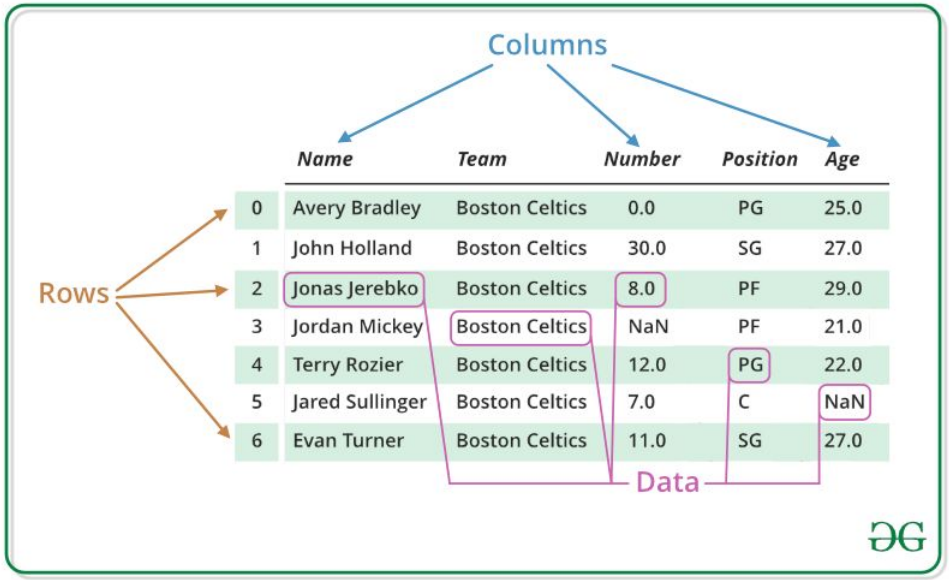
Lecture 5. Pandas 데이터 프레임의 활용(1)

기초 데이터 분석

Recap : Pandas 기본 사용법

1. Pandas 라이브러리?
2. Pandas vs Numpy
3. Pandas 사용 환경 설정



Recap : Pandas 기본 사용법

3. 시리즈와 데이터 프레임의 사용법 : pd.Series(), pd.DataFrame(), columns()

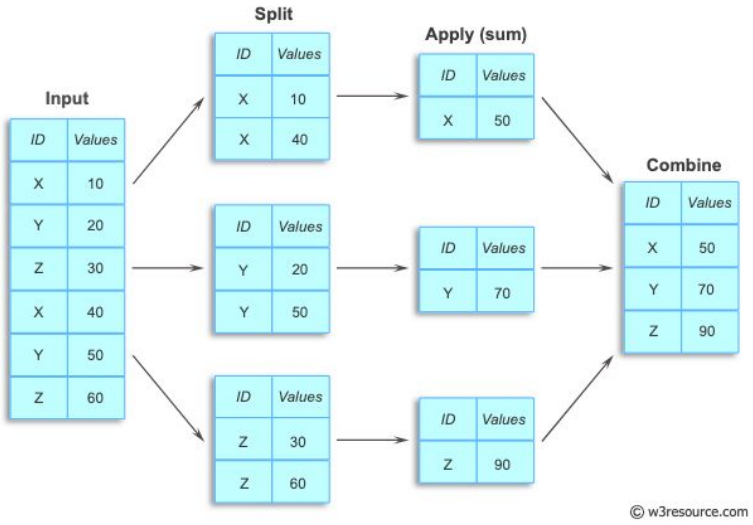
4. 데이터 불러오기 / 추출하기 : read_csv() , to_csv()

5. 데이터 통계 활용하기 : std(), mean(), describe()

6. 데이터 연결과 누락/중복 값 처리 방법 : concat(), join(), fillna(), dropna(), drop_duplicates()

Today : Pandas 데이터 활용법

1. **Sample** 데이터 불러오기
2. 데이터프레임 조작하기
3. 조건문 활용하여 데이터 조회하기
4. 데이터 프레임 내 문자열 처리
5. 그룹 연산 : **groupby()**



Sample Data 불러오기

- **Pandas 실습을 위해서 Seaborn 라이브러리에서 제공하는 데이터셋 사용**
  - pip install seaborn : seaborn 라이브러리 설치
  - seaborn에서 load_dataset을 활용하여 연습용 데이터 불러오기

In [2]:
```
import seaborn as sns # Seaborn 라이브러리
print(sns.get_dataset_names())
```

In [3]:
```
iris = sns.load_dataset('iris') # iris 라는 이름의 데이터셋 불러오기
iris.head()
```

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

DataFrame 조작 +@

- **특정 열 제거하기 : drop**
  - drop() 함수 내에 삭제하려는 column 이름이나 index를 작성
  - axis = 1 : 열을 제거 하는 경우 1, 행을 제거하는 경우 0으로 설정

In [2]:
```
print('Before : ', iris.columns.tolist())
iris = iris.drop(['petal_width'],axis=1) #petal_width 컬러 제거
print('After : ', iris.columns.tolist())
```

```
Before :  ['sepal_length', 'sepal_width', 'petal_length', 'petal_width', 'species']
After :  ['sepal_length', 'sepal_width', 'petal_length', 'species']
```

DataFrame 조작 +@

- **특정 열/행의 합 구하기**
  - sum() 함수를 활용해 특정 열/행 의 총 합을 계산한다.
  - 특정 열에 대한 정보만 얻거나 여러 열의 합을 구할 수도 있다.
  - Arguments
    - axis = 0 : 행(0) 또는 열(1) 중에서 계산을 할 축을 의미

In [2]:
```
iris.sum() #iris 데이터의 모든 열마다 합을 계산
```

```
sepal_length    876.5
sepal_width     458.6
petal_length    563.7
petal_width     179.9
```

DataFrame 조작 +@

- **중복치 처리 : duplicated()**
  - 데이터 내에 존재하는 중복된 행 or 열 확인
  - duplicated() : 중복된 열들을 확인한다
  - sum() 함수와 함께 사용해 중복된 갯수를 얻을 수 있다

In [2]:
```
iris['species'].duplicated().head() #붓꽃의 종류가 중복되는지 확인
```

```
0    False
1     True
2     True
3     True
4     True
Name: species, dtype: bool
```

DataFrame 조작 +@

- **중복치 처리 : drop_duplicates()**
  - 데이터 내에 존재하는 중복된 행 or 열 제거
  - drop_duplicates(column_name_list) : 중복된 행/열을 제거한다
  - Arguments
    - 기준으로 하는 열들의 이름 리스트
    - keep : 중복된 것들 중, 남길 항목 선정 [‘first’, ‘last’, False]

In [2]:
```
iris.drop_duplicates(['petal_length'],keep='first').head()
# petal_length가 같은 행을 모두 제거 + 가장 먼저 등장한 행 남기기
```

	sepal_length	sepal_width	petal_length	species
0	5.1	3.5	1.4	setosa
2	4.7	3.2	1.3	setosa
3	4.6	3.1	1.5	setosa
5	5.4	3.9	1.7	setosa
11	4.8	3.4	1.6	setosa

자료형 변환

- **특정 열의 자료형을 강제로 변경 : astype**
  - pandas 자료형 : object, int64, float64, bool, datetime, category…
  - 자료형의 효율성 차이가 있거나 자료형이 잘 못 지정된 경우 사용
  - df[변경대상컬럼].astype(변경할_데이터타입)

In [3]:
```
iris['sepal_width_str'] = iris['sepal_width'].astype(str) # sepal_width 를 str으로 저장
print(iris['sepal_width_str'].info())
```

```
Name: sepal_width_str, Length: 150, dtype: object
```

Indexing

- **기본적인 조건문 외에 직접 행/열 선택 : loc & iloc**
  - index를 사용하여 직접 행이나 열을 선택할 수 있다.
  - loc : 조건문이나 label을 통해 행/열을 선택
  - iloc : index를 기반으로 행/열을 선택

In [3]:
```
iris.loc[10,'species'] # 10이라는 index를 가지며, 'species'를 column label로 가지는
iris.iloc[10,3] # 10번 행 index와 column의 3번 index인 'species'를 불러옴
```

```
'setosa'
'setosa'
```

In [3]:
```
iris.iloc[:,3] # 1column의 3번 index인 'species'를 모두 불러옴
```

```
0    setosa
1    setosa
2    setosa
3    setosa
4    setosa
```

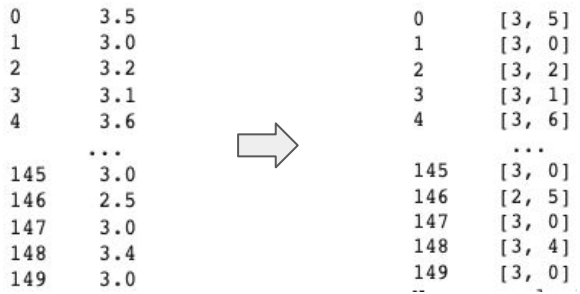
문자열 처리

- **공백 제거하기 : str.strip(), lstrip(), rstrip()**
  - 문자열 앞/ 뒤 에 공백이 들어가 있는 경우 제거
  - strip : 앞/뒤 모두 공백 제거
  - l & r + strip() : 앞 또는 뒤 의 공백만 제거
  - ex) : df[‘species’] = df[‘species’].str.strip()

문자열 처리

- **구분자로 문자열 나누기 : str.split(구분자)**
  - 구분자를 기준을 특정 컬럼을 여러 컬럼으로 나눈다
  - 예를 들어 “abcd@gmail.com” 을 ‘@’ 로 나눌 경우 ‘abcd’와 ‘gmail.com’ 두 개의 컬럼 생성
  - split 후 strip을 사용하여 에러 방지 가능

In [3]:
```
iris.sepal_width_str.str.split('.') # string화 한 sepal_width 를 .를 기준으로 나누기
```



0	3.5
1	3.0
2	3.2
3	3.1
4	3.6
...	...
145	3.0
146	2.5
147	3.0
148	3.4
149	3.0

0	[3, 5]
1	[3, 0]
2	[3, 2]
3	[3, 1]
4	[3, 6]
...	...
145	[3, 0]
146	[2, 5]
147	[3, 0]
148	[3, 4]
149	[3, 0]

문자열 처리

- **패턴 찾기**
  - str.startswith(pat) : 문자열이 pat 으로 시작하는지 여부를 반환 (Bool)
  - str.endswith(pat) : 문자열이 pat으로 끝나는지 여부를 반환 (Bool)
  - str.contains(pat) : 문자열이 pat을 포함하는지 여부를 반환 (Bool)
  - str.match(pat:regex) : 정규표현식인 pat을 만족하는지 여부를 반환

In [3]:
```
iris[iris.species.str.contains('ini')].head() #ini를 포함하는 virginica만 불러오게 하는 코드
```

	sepal_length	sepal_width	petal_length	species	sepal_width_str
100	6.3	3.3	6.0	virginica	3.3
101	5.8	2.7	5.1	virginica	2.7
102	7.1	3.0	5.9	virginica	3.0
103	6.3	2.9	5.6	virginica	2.9
104	6.5	3.0	5.8	virginica	3.0

GroupBy

- 그룹지어 연산하기 : **groupby() + agg()**
  - groupby 이후 여러 연산/통계를 한번에 구하고 싶은 경우
  - agg() 함수를 한번 더 활용한다.
  - agg()의 인자로 리스트가 오는 경우 : 모든 컬럼에 대해 해당 연산을 모두 진행
  - agg()의 인자가 dict 인 경우 : 해당하는 컬럼마다 dict의 연산만 진행

In [3]:
```
iris.groupby('species').agg(['mean','var']) # iris의 species에 따라 그룹을 짓고 mean & var를 구함
```

	sepal_length		sepal_width		petal_length	
	mean	var	mean	var	mean	var
species						
setosa	5.006	0.124249	3.428	0.143690	1.462	0.030159
versicolor	5.936	0.266433	2.770	0.098469	4.260	0.220816
virginica	6.588	0.404343	2.974	0.104004	5.552	0.304588

GroupBy

- 그룹지어 연산하기 : **groupby() + agg()**
  - groupby 이후 여러 연산/통계를 한번에 구하고 싶은 경우
  - agg() 함수를 한번 더 활용한다.
  - agg()의 인자로 리스트가 오는 경우 : 모든 컬럼에 대해 해당 연산을 모두 진행
  - agg()의 인자가 dict 인 경우 : 해당하는 컬럼마다 dict의 연산만 진행

In [3]:
```
iris.groupby('species').agg({
'sepal_length' : 'mean',
'petal_length' : ['var','sum']
}) # iris의 species에 따라 그룹을 짓고 mean & var를 구함
```

	sepal_length	petal_length	
	mean	var	sum
species			
setosa	5.006	0.030159	73.1
versicolor	5.936	0.220816	213.0
virginica	6.588	0.304588	277.6