렬

- sort_values를 사용하면 DataFrame도 손쉽게 정렬할 수 있음
- 주요 입력
  - by: 정렬 기준 컬럼 (목록)
  - ascending: 오름차순으로 정렬할 것인지 여부
  - key: 정렬 기준 함수 (주로 lambda 함수를 사용)
  - na_position: 결측이 있는 경우 어디에 배치할 것인지 결정 {first, last}

DataFrame 중복 제거

- drop_duplicates 함수를 사용하면 중복이 있는 행을 제거할 수 있음
- 주요 입력
  - ➢ subset: 중복 기준을 판단하는 컬럼 (목록)
  - ➢ keep: 중복이 있는 행의 어느 부분을 남길 것인지 결정 {‘first’, ‘last’, ‘false’}
    - first: 첫 번째 행을 남김
    - last: 마지막 행을 남김
    - false: 중복 행을 모두 제거

Chapter. 05

보고 싶은 데이터 찾아보기: 데이터 정렬과 인덱싱

인덱서를 활용한 인덱싱

FAST CAMPUS
ONLINE
데이터 탐색과 전처리 I

강사. 안길승

I 인덱싱과 슬라이싱 (remind)

- 판다스의 객체는 암묵적인 인덱스(위치 인덱스)와 명시적인 인덱스라는 두 종류의 인덱스가 있어, **명시**적인 인덱스를 참조하는 **loc 인덱서**와 암묵적인 인덱스를 참조하는 **iloc 인덱서**가 존재함

암묵적 인덱스	명시적 인덱스	Data
0	a	1
1	b	2
2	c	3
3	d	4

S

→

S.loc['a'] = 1
S.iloc[2] = 3

S.loc['a':'c'] = [1, 2, 3] loc를 이용한 슬라이싱에서는 맨 뒤 값을 포함
S.iloc[1:3] = [2, 3] iloc를 이용한 슬라이싱에서는 맨 뒤 값을 포함 X

- 데이터 프레임의 컬럼 선택: df[col_name] or df[col_name_list]

I 값 변경하기 (remind)

- 인덱서를 사용하여 조회한 값을 직접 변경할 수 있음

Index	Col1	Col2
a	1	5
b	2	6
c	3	7
d	4	8

df

df.iloc[2, 1] = 10

Index	Col1	Col2
a	1	5
b	2	6
c	3	10
d	4	8

인덱서를 사용해야 하는 이유

- DataFrame의 값을 바꾸는 경우에 가장 자주 흔히 보는 경고로 SettingWithCopyWarning이 있음
- df['A']는 view를 반환하고, df.loc['A']는 copy를 반환
  - view를 반환한 결과를 바꾸는 경우에는 원본 자체도 변경이 일어날 수 있음
  - copy를 반환한 결과를 바꾸더라도 원본에는 변경이 없음
- 따라서 view를 반환하지 않도록 가능하면 인덱서를 사용해야만, 사용자의 의도대로 DataFrame을 변경할 수 있음

Chapter. 05

보고 싶은 데이터 찾아보기: 데이터 정렬과 인덱싱

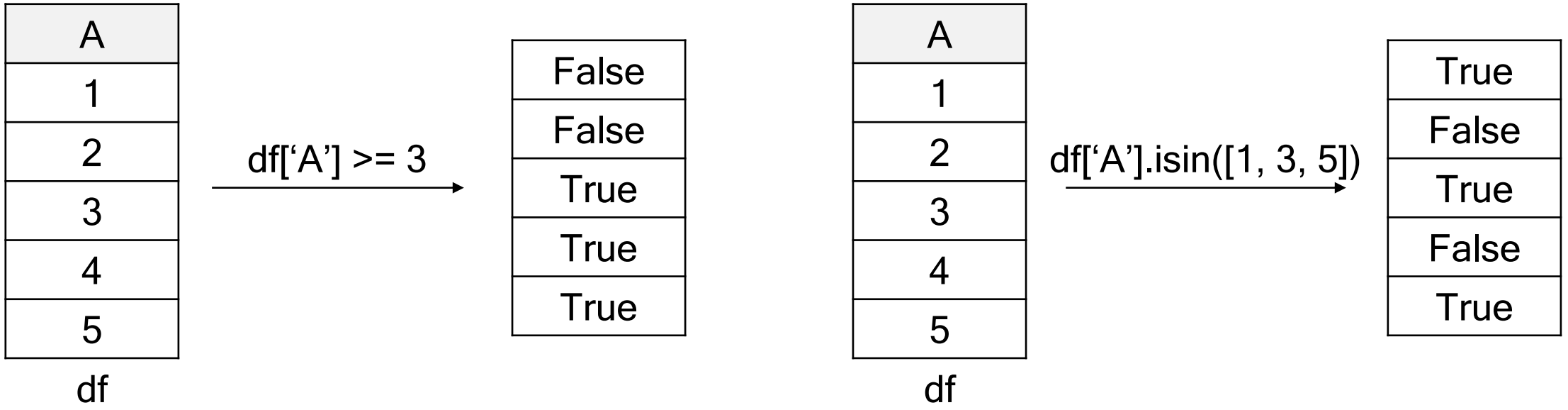
마스킹 검색

FAST CAMPUS
ONLINE
데이터 탐색과 전처리 I

강사. 안길승

| 비교 연산

- Series와 DataFrame의 data가 모두 ndarray이므로, **비교 연산** 및 **브로드캐스팅**이 적용될 수 있음
- 예시



| 부울 리스트 연산

- ndarray 타입의 부울 리스트에 대해, AND 연산(&)과 OR 연산(|)이 가능함
- 활용 예시

Ind	A	B
0	1	“X”
1	2	“X”
2	3	“Y”
3	4	“Y”
4	5	“X”

df

df[‘A’] >= 3

0	False
1	False
2	True
3	True
4	True

df[‘B’] == “Y”

0	False
1	False
2	True
3	True
4	False

(df[‘A’] >= 3) & (df[‘B’] == “Y”)

0	False
1	False
2	True
3	True
4	False

I 마스킹 검색

- df.loc[부울 리스트]: True인 요소의 위치에 대응되는 행만 가져옴

Ind	A	B
0	1	“X”
1	2	“X”
2	3	“Y”
3	4	“Y”
4	5	“X”

df

0	False
1	False
2	True
3	True
4	False

B

→

Ind	A	B
2	3	“Y”
3	4	“Y”

df.loc[B]

Chapter. 05

보고 싶은 데이터 찾아보기: 데이터 정렬과 인덱싱

문자열 검색

FAST CAMPUS
ONLINE
데이터 탐색과 전처리 I

강사. 안길승

Series.str

- 문자열로 구성된 Series에 대해서는 str accessor를 사용할 수 있으며, 이를 사용하면 string 관련 내장 함수를 자유 자재로 사용할 수 있음

- 예시
  - Series.str.strip(): 앞 뒤 공백 제거
  - Series.str.contains(s): 문자열 s를 포함하고 있는지 여부를 반환
  - Series.str.split(sep, expand): sep을 기준으로 Series를 분할
    - expand = True: 새로운 열 생성
    - expand = False: 새로운 열을 생성하지 않고 리스트를 반환

Series.astype

- Series의 자료형을 변환하는데 사용하는 함수
- (예시) Series.astype(str): 숫자 자료형에 str accessor를 사용하기 위해 적용함