Panda Guideline
Basics

To start using Panda, we must import the followings:

```
import pandas as pd
```

Basic Data Structure

1. Series: one-dimension or array-like
2. DataFrame: two-dimension or table-like

Example: Series

```
sr = pd.Series([1,3,5,7])
sr
```

```
0    1
1    3
2    5
3    7
dtype: int64
```

```
sr2 = pd.Series([1,3,5,7], index=['A','B','C','D'])
sr2
```

```
A    1
B    3
C    5
D    7
dtype: int64
```

Example: DataFrame

```
data = {'ID' : ['1001', '1002', '1003','1004'],
        'Name': ['Ann', 'Beth', 'Cathlyn', 'David'],
        'GPA': [3.51, 2.75, 3.04, 2.24]}
df = pd.DataFrame(data)
df
```

	GPA	ID	Name
0	3.51	1001	Ann
1	2.75	1002	Beth
2	3.04	1003	Cathlyn
3	2.24	1004	David

```
data = {'ID' : ['1001', '1002', '1003','1004'],
        'Name': ['Ann', 'Beth', 'Cathlyn', 'David'],
        'GPA': [3.51, 2.75, 3.04, 2.24]}
df2 = pd.DataFrame(data, columns=['ID','Name','GPA'])
df2
```

	ID	Name	GPA
0	1001	Ann	3.51
1	1002	Beth	2.75
2	1003	Cathlyn	3.04
3	1004	David	2.24

Basic I/O

Panda can read/write many input/output format. For example, text (csv, json, html), binary (excel, stata, sas), or SQL.

read_FORMAT: to read input from file with specific FORMAT.
Example of read_FORMAT: read_csv, read_excel, read_sql
to_FORMAT: to write output to file with specific FORMAT.
Example of to_FORMAT: to_csv, to_excel, to_sql

Example:

```
df2.to_csv('student.csv',index=False)
```

```
df3 = pd.read_csv('student.csv')
df3
```

	ID	Name	GPA
0	1001	Ann	3.51
1	1002	Beth	2.75
2	1003	Cathlyn	3.04
3	1004	David	2.24

```
df4 = pd.read_csv('student.csv', sep=',', header='infer', index_col='ID')
df4
```

ID	Name	GPA
1001	Ann	3.51
1002	Beth	2.75
1003	Cathlyn	3.04
1004	David	2.24

Example of output file "student.csv"

```
ID,Name,GPA
1001,Ann,3.51
1002,Beth,2.75
1003,Cathlyn,3.04
1004,David,2.24
```

Basic Information

shape: to find out about dimension of data structure

columns: to list attributes of data frame

dtypes: to display data types of each attribute

head(): to display some first rows of data (default = 5 rows)

tail(): to display some last rows of data (default = 5 rows)

Note that shape and dtypes can be used with series also.

Example:

```
df3.shape
```

```
(4, 3)
```

```
df3.columns
```

```
Index(['ID', 'Name', 'GPA'], dtype='object')
```

```
df3.index
```

```
RangeIndex(start=0, stop=4, step=1)
```

```
df3.dtypes
```

```
ID        int64
Name     object
GPA     float64
dtype: object
```

```
df3.head()
```

	ID	Name	GPA
0	1001	Ann	3.51
1	1002	Beth	2.75
2	1003	Cathlyn	3.04
3	1004	David	2.24

```
sr.shape
```

```
(4,)
```

```
sr.dtypes
```

```
dtype('int64')
```

Basic Indexing

For DataFrame, the first and the second indices respectively are the row and the column indices. The integer indices range between 0 and length-1.

[ ] is used for indexing.

Inside [ ], it can be replaced

- column name > to retrieve such column.
- m:n > to retrieve data from the m^{th} to the $(n-1)^{th}$ row
- m: > to retrieve data from the m^{th} rows to the last rows
- :n > to retrieve data from the first n rows
- condition (see next section)

loc[,] : to locate specific rows or columns with label

iloc : to locate specific rows or columns with integer ranging between 0 and length-1.

Note that

- semicolon (:) can be used to identify all row/column indices
- axis = 0 > by row
- axis = 1 > by column

Example:

```
df3['Name']
```

```
0        Ann
1       Beth
2    Cathlyn
3      David
Name: Name, dtype: object
```

```
df3[1:3]
```

	ID	Name	GPA
1	1002	Beth	2.75
2	1003	Cathlyn	3.04

```
df3[1:]
```

	ID	Name	GPA
1	1002	Beth	2.75
2	1003	Cathlyn	3.04
3	1004	David	2.24

```
df3[:2]
```

	ID	Name	GPA
0	1001	Ann	3.51
1	1002	Beth	2.75

Example (cont.):

```
df.loc[2:3]
```

	GPA	ID	Name
2	3.04	1003	Cathlyn
3	2.24	1004	David

```
df3.loc[2:3, 'Name']
```

```
2    Cathlyn
3      David
Name: Name, dtype: object
```

```
df3.loc[:, 'Name']
```

```
0        Ann
1       Beth
2    Cathlyn
3      David
Name: Name, dtype: object
```

```
df4.loc[1002:1003, 'Name']
```

```
ID
1002       Beth
1003    Cathlyn
Name: Name, dtype: object
```

```
df4.iloc[:3]
```

	Name	GPA
ID		
1001	Ann	3.51
1002	Beth	2.75
1003	Cathlyn	3.04

```
df4.iloc[2:4, 0]
```

```
ID
1003    Cathlyn
1004      David
Name: Name, dtype: object
```

```
df3.iloc[[0,2],[0,2]]
```

	ID	GPA
0	1001	3.51
2	1003	3.04

Boolean Indexing

For DataFrame, inside [ ], it can be filled with Boolean condition.

isin: to find out attribute with specific value

Example:

```
df3[df3.Name == 'Beth']
```

	ID	Name	GPA
1	1002	Beth	2.75

```
df3[df3.GPA > 3]
```

	ID	Name	GPA
0	1001	Ann	3.51
2	1003	Cathlyn	3.04

```
df3[df3.GPA > 3]['Name']
```

```
0        Ann
2    Cathlyn
Name: Name, dtype: object
```

```
df3[(df3.GPA > 2.5) & (df3.GPA < 3.5) ]
```

	ID	Name	GPA
1	1002	Beth	2.75
2	1003	Cathlyn	3.04

```
df3[df3.Name.isin(['Ann','David'])]
```

	ID	Name	GPA
0	1001	Ann	3.51
3	1004	David	2.24

Sorting

sort_values(by=COLUMN_NAME, ascending=True, inplace=False): to sort data by column in ascending order and display the sorted result without changing the original data frame. If you would like the sorted result to replace the original, set inplace to be True.

sort_index(axis=0, ascending=True, inplace=False): to sort data by row ascendingly. If you would like to sort by column, set axis = 1.

Example:

```
df3.sort_values(by='GPA',ascending=False, inplace=False)
```

	ID	Name	GPA
0	1001	Ann	3.51
2	1003	Cathlyn	3.04
1	1002	Beth	2.75
3	1004	David	2.24

```
df3
```

	ID	Name	GPA
0	1001	Ann	3.51
1	1002	Beth	2.75
2	1003	Cathlyn	3.04
3	1004	David	2.24

```
df5 = df3.copy()
df5.sort_values(by='GPA',ascending=True,inplace=True)
df5
```

	ID	Name	GPA
3	1004	David	2.24
1	1002	Beth	2.75
2	1003	Cathlyn	3.04
0	1001	Ann	3.51

```
df3
```

	ID	Name	GPA
0	1001	Ann	3.51
1	1002	Beth	2.75
2	1003	Cathlyn	3.04
3	1004	David	2.24

```
df5.sort_index(axis=0, ascending=False)
```

	ID	Name	GPA
3	1004	David	2.24
2	1003	Cathlyn	3.04
1	1002	Beth	2.75
0	1001	Ann	3.51

```
df5.sort_index(axis=1)
```

	GPA	ID	Name
3	2.24	1004	David
1	2.75	1002	Beth
2	3.04	1003	Cathlyn
0	3.51	1001	Ann

Inserting

To insert a row, use loc with index = length and specified values.
To insert a column, use assign() with new column name and specified values.

Example:

```
df3.loc[4] = ['1005', 'David', '2.32']
df3
```

	ID	Name	GPA
0	1001	Ann	3.51
1	1002	Beth	2.75
2	1003	Cathlyn	3.04
3	1004	David	2.24
4	1005	David	2.32

```
df3 = df3.assign(Gender=['F','F','F','M','M'])
df3
```

	ID	Name	GPA	Gender
0	1001	Ann	3.51	F
1	1002	Beth	2.75	F
2	1003	Cathlyn	3.04	F
3	1004	David	2.24	M
4	1005	David	2.32	M

Data Type Conversion

pd.to_numeric(COLUMN) : To change specific COLUMN from one data to numeric type

Other useful data type conversion is to_datetime() to change data to datetime type.

Example:

```
df3.dtypes
```

```
ID         object
Name       object
GPA        object
Gender     object
Age         int64
dtype: object
```

```
df3.GPA = pd.to_numeric(df3.GPA)
df3.dtypes
```

```
ID          object
Name        object
GPA        float64
Gender      object
Age          int64
dtype: object
```

Descriptive Statistics

describe(): compute 8 descriptive statistics (count, mean, sd, min, max, Q1, median (Q2), Q3) of numerical data
mean(): compute mean of numerical data
median(): compute median of numerical data
mode(): compute mode of numerical data
min(): compute minimum of numerical data
max(): compute maximum of numerical data
value_counts(): compute frequencies of each unique data

Example:

```
df3.describe()
```

	GPA	Age
count	5.000000	5.00000
mean	2.772000	20.80000
std	0.525424	1.30384
min	2.240000	19.00000
25%	2.320000	20.00000
50%	2.750000	21.00000
75%	3.040000	22.00000
max	3.510000	22.00000

```
df3.mean()
```

```
GPA     2.772
Age    20.800
dtype: float64
```

```
df3.median()
```

```
ID     1003.00
GPA       2.75
Age      21.00
dtype: float64
```

```
df3.Age.mode()
```

```
0    22
dtype: int64
```

```
df3.GPA.max()
```

```
3.5099999999999998
```

```
df3.Age.min()
```

```
19
```

```
df3.Age.value_counts()
```

```
22    2
21    1
20    1
19    1
Name: Age, dtype: int64
```

Basic Histogram

To display plot in notebook, the command %matplotlib inline must be added.

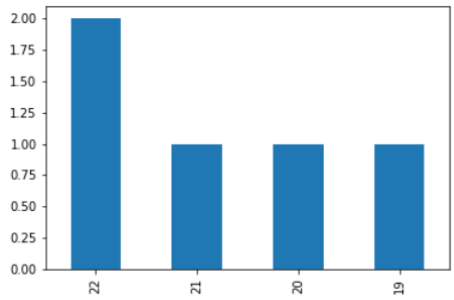
```
%matplotlib inline
```

plot.bar(): create bar plot from frequencies of data
hist(): create histogram plot from samples

Example:

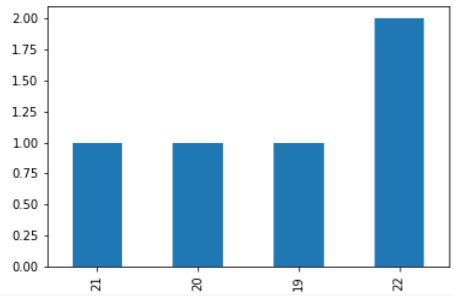
```
df3.Age.value_counts().plot.bar()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1ed227bda20>
```



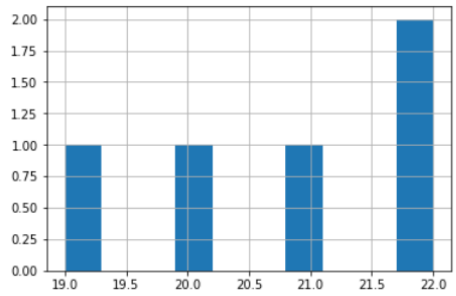

```
df3.Age.value_counts().sort_values().plot.bar()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1ed22e36b70>
```




```
df3.Age.hist()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1ed227eb668>
```



Other Usefule Commands

chdir(DIR): to change working directory to DIR directory
getcwd(): to display the current working directory

Example:

```
import os
os.chdir('C:\\')
os.getcwd()
```

```
'C:\\'
```

References

1. Panda documentation: http://pandas.pydata.org/pandas-docs/version/0.19/index.html
2. Introduction to Pandas: http://www.ritchieng.com/pandas-introduction/
3. Python Pandas Tutorials: https://www.tutorialspoint.com/python_pandas/index.htm