Lecture 06: Pandas

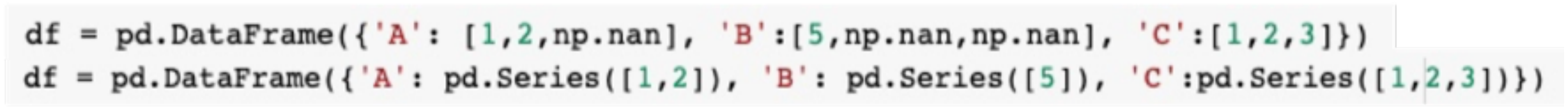
Hyeryung Jang
(hyeryung.jang@dgu.ac.kr)
AI Department, Dongguk University

Syllabus: Today's Topic

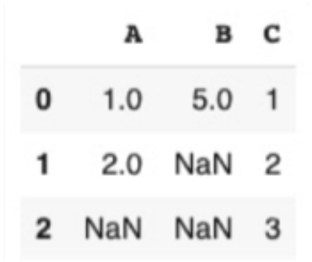
Week	Topics
1	Introduction to Data Science, Environment Set-up
2	Python Basics 1
3	Python Basics 2
4	Python for Data Analysis: NumPy
5	Python for Data Analysis: Pandas 1
6	**Python for Data Analysis: Pandas 2**
7	Python for Data Analysis: Web Crawling
8	Midterm Exam
9	Python for Data Visualization: Basics
10	Python for Data Visualization: Advanced
11	Machine Learning with Python: Supervised Learning
12	Machine Learning with Python: Unsupervised Learning
13	Machine Learning with Python: Recommender System
14	Project Presentation
15	Final Exam

Missing Data

- Some methods to deal with missing data in Pandas:

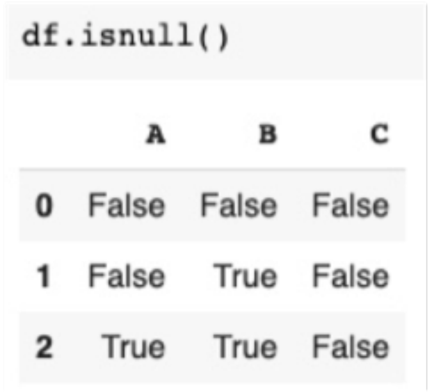


```python
df = pd.DataFrame({'A': [1,2,np.nan], 'B':[5,np.nan,np.nan], 'C':[1,2,3]})
df = pd.DataFrame({'A': pd.Series([1,2]), 'B': pd.Series([5]), 'C':pd.Series([1,2,3])})
```



	A	B	C
0	1.0	5.0	1
1	2.0	NaN	2
2	NaN	NaN	3

- **.isnull()**: find null values

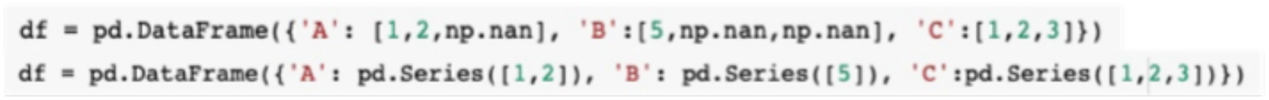


```python
df.isnull()
```

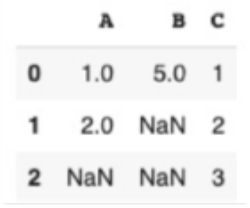
	A	B	C
0	False	False	False
1	False	True	False
2	True	True	False

Missing Data

- Some methods to deal with missing data in Pandas:

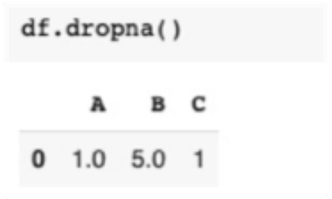


```
df = pd.DataFrame({'A': [1,2,np.nan], 'B':[5,np.nan,np.nan], 'C':[1,2,3]})
df = pd.DataFrame({'A': pd.Series([1,2]), 'B': pd.Series([5]), 'C':pd.Series([1,2,3])})
```



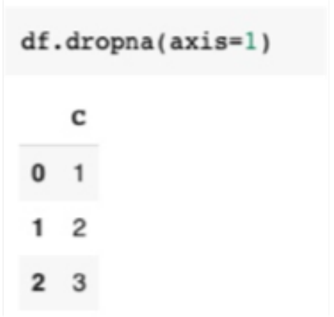
	A	B	C
0	1.0	5.0	1
1	2.0	NaN	2
2	NaN	NaN	3

- **.dropna()**: remove missing values
  - axis: drop row or columns
    - 0 or ‘index’: drop rows which contain missing values
    - 1 or ‘column’: drop columns which contain missing values
  - how:
    - ‘any’: if any NA values are present, drop that row or column
    - ‘all’: if all values are NA, drop that row or column



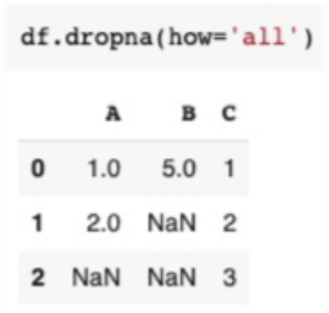
```
df.dropna()
```

	A	B	C
0	1.0	5.0	1



```
df.dropna(axis=1)
```

	C
0	1
1	2
2	3

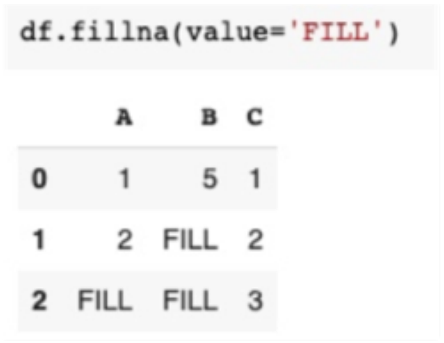


```
df.dropna(how='all')
```

	A	B	C
0	1.0	5.0	1
1	2.0	NaN	2
2	NaN	NaN	3

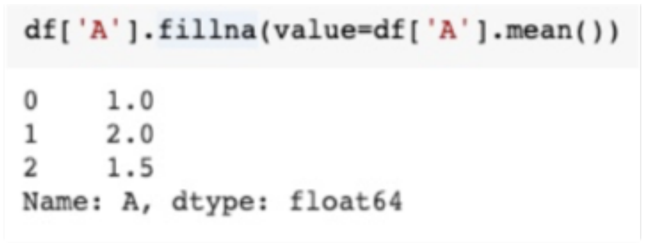
Missing Data

- **.fillna()**: fill NA/NaN values
  - value: value to use to fill holes
  - method: {‘backfill’, ‘bfill’, ‘pad’, ‘ffill’, None}
  - inplace: bool



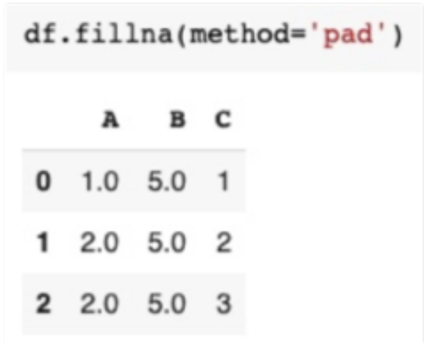
```
df.fillna(value='FILL')
```

	A	B	C
0	1	5	1
1	2	FILL	2
2	FILL	FILL	3



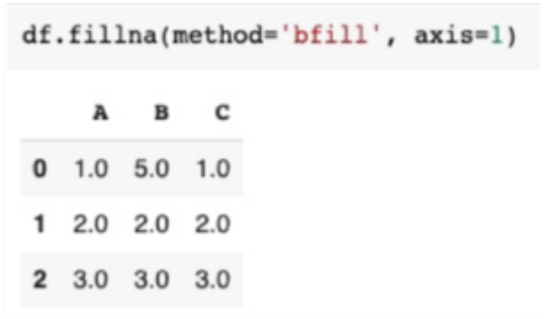
```
df['A'].fillna(value=df['A'].mean())
```

```
0    1.0
1    2.0
2    1.5
Name: A, dtype: float64
```



```
df.fillna(method='pad')
```

	A	B	C
0	1.0	5.0	1
1	2.0	5.0	2
2	2.0	5.0	3

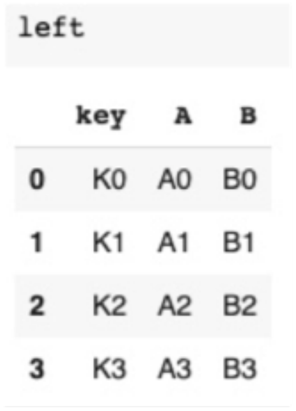


```
df.fillna(method='bfill', axis=1)
```

	A	B	C
0	1.0	5.0	1.0
1	2.0	2.0	2.0
2	3.0	3.0	3.0

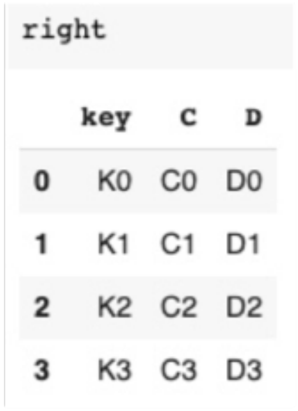
Merging and Concatenating

- **.merge()**: merge DataFrames together
  - .merge(left, right, how, on)
  - Look for one or more matching column names between two DataFrames



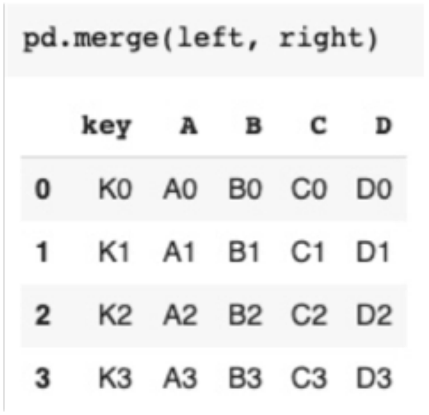
```
left
```

	key	A	B
0	K0	A0	B0
1	K1	A1	B1
2	K2	A2	B2
3	K3	A3	B3



```
right
```

	key	C	D
0	K0	C0	D0
1	K1	C1	D1
2	K2	C2	D2
3	K3	C3	D3

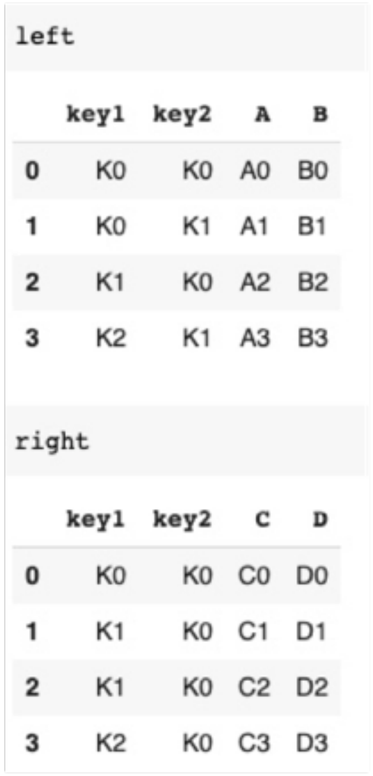


```
pd.merge(left, right)
```

	key	A	B	C	D
0	K0	A0	B0	C0	D0
1	K1	A1	B1	C1	D1
2	K2	A2	B2	C2	D2
3	K3	A3	B3	C3	D3

Merging and Concatenating

- **.merge()**: merge DataFrames together
  - How: ‘outer’ (use union: 합집합) / ‘inner’ (use intersection: 교집합)
  - On: column or index level names to join on

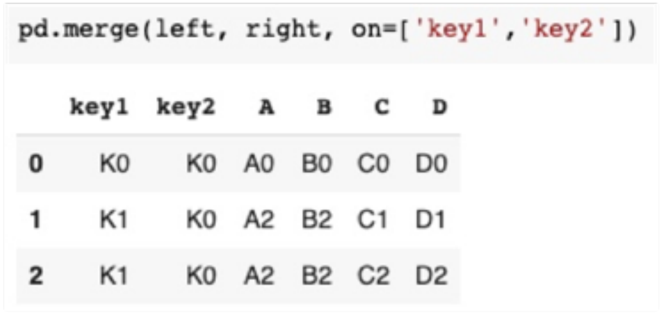


```
left
```

	key1	key2	A	B
0	K0	K0	A0	B0
1	K0	K1	A1	B1
2	K1	K0	A2	B2
3	K2	K1	A3	B3

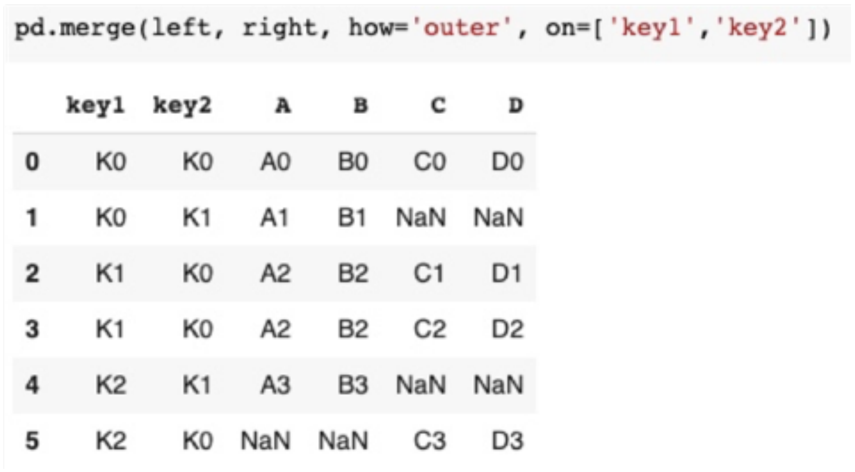
```
right
```

	key1	key2	C	D
0	K0	K0	C0	D0
1	K1	K0	C1	D1
2	K1	K0	C2	D2
3	K2	K0	C3	D3



```
pd.merge(left, right, on=['key1','key2'])
```

	key1	key2	A	B	C	D
0	K0	K0	A0	B0	C0	D0
1	K1	K0	A2	B2	C1	D1
2	K1	K0	A2	B2	C2	D2

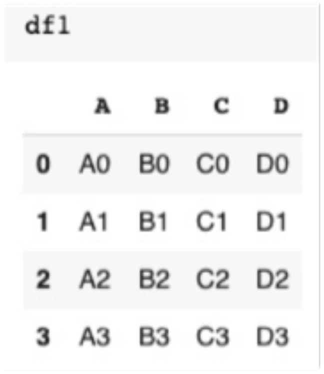


```
pd.merge(left, right, how='outer', on=['key1','key2'])
```

	key1	key2	A	B	C	D
0	K0	K0	A0	B0	C0	D0
1	K0	K1	A1	B1	NaN	NaN
2	K1	K0	A2	B2	C1	D1
3	K1	K0	A2	B2	C2	D2
4	K2	K1	A3	B3	NaN	NaN
5	K2	K0	NaN	NaN	C3	D3

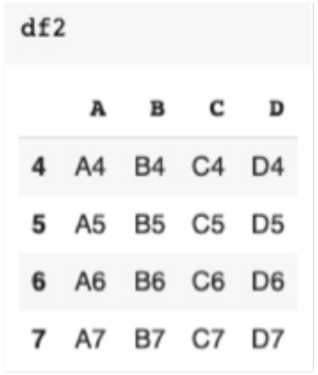
Merging and Concatenating

- **.concat()**: glues together DataFrames



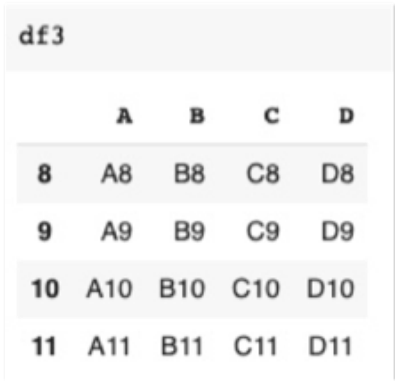
```
df1
```

	A	B	C	D
0	A0	B0	C0	D0
1	A1	B1	C1	D1
2	A2	B2	C2	D2
3	A3	B3	C3	D3



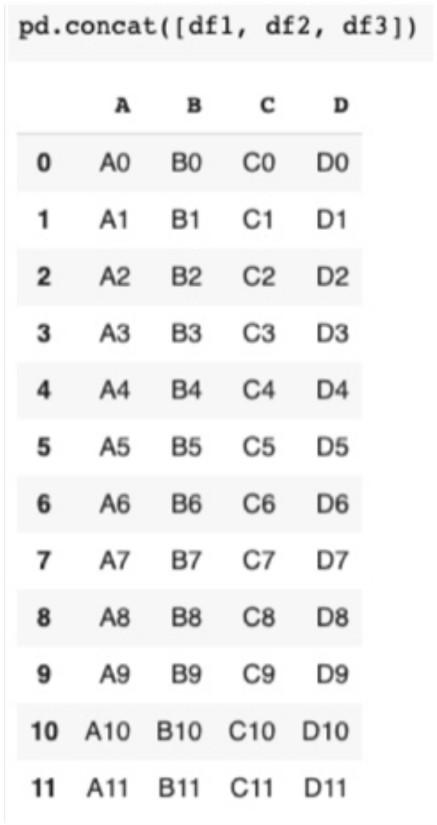
```
df2
```

	A	B	C	D
4	A4	B4	C4	D4
5	A5	B5	C5	D5
6	A6	B6	C6	D6
7	A7	B7	C7	D7



```
df3
```

	A	B	C	D
8	A8	B8	C8	D8
9	A9	B9	C9	D9
10	A10	B10	C10	D10
11	A11	B11	C11	D11

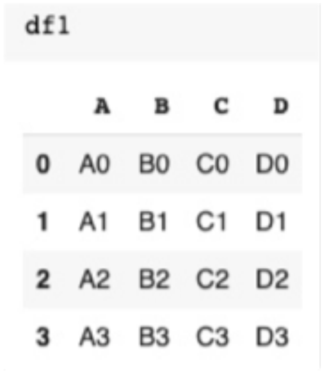


```
pd.concat([df1, df2, df3])
```

	A	B	C	D
0	A0	B0	C0	D0
1	A1	B1	C1	D1
2	A2	B2	C2	D2
3	A3	B3	C3	D3
4	A4	B4	C4	D4
5	A5	B5	C5	D5
6	A6	B6	C6	D6
7	A7	B7	C7	D7
8	A8	B8	C8	D8
9	A9	B9	C9	D9
10	A10	B10	C10	D10
11	A11	B11	C11	D11

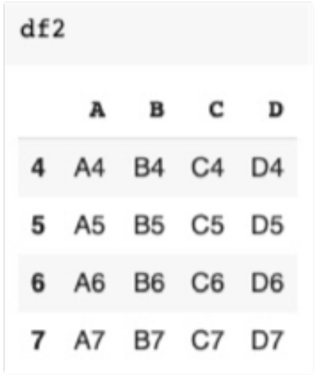
Merging and Concatenating

- **.concat()**: glues together DataFrames



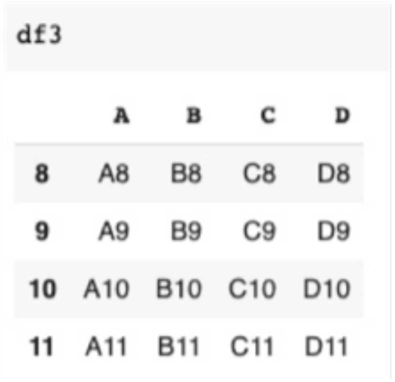
```
df1
```

	A	B	C	D
0	A0	B0	C0	D0
1	A1	B1	C1	D1
2	A2	B2	C2	D2
3	A3	B3	C3	D3



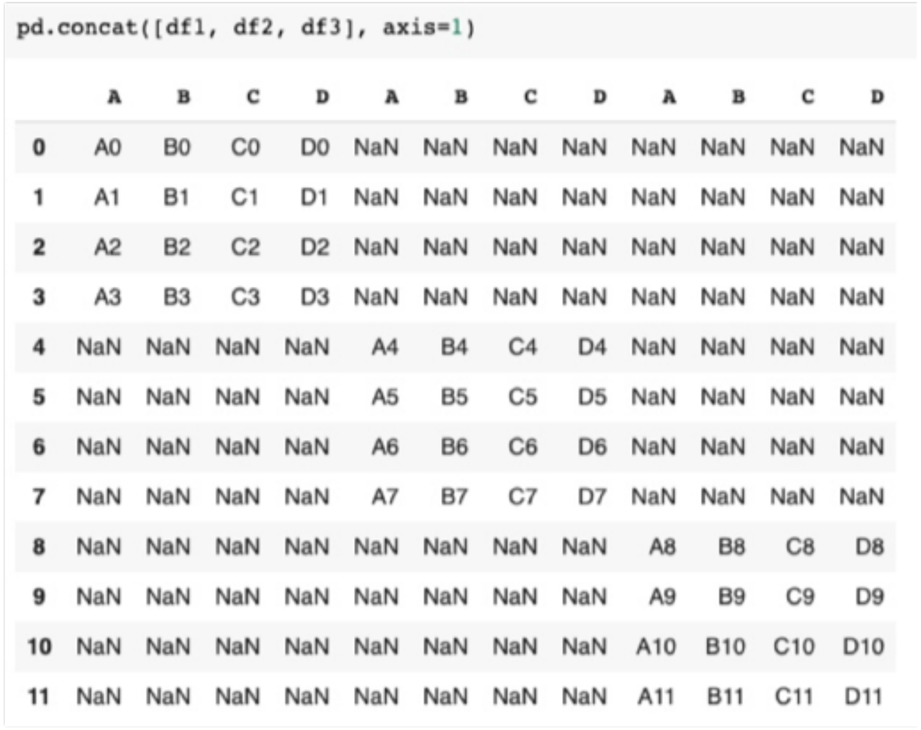
```
df2
```

	A	B	C	D
4	A4	B4	C4	D4
5	A5	B5	C5	D5
6	A6	B6	C6	D6
7	A7	B7	C7	D7



```
df3
```

	A	B	C	D
8	A8	B8	C8	D8
9	A9	B9	C9	D9
10	A10	B10	C10	D10
11	A11	B11	C11	D11

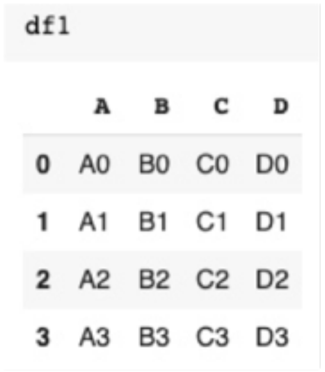


```
pd.concat([df1, df2, df3], axis=1)
```

	A	B	C	D	A	B	C	D	A	B	C	D
0	A0	B0	C0	D0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	A1	B1	C1	D1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	A2	B2	C2	D2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	A3	B3	C3	D3	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	NaN	NaN	NaN	NaN	A4	B4	C4	D4	NaN	NaN	NaN	NaN
5	NaN	NaN	NaN	NaN	A5	B5	C5	D5	NaN	NaN	NaN	NaN
6	NaN	NaN	NaN	NaN	A6	B6	C6	D6	NaN	NaN	NaN	NaN
7	NaN	NaN	NaN	NaN	A7	B7	C7	D7	NaN	NaN	NaN	NaN
8	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	A8	B8	C8	D8
9	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	A9	B9	C9	D9
10	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	A10	B10	C10	D10
11	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	A11	B11	C11	D11

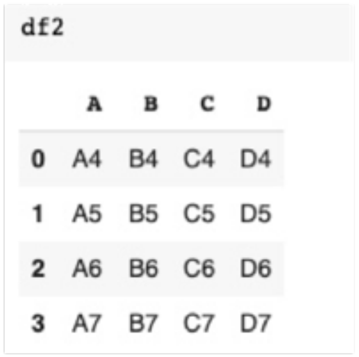
Merging and Concatenating

- **.concat()**: glues together DataFrames



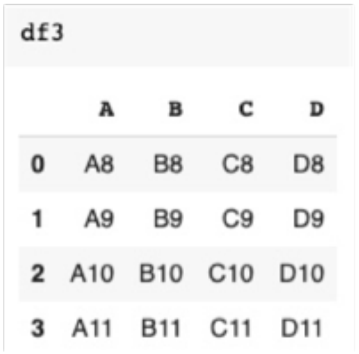
```
df1
```

	A	B	C	D
0	A0	B0	C0	D0
1	A1	B1	C1	D1
2	A2	B2	C2	D2
3	A3	B3	C3	D3



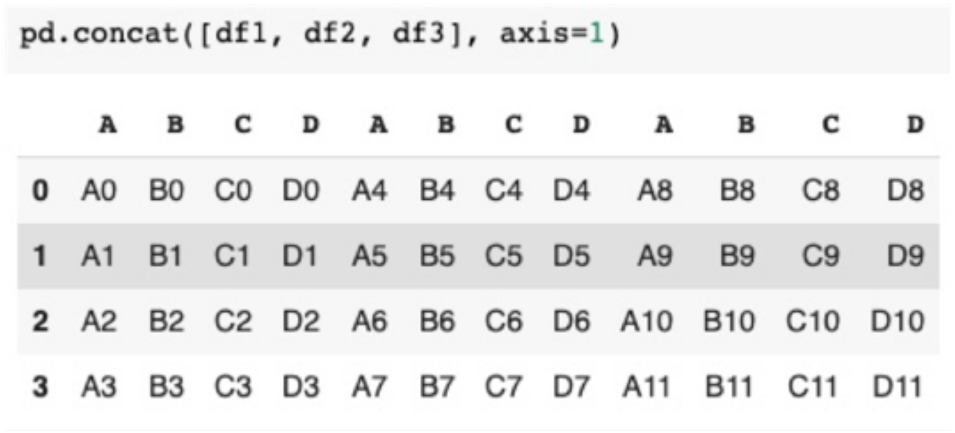
```
df2
```

	A	B	C	D
0	A4	B4	C4	D4
1	A5	B5	C5	D5
2	A6	B6	C6	D6
3	A7	B7	C7	D7



```
df3
```

	A	B	C	D
0	A8	B8	C8	D8
1	A9	B9	C9	D9
2	A10	B10	C10	D10
3	A11	B11	C11	D11

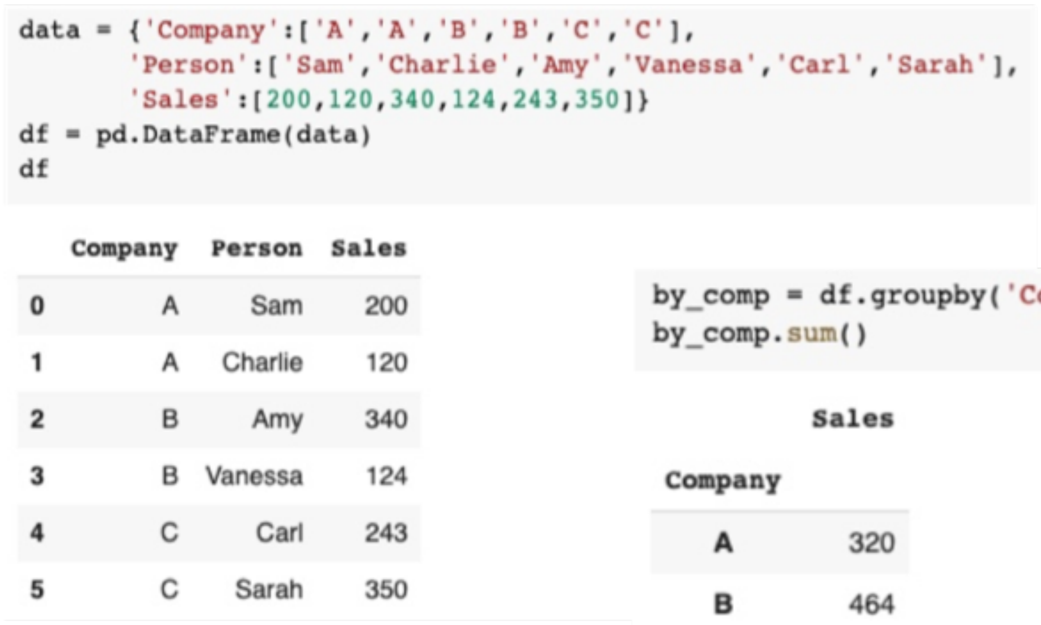


```
pd.concat([df1, df2, df3], axis=1)
```

	A	B	C	D	A	B	C	D	A	B	C	D
0	A0	B0	C0	D0	A4	B4	C4	D4	A8	B8	C8	D8
1	A1	B1	C1	D1	A5	B5	C5	D5	A9	B9	C9	D9
2	A2	B2	C2	D2	A6	B6	C6	D6	A10	B10	C10	D10
3	A3	B3	C3	D3	A7	B7	C7	D7	A11	B11	C11	D11

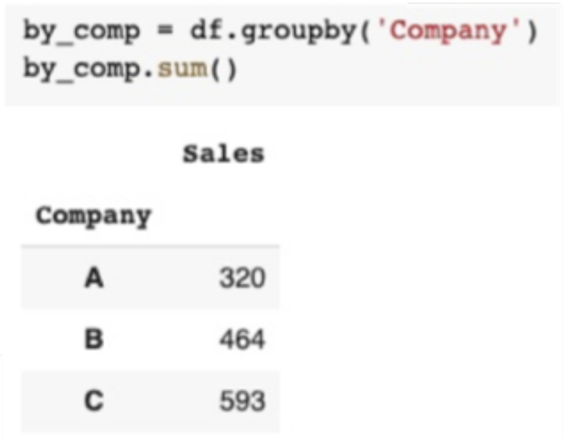
GroupBy

- **.groupby()**: group rows of data together and call aggregate functions
  - .groupby(column name)
  - Aggregate functions: .sum(), .median(), .std(), .min(),



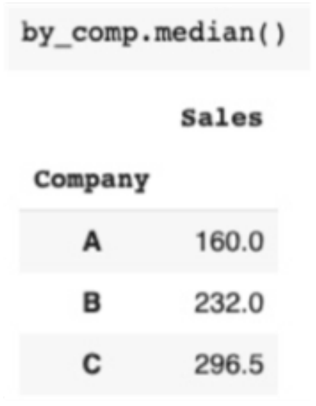
```python
data = {'Company':['A','A','B','B','C','C'],
        'Person':['Sam','Charlie','Amy','Vanessa','Carl','Sarah'],
        'Sales':[200,120,340,124,243,350]}
df = pd.DataFrame(data)
df
```

	Company	Person	Sales
0	A	Sam	200
1	A	Charlie	120
2	B	Amy	340
3	B	Vanessa	124
4	C	Carl	243
5	C	Sarah	350



```python
by_comp = df.groupby('Company')
by_comp.sum()
```

	Sales
Company	
A	320
B	464
C	593

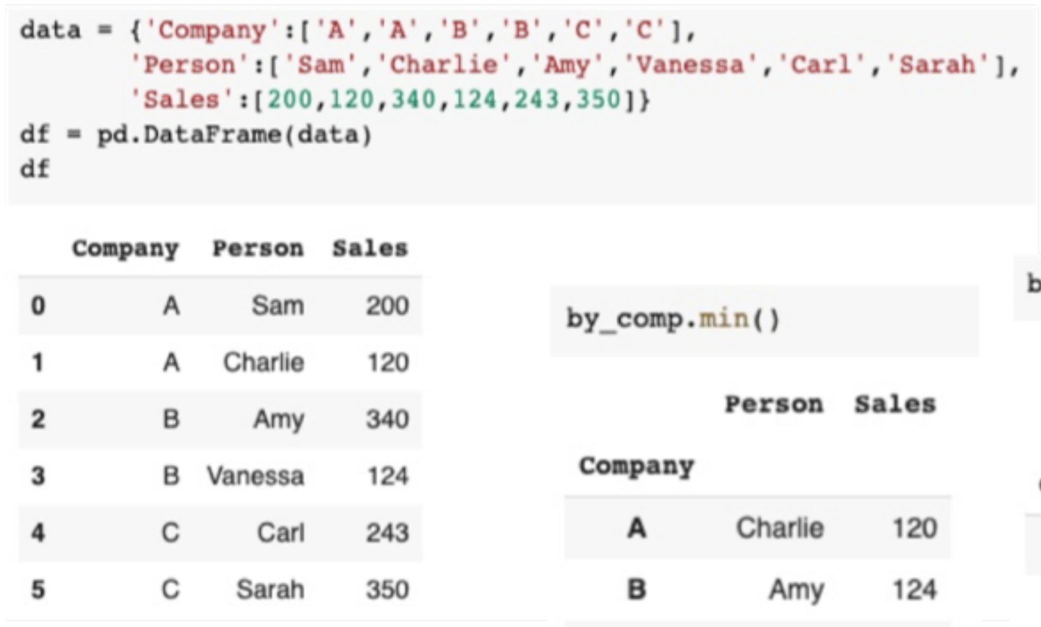


```python
by_comp.median()
```

	Sales
Company	
A	160.0
B	232.0
C	296.5

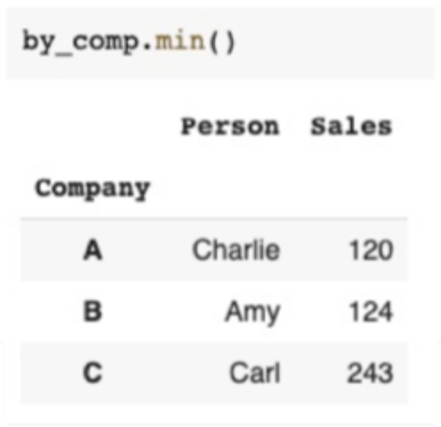
GroupBy

- **.groupby()**: group rows of data together and call aggregate functions
  - .groupby(column name)
  - Aggregate functions: .sum(), .median(), .std(), .min(),



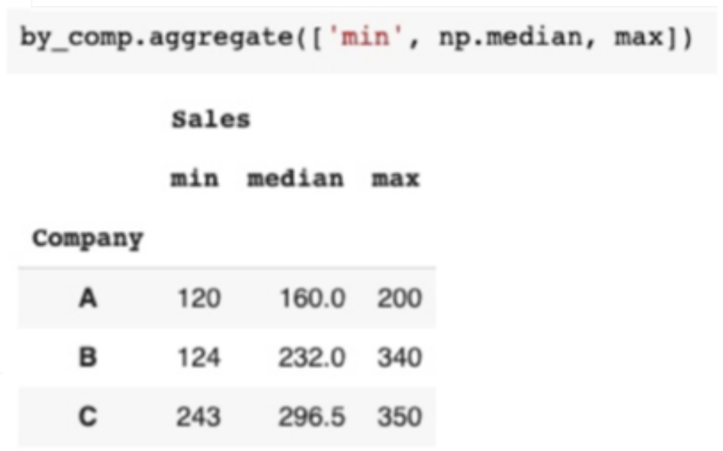
```python
data = {'Company':['A','A','B','B','C','C'],
        'Person':['Sam','Charlie','Amy','Vanessa','Carl','Sarah'],
        'Sales':[200,120,340,124,243,350]}
df = pd.DataFrame(data)
df
```

```
   Company   Person  Sales
0        A      Sam    200
1        A  Charlie    120
2        B      Amy    340
3        B  Vanessa    124
4        C     Carl    243
5        C    Sarah    350
```



```python
by_comp.min()
```

```
          Person  Sales
Company
A        Charlie    120
B            Amy    124
C           Carl    243
```

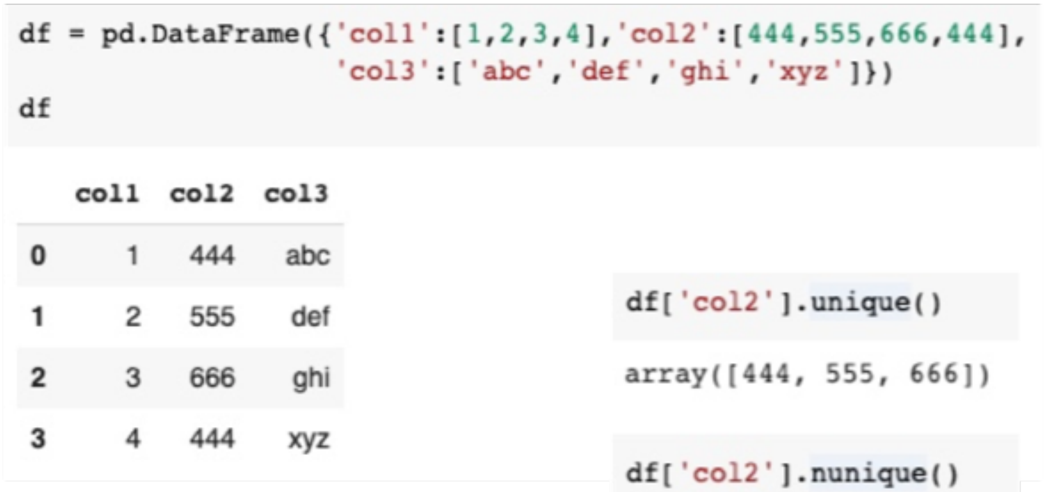


```python
by_comp.aggregate(['min', np.median, max])
```

```
         Sales
         min  median  max
Company
A        120   160.0  200
B        124   232.0  340
C        243   296.5  350
```

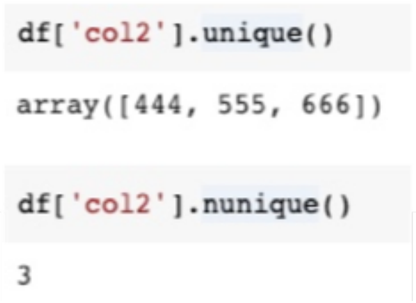
Operations

- Information on unique values
  - **.unique()** : unique values
  - **.nunique()**: count distinct values
  - **.value_counts()**: a Series containing counts of unique rows



```
df = pd.DataFrame({'col1':[1,2,3,4],'col2':[444,555,666,444],
                   'col3':['abc','def','ghi','xyz']})
df
```

	col1	col2	col3
0	1	444	abc
1	2	555	def
2	3	666	ghi
3	4	444	xyz

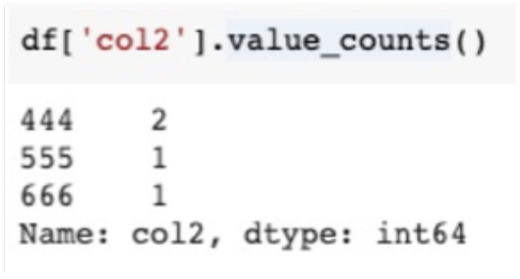


```
df['col2'].unique()
```

```
array([444, 555, 666])
```

```
df['col2'].nunique()
```

```
3
```

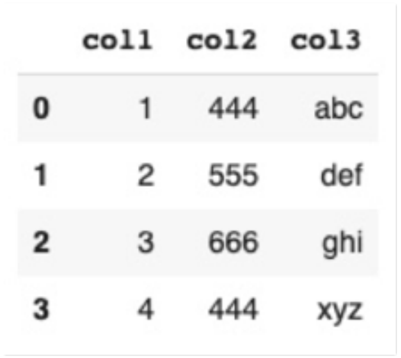


```
df['col2'].value_counts()
```

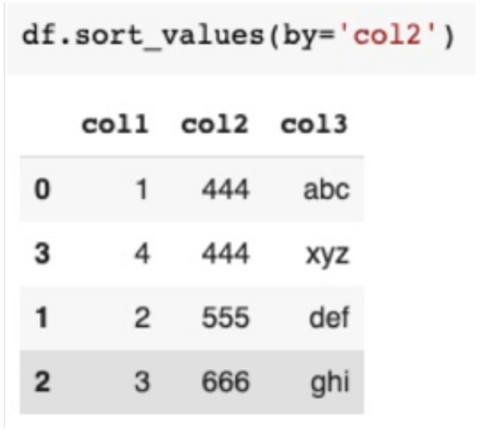
```
444    2
555    1
666    1
Name: col2, dtype: int64
```

Operations

- Sorting and Ordering
  - **.sort_values()**



	col1	col2	col3
0	1	444	abc
1	2	555	def
2	3	666	ghi
3	4	444	xyz

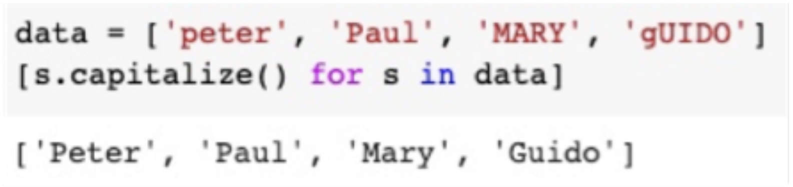


```
df.sort_values(by='col2')
```

	col1	col2	col3
0	1	444	abc
3	4	444	xyz
1	2	555	def
2	3	666	ghi

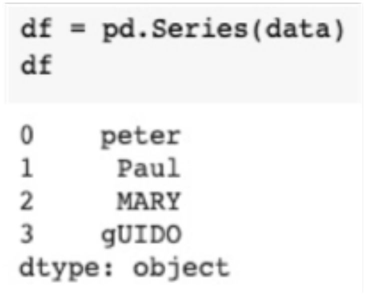
Operations

- String operations



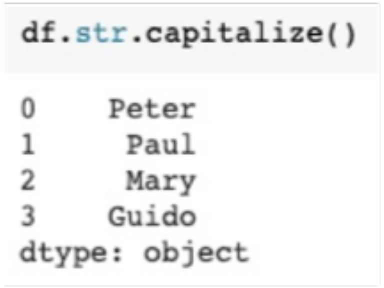
```
data = ['peter', 'Paul', 'MARY', 'gUIDO']
[s.capitalize() for s in data]
```

```
['Peter', 'Paul', 'Mary', 'Guido']
```



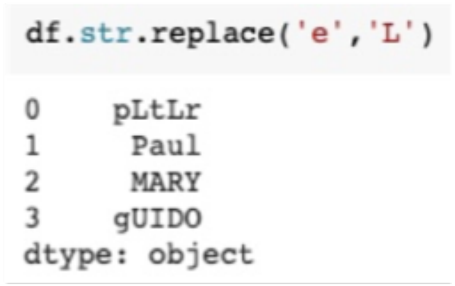
```
df = pd.Series(data)
df
```

```
0     peter
1      Paul
2      MARY
3     gUIDO
dtype: object
```



```
df.str.capitalize()
```

```
0     Peter
1      Paul
2      Mary
3     Guido
dtype: object
```



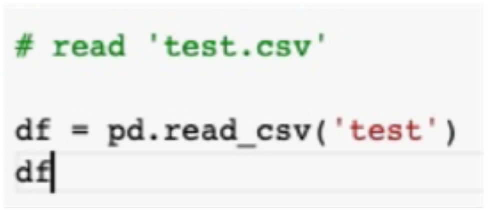
```
df.str.replace('e','L')
```

```
0     pLtLr
1      Paul
2      MARY
3     gUIDO
dtype: object
```

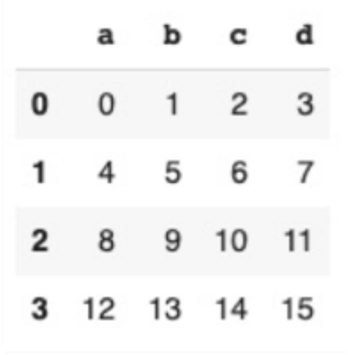
- len(), lower(), upper(), capitalize(), split(), strip(), replace() …

Data Input and Output

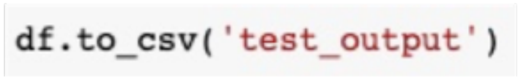
- CSV input/output
  - **.read_csv()**: read csv input file
  - **.to_csv()**: write csv output file



```
# read 'test.csv'

df = pd.read_csv('test')
df
```



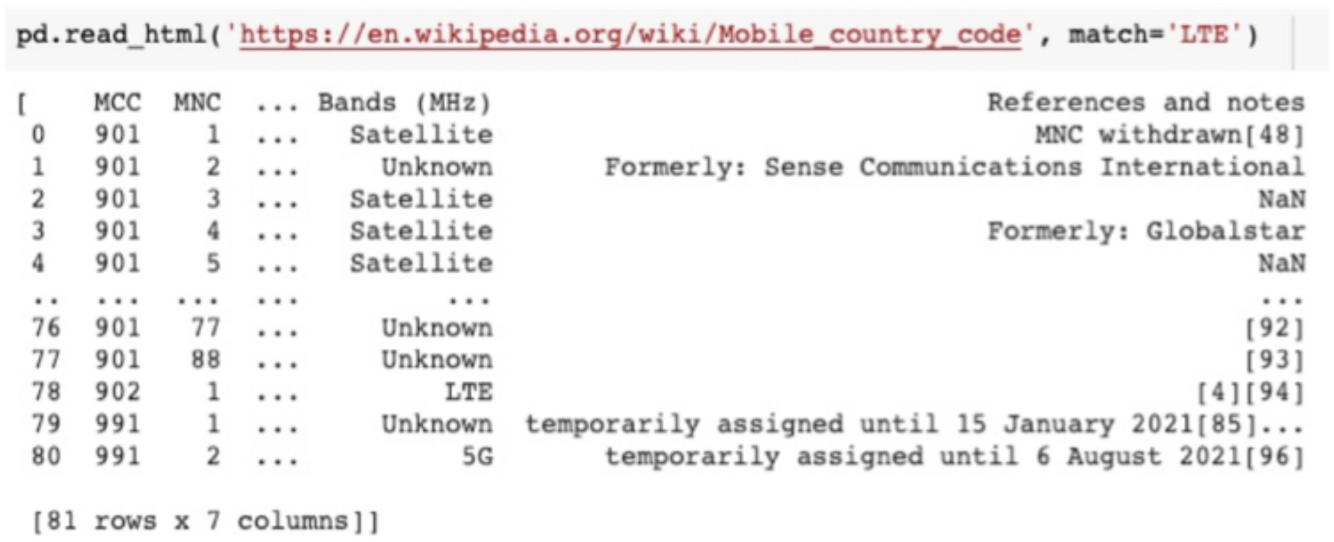
	a	b	c	d
0	0	1	2	3
1	4	5	6	7
2	8	9	10	11
3	12	13	14	15



```
df.to_csv('test_output')
```

Data Input and Output

- HTML data
  - Need to install html5lib, lxml, BeautifulSoup4
  - **.read_html()**: read html input



```
pd.read_html('https://en.wikipedia.org/wiki/Mobile_country_code', match='LTE')
```

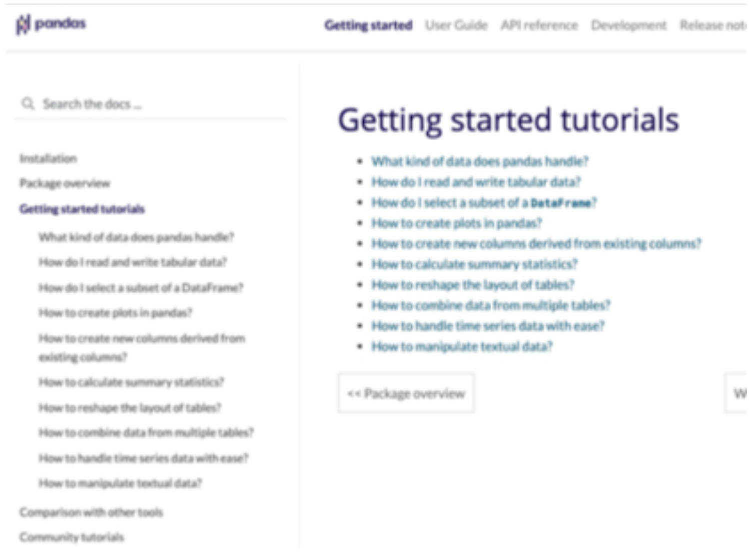
```
[     MCC  MNC  ... Bands (MHz)                              References and notes
 0    901    1  ...   Satellite                                MNC withdrawn[48]
 1    901    2  ...     Unknown    Formerly: Sense Communications International
 2    901    3  ...   Satellite                                              NaN
 3    901    4  ...   Satellite                            Formerly: Globalstar
 4    901    5  ...   Satellite                                              NaN
 ..   ...  ...  ...         ...                                              ...
 76   901   77  ...     Unknown                                             [92]
 77   901   88  ...     Unknown                                             [93]
 78   902    1  ...         LTE                                          [4][94]
 79   991    1  ...     Unknown  temporarily assigned until 15 January 2021[85]...
 80   991    2  ...          5G     temporarily assigned until 6 August 2021[96]

 [81 rows x 7 columns]]
```

In this lesson, you have learned:

- Pandas
- Missing Data
- GroupBy, Merging, Concatenating
- Operations
- Data Input/Output

pandas

pandas.pydata.org



pandas

Getting started User Guide API reference Development Release not

Search the docs ...

Installation

Package overview

Getting started tutorials

- What kind of data does pandas handle?
- How do I read and write tabular data?
- How do I select a subset of a DataFrame?
- How to create plots in pandas?
- How to create new columns derived from existing columns?
- How to calculate summary statistics?
- How to reshape the layout of tables?
- How to combine data from multiple tables?
- How to handle time series data with ease?
- How to manipulate textual data?

Comparison with other tools

Community tutorials

Getting started tutorials

- What kind of data does pandas handle?
- How do I read and write tabular data?
- How do I select a subset of a `DataFrame`?
- How to create plots in pandas?
- How to create new columns derived from existing columns?
- How to calculate summary statistics?
- How to reshape the layout of tables?
- How to combine data from multiple tables?
- How to handle time series data with ease?
- How to manipulate textual data?

<< Package overview

W

Thank you!

Any Questions?

hyeryung.jang@dgu.ac.kr