Hey, Python Pandas

Outline

- **What's Python Pandas & What's it for**
- **What's its features/capabilities**
- **Use case demo**
- References

*NOT a hands-on practice session

Reference

- Website Reference
  - pandas.pydata.org
  - freecodecamp
  - towardsdatascience.com
  - Céline Comte Nokia Bell Labs France & Télécom ParisTech Python Academy - May 20, 2019
  - Google, Stackoverflow, etc
- Youtube videos on Pandas
- Courses:
  - Udemy Data Analysis with Pandas and Python
  - Codecademy Learn Data Analysis with Pandas
  - Coursera: multiple classes, e.g. Data Analysis with Python

Intro. to Pandas

- Pandas is a widely-used open-source (Github link) Python library with user friendly data structure and data analysis tools for data analysis and data manipulation.
- Originally created by Wes McKinney in 2008.
- Pandas, the name is derived from the term “panel data”.
- Install Pandas library, import it and use it!

Pandas library dependencies

- **Pandas is built on top of NumPy library**
  - NumPy is used for efficient numerical operations on large quantities of data (multidimensional array, masks, matrices, etc).
  - There are a few functions that exist in NumPy that we use on Pandas.
- Pandas has other dependencies.

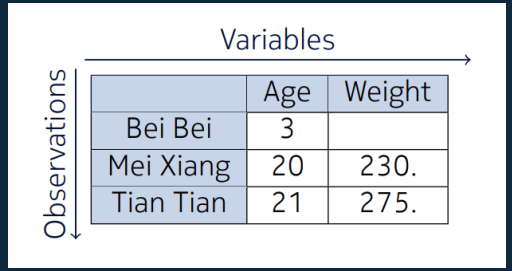
```
import pandas as pd
import numpy as np
```

Data Analysis Using Pandas

- **Data structures and tools designed to work with table-like data (i.e. spreadsheet; series and data frames in R)**
- **Data structures**
  - Series: 1-dimensional array with labels
  - DataFrame: 2-dimensional array with labels
- **Common operations with its provided tools for data analysis**
  - Converting data into Series or DataFrame, handling missing data, data alignment, reshaping, merging, sorting, slicing, aggregation/group by

Pandas data structures

- **Series**: One-dimensional array with axis labels (including time series).
- **DataFrame**: Two-dimensional size-mutable, potentially heterogeneous tabular data structure with labeled axes (rows and columns).
- **Philosophy**: it gives a semantical meaning to the axes
  - Columns ≃ Variables
  - Rows ≃ Observations



- **Series: some basic operations**
  - Create a Series
  - Add labels
  - Indexing
  - Merge two Series

- In Jupyter Notebook:
  - Import libraries
  - Create a few starter variables
  - Create a Series using .Series( ) method

```
import pandas as pd
import numpy as np
```

```
labels = ['a', 'b', 'c']
my_list = [10, 20, 30]
arr = np.array([10, 20, 30])
d = {'a':10, 'b':20, 'c':30}
```

```
pd.Series(my_list)
```

Output

```
0    10
1    20
2    30
dtype: int64
```

Colum of labels & column of data

Series: some basic operations

- Create a Series **(multiple ways available)**
- Add labels
- Indexing using [ ] to index its labels
- Merge two Series

```
labels = ['a', 'b', 'c']
my_list = [10, 20, 30]
arr = np.array([10, 20, 30])
d = {'a':10, 'b':20, 'c':30}
```

```
pd.Series(my_list, index=labels)
```

[7] ✓ 0.5s

Output

```
a    10
b    20
c    30
dtype: int64
```

- pass in a dictionary to create a pandas Series

```
pd.Series(d)
```

[8] ✓ 0.1s

```
a    10
b    20
c    30
dtype: int64
```

- Can reference an element of the Series using its label or its numerical index.

Data Munging

- **DataFrame: key operations**
  - Create a DataFrame
  - Define row labels and column labels
  - Access DataFrame Info.
  - Index and select data
  - Add/remove columns of data frame
  - Handle missing data,
  - Data alignment
  - Reshape and pivot tables
  - Merge, join, concatenate and compare
  - Sort
  - Group by
  - Plot and Visualization
  - Time series
  - Get data in/out, various file formats (e.g. csv)
  - ……..

DataFrame operations

- Create a DataFrame
  - From a python list, a dictionary, a csv file, a SQL query, etc
- Define row labels and column labels

```
import pandas as pd
import numpy as np
```

```
rows = ['X','Y','Z']
cols = ['A', 'B', 'C', 'D', 'E']
data = np.round(np.random.randn(3,5),2)
pd.DataFrame(data, rows, cols)
```

[10]

Output

	A	B	C	D	E
X	-0.42	1.72	-1.19	-1.52	-0.43
Y	0.35	-1.49	-0.19	1.28	0.87
Z	-0.41	-0.36	1.31	-0.97	-0.60

DataFrame operations

- Index and select data

df dataframe:

	A	B	C	D	E
X	0.09	0.97	-1.69	1.38	1.01
Y	0.07	-0.31	1.36	-0.04	1.43
Z	-1.38	-0.12	2.33	0.71	-1.52

- select column(s)

```
df = pd.DataFrame(data, rows, cols)
df['A']
```

[14] ✓ 0.6s

```
X    0.09
Y    0.07
Z   -1.38
Name: A, dtype: float64
```

```
df[['A', 'E']]
```

[16] ✓ 0.8s

	A	E
X	0.09	1.01
Y	0.07	1.43
Z	-1.38	-1.52

- select row(s)

```
df.loc["X"]
```

[18] ✓ 0.7s

```
A    0.09
B    0.97
C   -1.69
D    1.38
E    1.01
```

- select element

```
df['A']['X']
```

[15] ✓ 0.6s

```
0.09
```

DataFrame operations

- Add/remove columns of data frame

- create a new column called ‘A + B’ which is the sum of columns A and B

```
df['A + B'] = df['A'] + df['B']
df
```

[17] ✓ 0.2s

...

	A	B	C	D	E	A + B
X	0.09	0.97	-1.69	1.38	1.01	1.06
Y	0.07	-0.31	1.36	-0.04	1.43	-0.24
Z	-1.38	-0.12	2.33	0.71	-1.52	-1.50

- **DataFrame operation:** Handle missing data
  - df.dropna: Deletes columns or rows that contain missing values (NaN).
  - df.fillna: Fills the NaN with the provided value.
  - df.isna or pd.isna(df): Returns a DataFrame of the same size as df with boolean values that say if the original value in df is NaN.

```
df_1 = pd.DataFrame(np.array([[1, 5, 1],[2, np.nan, 2],[np.nan, np.nan, 3]]))

df_1.columns = ['A', 'B', 'C']

df_1
```

[19] ✓ 0.1s

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

```
df_1.dropna()
```

[20] ✓ 0.9s

	A	B	C
0	1.0	5.0	1.0

Take Home Messages:

- Pandas is a widely-used open-source Python library with user friendly data structure (series and dataframe) and data analysis tools for data analysis and manipulation.

- Google it or take courses to learn Pandas' powerful operations on Series and DataFrame when you need Pandas for data munging in Python.

Thank you for your attention!

Any Questions?