

# Data Wrangling

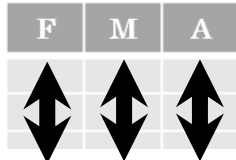
with pandas Cheat Sheet

<http://pandas.pydata.org>

Pandas [API Reference](#) Pandas [User Guide](#)


## 정돈된 데이터(Tidy Data) - A foundation for wrangling in pandas

In a tidy data set:

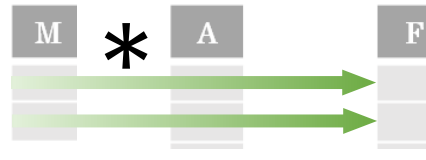


각 variable은 열에 저장됩니다.

Tidy data는 pandas's **vectorized operations**을 보완합니다. 데이터를 조작함에 따라 pandas는 자동적으로 이를 보존합니다. pandas는 가장 직관적으로 동작하는 데이터 처리 도구입니다.



각 observation은 행에 저장됩니다.



M \* A

## 데이터 생성(Creating DataFrame)

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

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

각 필드의 값을 키워드와 값으로 명시합니다.

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

각 필드의 값을 리스트로 명시합니다.

		a	b	c
N	v			
D	1	4	7	10
	2	5	8	11
e	2	6	9	12

```
df = pd.DataFrame(  
    {"a" : [4, 5, 6],  
     "b" : [7, 8, 9],  
     "c" : [10, 11, 12]},  
    index = pd.MultiIndex.from_tuples(  
        [('d', 1), ('d', 2),  
         ('e', 2)], names=['n', 'v']))
```

다중인덱스를 포함한 데이터프레임을 생성합니다.

## 메소드 체이닝(Method Chaining)

pandas 내장 함수 대부분은 데이터프레임을 반환합니다. 따라서, 함수의 결과가 또다른 함수에 적용될 수 있습니다. 이를 통해 코드의 가독성과 사용성을 개선하고 있습니다.

```
df = (pd.melt(df)  
      .rename(columns={  
          'variable': 'var',  
          'value': 'val'})  
      .query('val >= 200'))
```

## 데이터 재구조화(Data Reshaping)-레이아웃 변경(Change layout), 정렬(sorting), 재구성(reindexing), 이름 변경(naming)



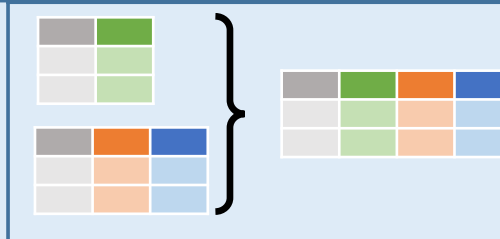
`pd.melt(df)`  
열을 행으로 재구조화합니다.



`df.pivot(columns='var', values='val')`  
행을 열로 재구조화합니다.



`pd.concat([df1, df2])`  
데이터프레임을 행(가로)에 이어붙입니다.



`pd.concat([df1, df2], axis=1)`  
데이터프레임을 열(세로)에 이어붙입니다.

`df.sort_values('mpg')`  
데이터를 오름차순으로 정렬합니다.

`df.sort_values('mpg', ascending=False)`  
데이터를 내림차순으로 정렬합니다.


`df.rename(columns = {'y': 'year'})`  
열 이름을 변경합니다.

`df.sort_index()`  
인덱스를 정렬합니다.

`df.reset_index()`  
현재 인덱스를 열로 구성 합니다.  
새 인덱스는 행 번호로 재설정합니다.

`df.drop(columns=['Length', 'Height'])`  
열을 삭제합니다.

## Subset Observations-rows



```
df[df.Length > 7]  
조건을 만족하는 행을 추출합니다.  
df.drop_duplicates()  
중복되는 행을 제거합니다.  
df.sample(frac=0.5)  
일정 비율로 행을 임의 선택합니다.  
df.sample(n=10)  
지정한 개수만큼 행을 임의 선택합니다.  
df.nlargest(n, 'value')  
상위 n개의 엔트리만 선택해 정렬합니다.  
df.nsmallest(n, 'value')  
하위 n개의 엔트리만 선택해 정렬합니다.  
df.head(n)  
처음부터 n개 행을 선택합니다.  
df.tail(n)  
마지막부터 n개 행을 선택합니다.
```

## Subset Variables-columns



```
df[['width', 'length', 'species']]  
특정 열을 선택합니다.  
df['width'] or df.width  
하나의 열을 선택합니다.  
df.filter(regex='regex')  
정규표현식에 매치되는 열을 선택합니다.
```

## 쿼리 사용(Using query)

query()는 행을 조건에따라 선택하기 위한 부울 표현을 허용합니다.

```
df.loc[df['a'] > 10, ['a', 'c']]  
조건을 만족하는 데이터를 선택합니다.  
df.iat[1, 2] 인덱스를 통해 값에 접근합니다.  
df.at[4, 'A'] 라벨을 통해 값에 접근합니다.
```

## Subsets-rows and columns

행과 열을 선택하기 위해 `df.loc[]`과 `df.iloc[]` 사용하십시오.  
데이터 하나에 접근하려면 `df.at[]`와 `df.iat[]`를 사용하십시오.  
네 가지 속성의 첫 번째 인덱스는 행을 선택합니다.  
네 가지 속성의 두 번째 인덱스는 열을 선택합니다.  
`df.iloc[10:20]`  
10번째부터 20번째까지 행을 선택합니다.  
`df.iloc[:, [1, 2, 5]]`  
모든 행에서 열 1,2,5의 값을 선택합니다.(열 시작 번호 : 0번)  
`df.loc[:, 'x2' : 'x4']`  
모든 행에서 'x2'와 'x4' 사이의 값을 선택합니다.

Logic in Python (and pandas)			
<	미만	!=	동등하지 않은
>	초과	<code>df.column.isin(values)</code>	포함돼 있는가?
==	동등	<code>pd.isnull(obj)</code>	Not a Number인가?
<=	이하	<code>pd.notnull(obj)</code>	Not a Number가 아닌가?
>=	이상	<code>&amp;,  , ~, ^, df.any(), df.all()</code>	Logical and, or, not, xor, any, all

regex (Regular Expressions) Examples	
'\.'	마침표 '.'을 포함하는 문장과 매치
'Length\$'	단어 'Length' 로 끝나는 문자열과 매치
'^Sepal'	단어 'Sepal' 로 시작하는 문자열과 매치
'^x[1-5]\$'	'x'로 시작하고 1,2,3,4,5 중 하나로 끝나는 문자열과 매치
'^(?!Species\$).*	문자열을 제외한 모든 문자열과 매치

## 데이터 요약(Summarize Data)

`df['w'].value_counts()` 동일한 데이터가 몇 개나 있는지 측정합니다.

`len(df)` 데이터프레임 내 행의 수를 반환합니다.

`df.shape` 행과 열의 수를 튜플로 반환합니다.

`df['w'].nunique()` 중복되지 않는 데이터의 수를 반환합니다.

`df.describe()` 기본적인 통계값을 반환합니다.



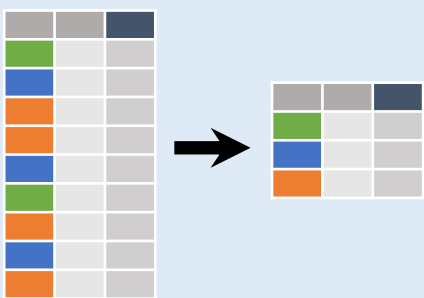
pandas는 여러 종류의 pandas object(DataFrame columns, Series, GroupBy, Expanding and Rolling)에서 동작하는 함수들을 제공합니다.

그리고 그룹의 각각에 대한 반환값을 생성합니다.

DataFrame에 적용이 된다고 할 때, 결과는 각 열에 대응해서 pandas Series로 반환됩니다. 예시:

<code>sum()</code>	<code>min()</code>
합을 반환합니다.	최소값을 반환합니다.
<code>count()</code>	<code>max()</code>
빈도를 반환합니다.	최대값을 반환합니다.
<code>median()</code>	<code>mean()</code>
중앙값을 반환합니다.	평균값을 반환합니다.
<code>quantile([0.25,0.75])</code>	<code>var()</code>
백분위수를 반환합니다.	분산을 반환합니다.
<code>apply(function)</code>	<code>std()</code>
여러 가지 함수를 적용합니다.	표준편차를 반환합니다.

## 데이터 그룹(Group Data)



`df.groupby(by="col")`

이름이 "col"인 열의 값으로 그룹화된 GroupBy 객체를 반환합니다.

`df.groupby(level="ind")`

이름이 "ind"인 인덱스 레벨의 값으로 그룹화된 GroupBy 객체를 반환합니다.

위의 모든 함수들은 group에도 적용될 수 있습니다.

추가적인 GroupBy functions:

<code>size()</code>	<code>agg(function)</code>
각 그룹의 크기.	여러 열에 함수를 적용합니다.

## Windows

`df.expanding()` summary function이 누적 적용될 수 있게 하는 Expanding object를 반환합니다.

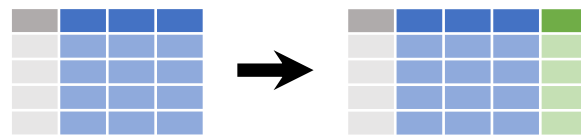
`df.rolling(n)` summary function이 길이가 n인 window에 적용될 수 있게 하는 Rolling object를 반환합니다.

## 누락데이터 관리(Handling Missing Data)

`df.dropna()` 결측값이 있는 행 또는 열을 삭제합니다.

`df.fillna(value)` 결측값이 있는 행 또는 열을 value로 교체합니다.

## 새로운 열 생성(Make New Columns)



`df.assign(Area=lambda df: df.Length*df.Height)` 함수를 적용해 열을 추가합니다.

`df['Volume'] = df.Length*df.Height*df.Depth` 한 개의 열을 추가합니다.

`pd.qcut(df.col, n, labels=False)` 동일한 개수로 범주를 나눕니다.



pandas는 DataFrame의 모든 열이나 Series에 대해

동작하는 많은 수의 vector functions를 제공합니다.

이 함수들은 열이나 시리즈에 대한 값의 벡터를 생성합니다. 예시:

<code>max(axis=1)</code>	<code>min(axis=1)</code>
행 단위로 최대값을 반환합니다.	행 단위로 최소값을 반환합니다.
<code>clip(lower=-10, upper=10)</code>	<code>abs()</code>
임계치 미만 혹은 초과 데이터를 수정합니다.	절대값을 반환합니다.

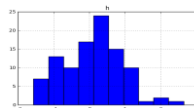
아래의 예시들도 group에 적용될 수 있습니다.

함수는 그룹 단위로 적용되며 반환된 벡터는 원래 데이터 프레임의 길이입니다.

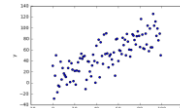
<code>shift(1)</code>	<code>shift(-1)</code>
행을 한칸 내립니다.	행을 한칸 올립니다.
<code>rank(method='dense')</code>	<code>cumsum()</code>
동점 시 낮은 순위를 부여하지만 1씩 증가.	누적 합을 반환합니다.
<code>rank(method='min')</code>	<code>cummax()</code>
동점 시 낮은 순위를 부여합니다.	누적 최대값을 반환합니다.
<code>rank(pct=True)</code>	<code>cummin()</code>
순위를 백분위로 표현합니다.	누적 최소값을 반환합니다.
<code>rank(method='first')</code>	<code>cumprod()</code>
동점 시 빠른 순서대로 순위를 부여합니다.	누적 곱을 반환합니다.

## Plotting

`df.plot.hist()` 각 열의 히스토그램



`df.plot.scatter(x='w', y='h')` 산점도 차트



## 데이터셋 결합(Combine Data Sets)

adf		bdf	
x1	x2	x1	x3
A	1	A	T
B	2	B	F
C	3	D	T



### Standard Joins

x1	x2	x3
A	1	T
B	2	F
C	3	NaN

`pd.merge(adf, bdf, how='left', on='x1')`  
bdf와 adf의 매치되는 행을 조인합니다.

x1	x2	x3
A	1.0	T
B	2.0	F
D	NaN	T

`pd.merge(adf, bdf, how='right', on='x1')`  
bdf와 adf의 매치되는 행을 조인합니다.

x1	x2	x3
A	1	T
B	2	F

`pd.merge(adf, bdf, how='inner', on='x1')`  
공통된 데이터가 있는 행만 보존해 조인합니다.

x1	x2	x3
A	1	T
B	2	F
C	3	NaN
D	NaN	T

`pd.merge(adf, bdf, how='outer', on='x1')`  
모든 값과 행을 보존해 조인합니다.

### Filtering Joins

x1	x2
A	1
B	2

`adf[adf.x1.isin(bdf.x1)]`  
bdf와 매치되는 모든 adf의 행만 조인합니다.

x1	x2
C	3

`adf[~adf.x1.isin(bdf.x1)]`  
bdf와 매치되지 않는 모든 adf의 행만 조인합니다.

ydf		zdf	
x1	x2	x1	x2
A	1	B	2
B	2	C	3
C	3	D	4



### Set-like Operations

x1	x2
B	2
C	3

`pd.merge(ydf, zdf)`  
교집합의 방식으로 병합합니다.

x1	x2
A	1
B	2
C	3
D	4

`pd.merge(ydf, zdf, how='outer')`  
합집합의 방식으로 병합합니다.

`pd.merge(ydf, zdf, how='outer', indicator=True)`  
`.query('_merge == "left_only"')`  
`.drop(columns=['_merge'])`

지정한 쿼리와 함수에 따라서 병합합니다.