Lecture 5: Pandas

Course: Biomedical Data Science

Parisa Rashidi
Fall 2018

Agenda

- More Python
- Pandas
- Preprocessing structured data
- Preprocessing unstructured data (time series, text, ..)

Disclaimer

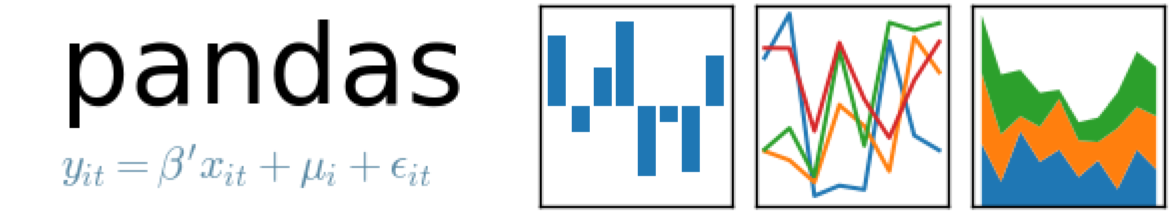
The following slides are based on:

STAT 504 Analytics, Stephen Lee

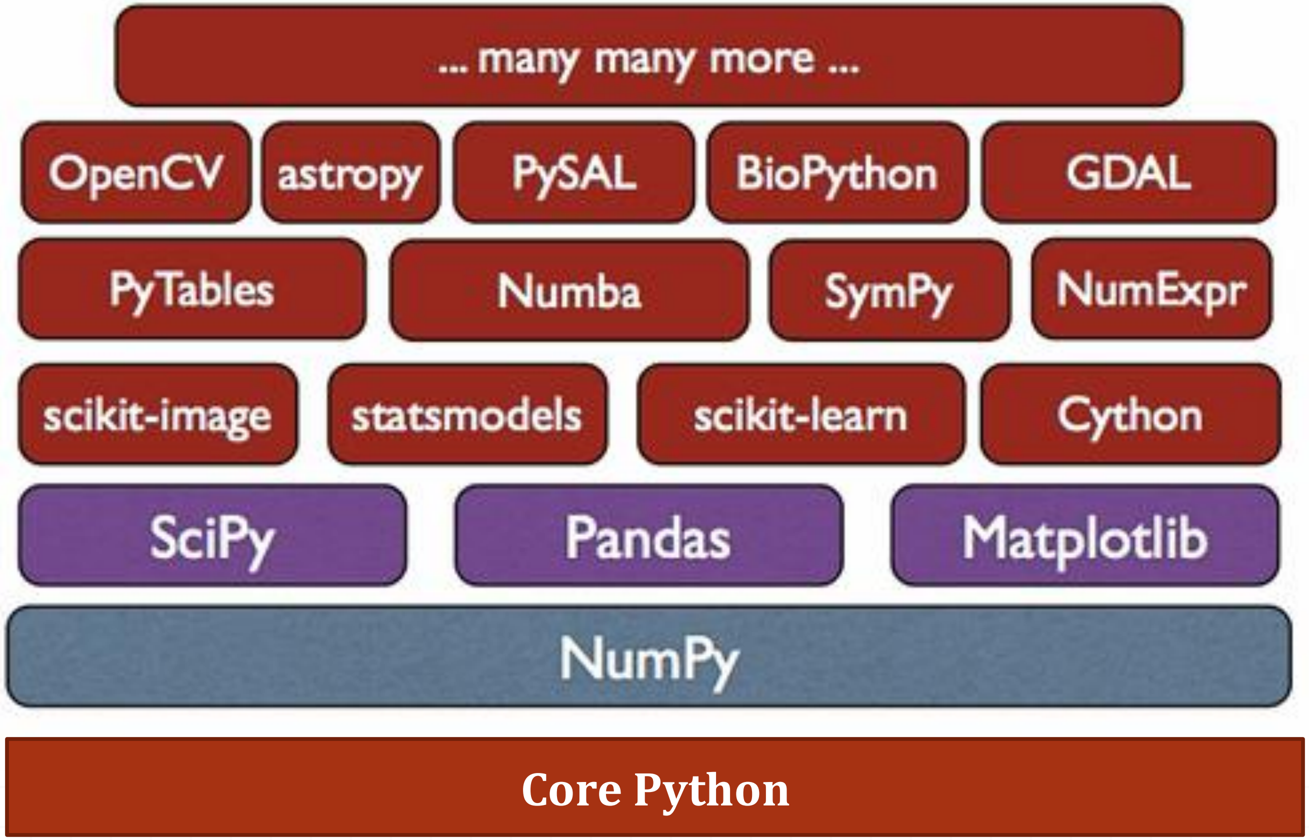
http://www.webpages.uidaho.edu/~stevel/stat504.htm

Why Pandas?

- NumPy is low-level library
- It allows us to deal with data in a user-friendly; using labelled columns and indexes
- It allows us to easily import data from files such as .csv files
- It allows us to perform complex functions on data
- ....



Python Library Stack



... many many more ...

OpenCV astropy PySAL BioPython GDAL

PyTables Numba SymPy NumExpr

scikit-image statsmodels scikit-learn Cython

SciPy Pandas Matplotlib

NumPy

Core Python

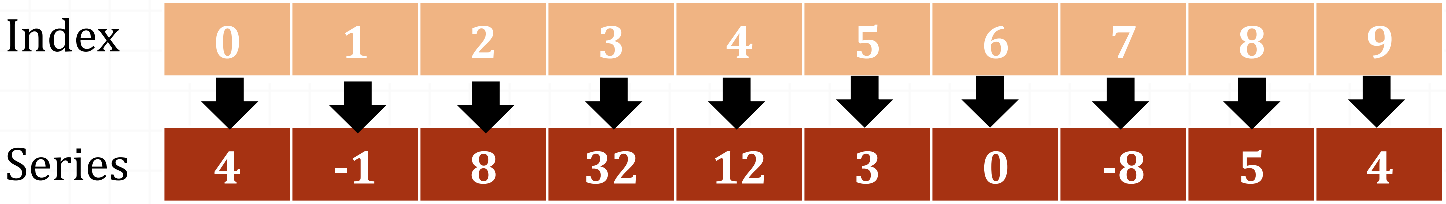
http://quasiben.github.io/PyDataEMC/#/9

Pandas Data Structures

- Series
- DataFrame

Series

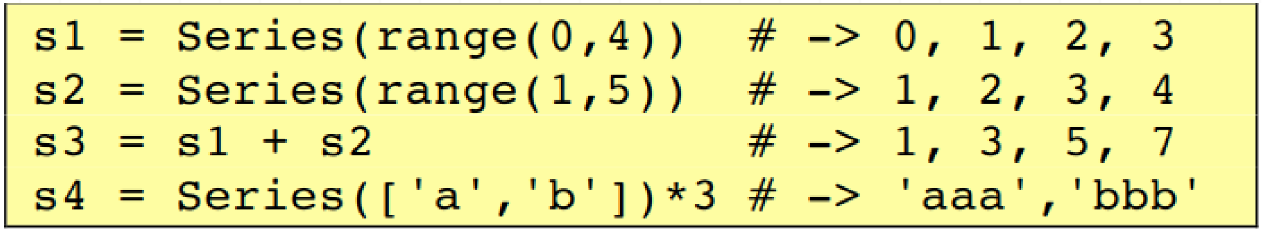
- An ordered, one-dimensional array of data with an index.
- All the data in a Series is of the same data type.



Index	0	1	2	3	4	5	6	7	8	9
Series	4	-1	8	32	12	3	0	-8	5	4

Series

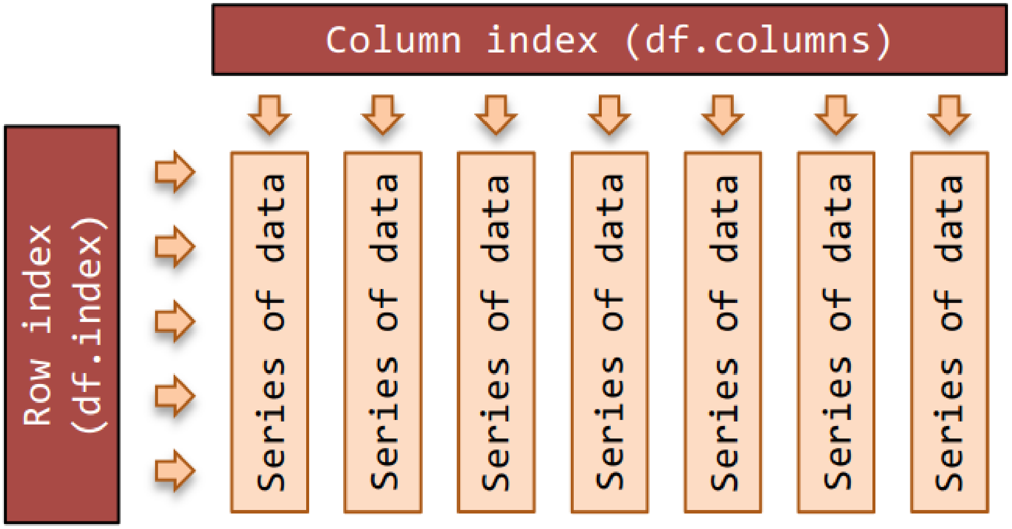
- Series arithmetic is vectorized.



```
s1 = Series(range(0,4))  # -> 0, 1, 2, 3
s2 = Series(range(1,5))  # -> 1, 2, 3, 4
s3 = s1 + s2             # -> 1, 3, 5, 7
s4 = Series(['a','b'])*3 # -> 'aaa','bbb'
```

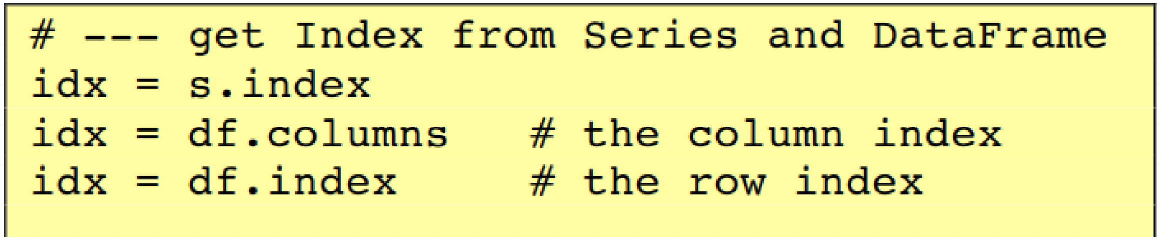
DataFrame

- The pandas DataFrame is a two-dimensional table of data with column and row indices.
- The columns are made up of pandas Series objects.



Index

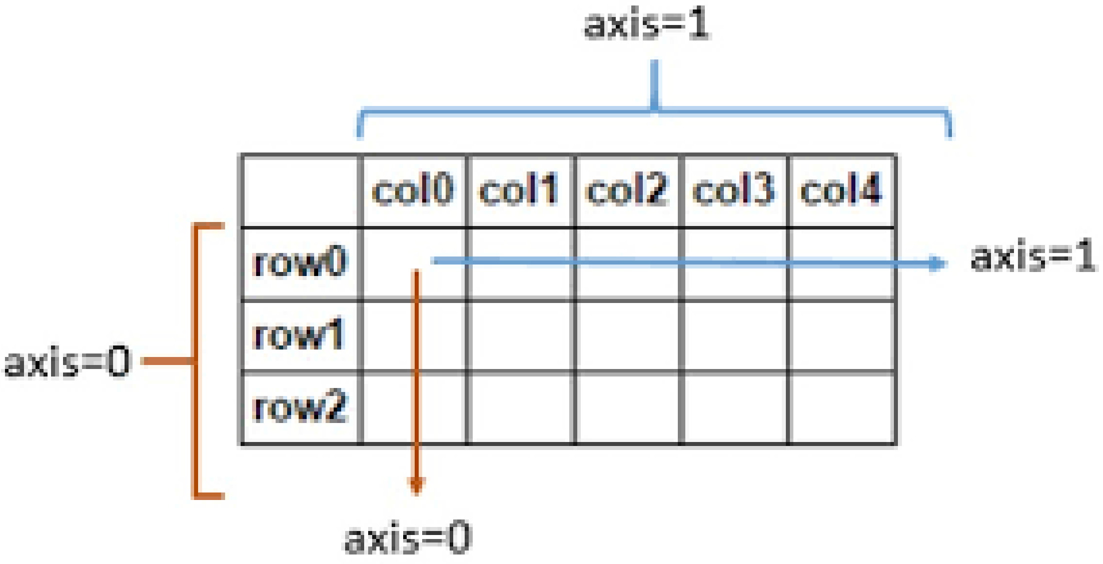
- The pandas Index provides the axis labels for the Series and DataFrame objects.
- A pandas Series has one Index; and a DataFrame has two Indices.



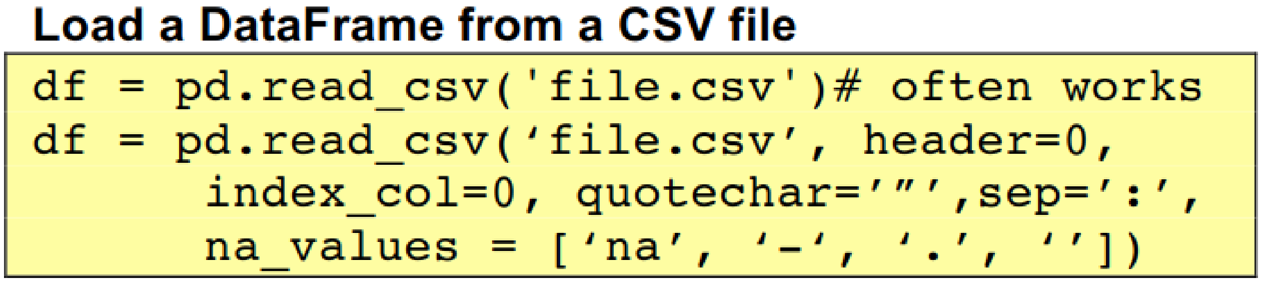
```
# --- get Index from Series and DataFrame
idx = s.index
idx = df.columns    # the column index
idx = df.index      # the row index
```

DataFrame

- Axis 0 and axis 1 are similar to NumPy's axes

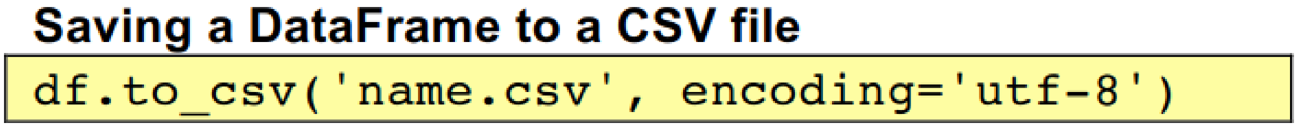


Reading and Writing Data



Load a DataFrame from a CSV file

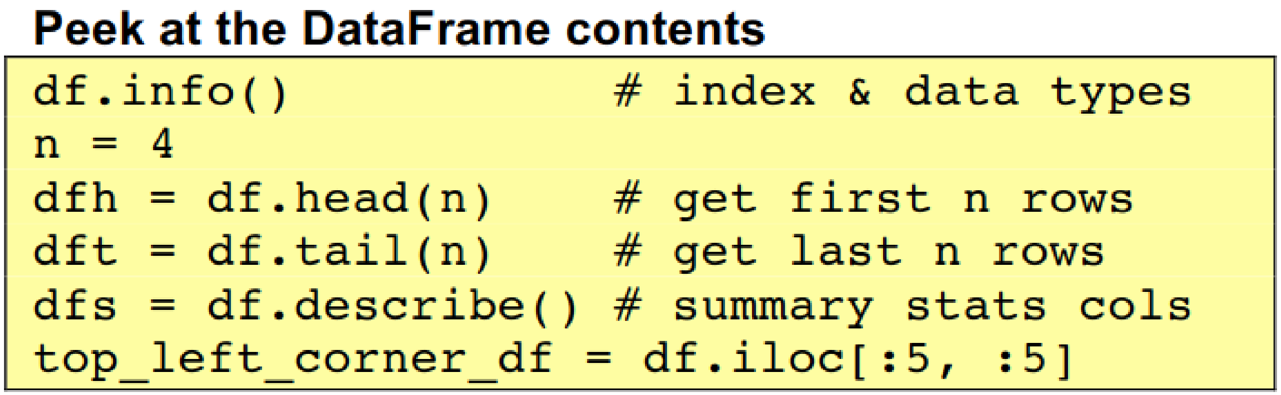
```
df = pd.read_csv('file.csv')# often works
df = pd.read_csv(‘file.csv’, header=0,
      index_col=0, quotechar=’”’,sep=’:’,
      na_values = [‘na’, ‘-‘, ‘.’, ‘’])
```



Saving a DataFrame to a CSV file

```
df.to_csv('name.csv', encoding='utf-8')
```

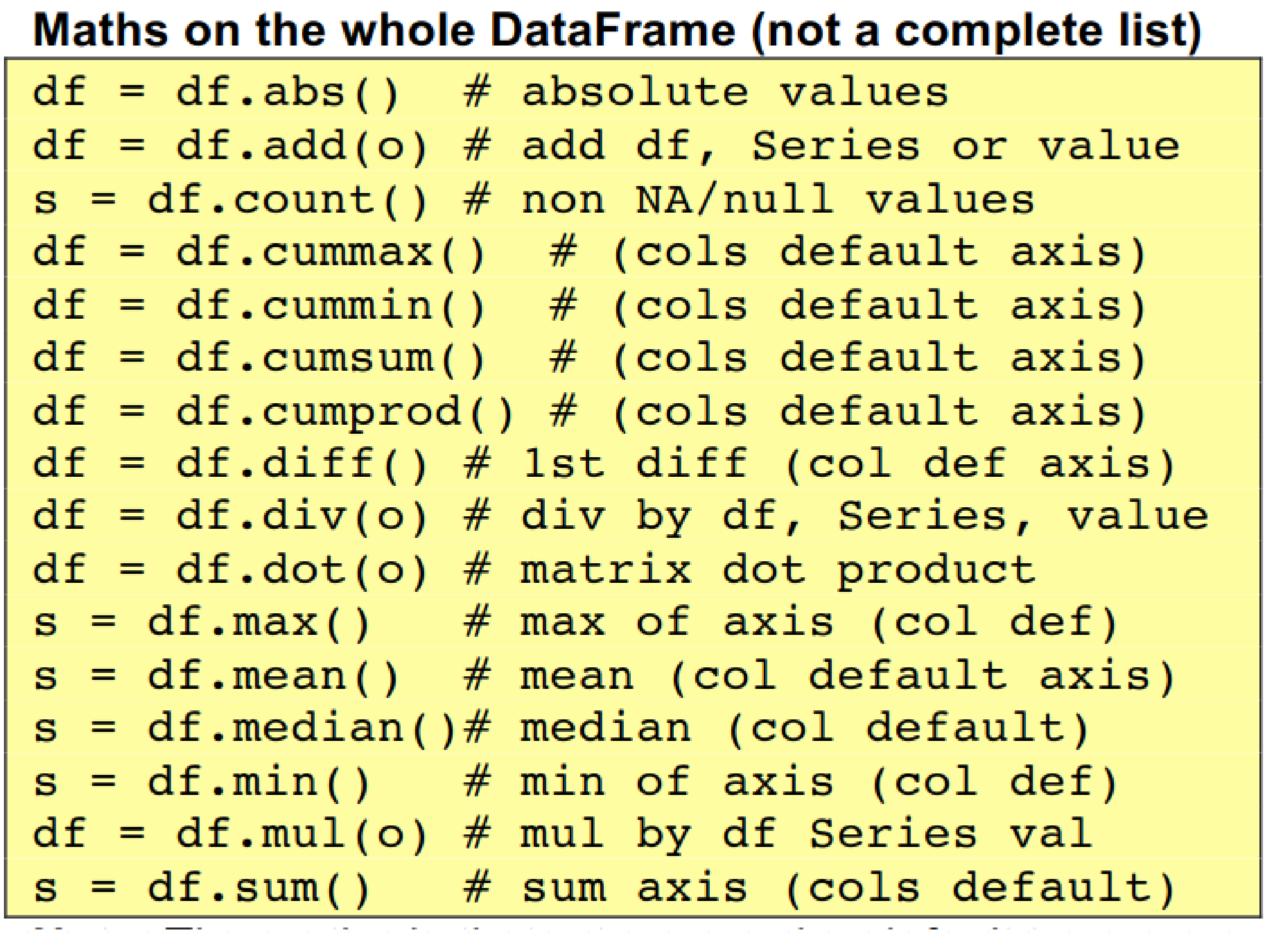
Working with Data



Peek at the DataFrame contents

```
df.info()           # index & data types
n = 4
dfh = df.head(n)    # get first n rows
dft = df.tail(n)    # get last n rows
dfs = df.describe() # summary stats cols
top_left_corner_df = df.iloc[:5, :5]
```

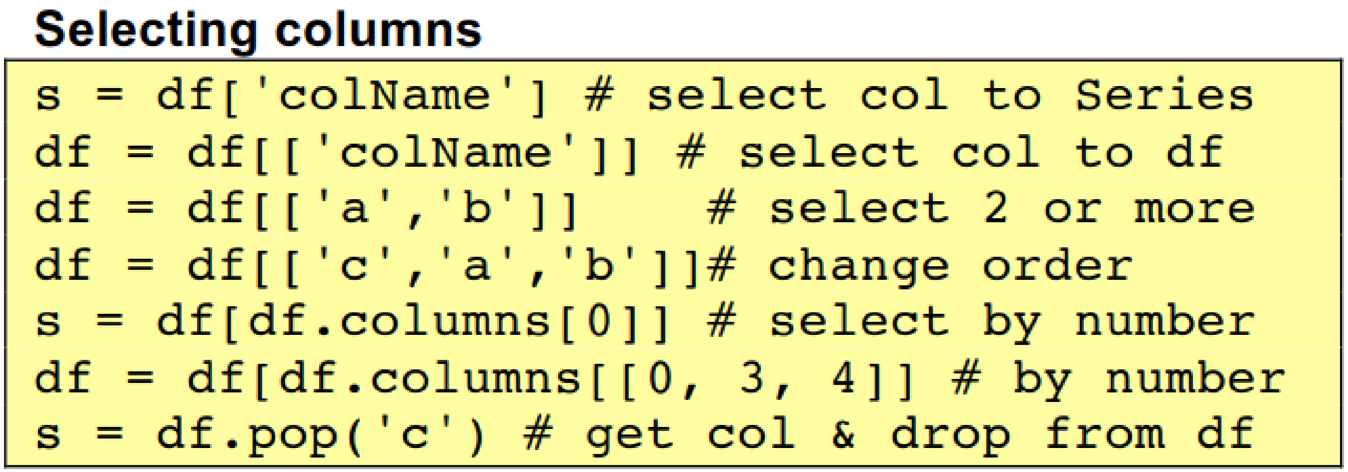
Working with Data



Maths on the whole DataFrame (not a complete list)

```
df = df.abs()  # absolute values
df = df.add(o) # add df, Series or value
s = df.count() # non NA/null values
df = df.cummax()  # (cols default axis)
df = df.cummin()  # (cols default axis)
df = df.cumsum()  # (cols default axis)
df = df.cumprod() # (cols default axis)
df = df.diff() # 1st diff (col def axis)
df = df.div(o) # div by df, Series, value
df = df.dot(o) # matrix dot product
s = df.max()   # max of axis (col def)
s = df.mean()  # mean (col default axis)
s = df.median()# median (col default)
s = df.min()   # min of axis (col def)
df = df.mul(o) # mul by df Series val
s = df.sum()   # sum axis (cols default)
```

Selecting Columns



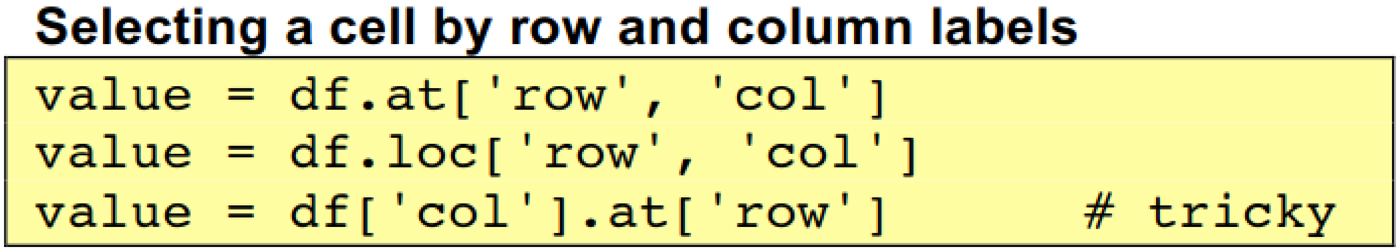
Selecting columns

```
s = df['colName'] # select col to Series
df = df[['colName']] # select col to df
df = df[['a','b']]    # select 2 or more
df = df[['c','a','b']]# change order
s = df[df.columns[0]] # select by number
df = df[df.columns[[0, 3, 4]] # by number
s = df.pop('c') # get col & drop from df
```

Selecting a Cell

We can select specific ranges of data in both the row and column directions using either label or integer-based indexing. We can use one of these methods:

- loc: indexing via labels
- iloc: indexing via integers
- at: returns a scalar



Selecting a cell by row and column labels

```
value = df.at['row', 'col']
value = df.loc['row', 'col']
value = df['col'].at['row']       # tricky
```