# Data Wrangling

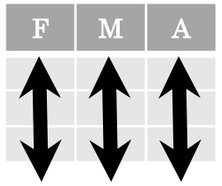
with pandas

Cheat Sheet

http://pandas.pydata.org

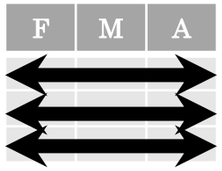
## Tidy Data – A foundation for wrangling in pandas

In a tidy data set:



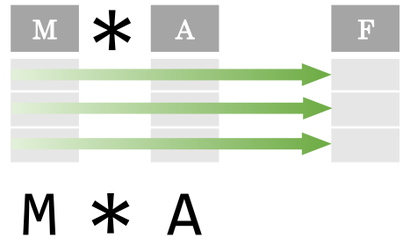

Each **variable** is saved in its own **column**

&




Each **observation** is saved in its own **row**

Tidy data complements pandas's **vectorized operations**. pandas will automatically preserve observations as you manipulate variables. No other format works as intuitively with pandas.




## Syntax – Creating DataFrames

| | 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])
```

Specify values for each column.

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

Specify values for each row.

| | | 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'])))
```

Create DataFrame with a MultiIndex

## Method Chaining

Most pandas methods return a DataFrame so that another pandas method can be applied to the result. This improves readability of code.

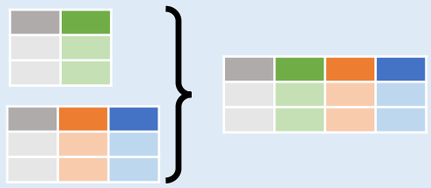
```
df = (pd.melt(df)
        .rename(columns={
                'variable' : 'var',
                'value' : 'val'})
        .query('val >= 200')
    )
```

## Reshaping Data – Change the layout of a data set

**pd.melt(df)**
Gather columns into rows.

**df.pivot(columns='var', values='val')**
Spread rows into columns.

**pd.concat([df1,df2])**
Append rows of DataFrames



**pd.concat([df1,df2], axis=1)**
Append columns of DataFrames

**df=df.assign(a=[1:3], b=[4:6])**
Add new columns to DataFrame.

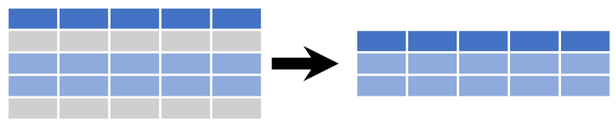
**df=df.sort_values('mpg')**
Order rows by values of a column (low to high).

**df=df.sort_values('mpg', ascending=False)**
Order rows by values of a column (high to low).

**df=df.rename(columns= {'y':'year'})**
Rename the columns of a DataFrame

**df=df.sort_index()**
Sort the index of a DataFrame

## Subset Observations (Rows)



**df[df.Length > 7]**
Extract rows that meet logical criteria.

**df.drop_duplicates()**
Remove duplicate rows (only considers columns).

**df.sample(frac=0.5)**
Randomly select fraction of rows.

**df.sample(n=10)**
Randomly select n rows.

**df.iloc[10:20]**
Select rows by position.

**df.nlargest(10, 'value')**
Select and order top n entries.

**Logic in Python (and pandas)**

| | | | |
|---|---|---|---|
| < | Less than | != | Not equal to |
| > | Greater than | df.column.isin(*values*) | Group membership |
| == | Equals | pd.isnull(*obj*) | Is NaN |
| <= | Less than or equals | Pd.notnull(*obj*) | Is not NaN |
| >= | Greater than or equals | &,\|,~,^,df.any(),df.all() | Logical and, or, not, xor, any, all |

## Subset Variables (Columns)

**df[['width','length','species']]**
Select multiple columns with specific names.

**df['width']** *or* **df.width**
Select single column with specific name.

**df[[i for i in df.columns if '.' in i]]**
Select columns whose name contains a character string.

**df[[i for i in df.columns if i.endswith("Length")]]**
Select columns whose name ends with a character string.

**df[[i for i in df.columns if re.match('.t.',i)]]**
Select columns whose name matches a regular expression.

**df[["x"+str(i) for i in range(1,6)]]**
Select columns named x1, x2, x3, x4, x5.

**df[[i for i in df.columns if i.startswith("Length")]]**
Select columns whose name starts with a character string.

**df.loc[:,"x2":"x4"]**
Select all columns between x2 and x4 (inclusive).

**df[[i for i in df.columns if i != "Species"]]**
Select all columns except Species.

**df.iloc[:,[1,2,5]]**
Select columns in positions 1, 2 and 5 (first column is 0).