Data Processing Using Python

Data retrieval and preprocessing of Python

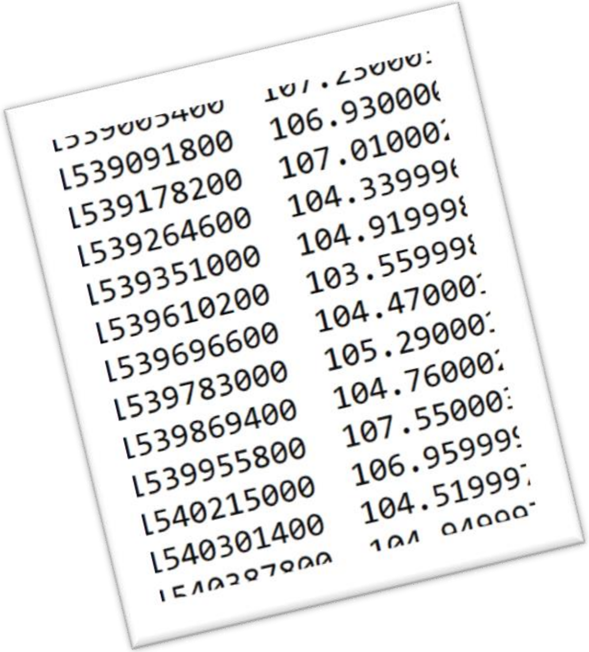
ZHANG Li/Dazhuang
Nanjing University
Department of Computer Science and Technology
Department of University Basic Computer Teaching

Basic Data Processing Procedure

Data Processing Using Python

1 CONVENIENT AND FAST DATA ACQUISITION

Fetch Data with Python

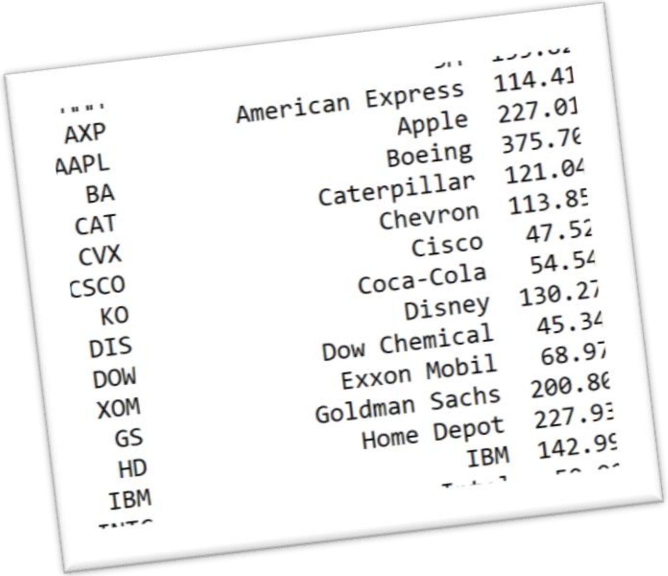


How to get local data?

Open, read/write, close of file

- File open
- File read
- File write
- File close

Fetch Data with Python



AXP	American Express	114.41
AAPL	Apple	227.01
BA	Boeing	375.76
CAT	Caterpillar	121.04
CVX	Chevron	113.85
CSCO	Cisco	47.52
KO	Coca-Cola	54.54
DIS	Disney	130.27
DOW	Dow Chemical	45.34
XOM	Exxon Mobil	68.97
GS	Goldman Sachs	200.86
HD	Home Depot	227.93
IBM	IBM	142.99

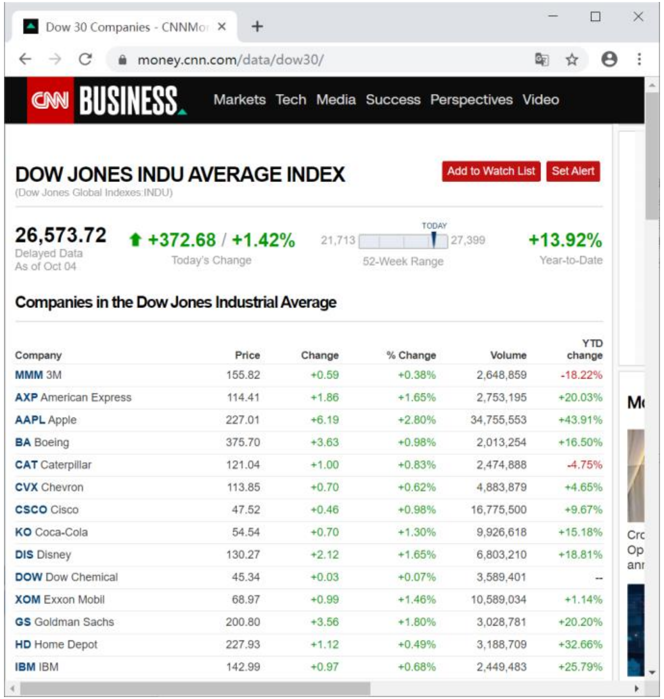
How to get (crawl) data from net?

Crawl pages and interpret content

- Crawling
  - **Urllib** built-in module
    - urllib.request
  - **Requests**

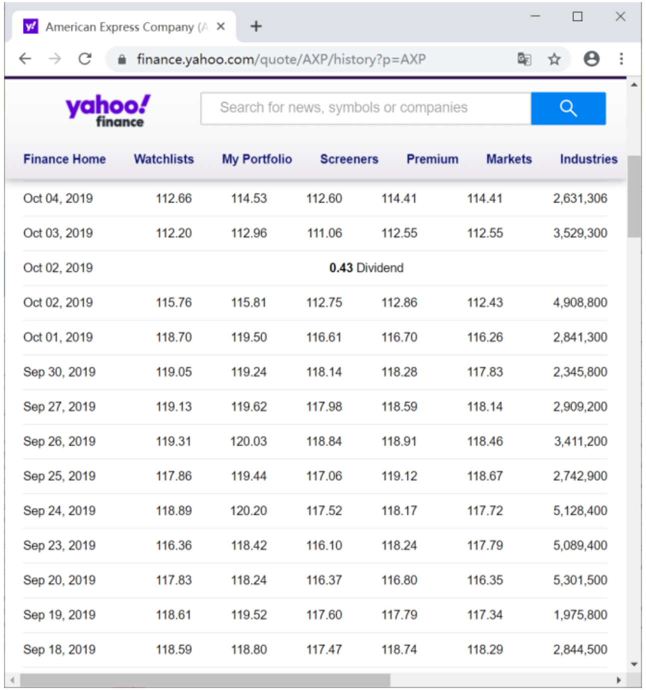
    (third party library)
  - **Scrapy** framework
- Interpreting
  - **BeautifulSoup** library
  - **re** module

Dow Jones Constituent



Company	Price	Change	% Change	Volume	YTD change
MMM 3M	155.82	+0.59	+0.38%	2,648,859	-18.22%
AXP American Express	114.41	+1.86	+1.65%	2,753,195	+20.03%
AAPL Apple	227.01	+6.19	+2.80%	34,755,553	+43.91%
BA Boeing	375.70	+3.63	+0.98%	2,013,254	+16.50%
CAT Caterpillar	121.04	+1.00	+0.83%	2,474,888	-4.75%
CVX Chevron	113.85	+0.70	+0.62%	4,883,879	+4.65%
CSCO Cisco	47.52	+0.46	+0.98%	16,775,500	+9.67%
KO Coca-Cola	54.54	+0.70	+1.30%	9,926,618	+15.18%
DIS Disney	130.27	+2.12	+1.65%	6,803,210	+18.81%
DOW Dow Chemical	45.34	+0.03	+0.07%	3,589,401	--
XOM Exxon Mobil	68.97	+0.99	+1.46%	10,589,034	+1.14%
GS Goldman Sachs	200.80	+3.56	+1.80%	3,028,781	+20.20%
HD Home Depot	227.93	+1.12	+0.49%	3,188,709	+32.66%
IBM IBM	142.99	+0.97	+0.68%	2,449,483	+25.79%

dji



Oct 04, 2019	112.66	114.53	112.60	114.41	114.41	2,631,306
Oct 03, 2019	112.20	112.96	111.06	112.55	112.55	3,529,300
Oct 02, 2019			0.43 Dividend			
Oct 02, 2019	115.76	115.81	112.75	112.86	112.43	4,908,800
Oct 01, 2019	118.70	119.50	116.61	116.70	116.26	2,841,300
Sep 30, 2019	119.05	119.24	118.14	118.28	117.83	2,345,800
Sep 27, 2019	119.13	119.62	117.98	118.59	118.14	2,909,200
Sep 26, 2019	119.31	120.03	118.84	118.91	118.46	3,411,200
Sep 25, 2019	117.86	119.44	117.06	119.12	118.67	2,742,900
Sep 24, 2019	118.89	120.20	117.52	118.17	117.72	5,128,400
Sep 23, 2019	116.36	118.42	116.10	118.24	117.79	5,089,400
Sep 20, 2019	117.83	118.24	116.37	116.80	116.35	5,301,500
Sep 19, 2019	118.61	119.52	117.60	117.79	117.34	1,975,800
Sep 18, 2019	118.59	118.80	117.47	118.74	118.29	2,844,500

quotes

Data Format

	code	name	price
0	MMM	3M	155.82
1	AXP	American Express	114.41
2	AAPL	Apple	227.01
3	BA	Boeing	375.70
4	CAT	Caterpillar	121.04
5	CVX	Chevron	113.85
6	CSCO	Cisco	47.52
7	KO	Coca-Cola	54.54
8	DIS	Disney	130.27
9	DOW	Dow Chemical	45.34
10	XOM	Exxon Mobil	68.97
11	GS	Goldman Sachs	200.80
12	HD	Home Depot	227.93
13	IBM	IBM	142.99
14	INTC	Intel	50.92
15	JNJ	Johnson & Johnson	133.66
16	JPM	JPMorgan Chase	114.62
17	MCD	McDonald's	211.69
18	MRK	Merck	85.00
19	MSFT	Microsoft	138.12
20	NKE	Nike	93.07
21	PFE	Pfizer	35.93
22	PG	Procter & Gamble	124.00
23	TRV	Travelers Companies Inc	144.96
24	UTX	United Technologies	133.21
25	UNH	UnitedHealth	219.80
26	VZ	Verizon	59.90
27	V	Visa	175.98
28	WMT	Wal-Mart	118.16
29	WBA	Walgreen	52.97

djidf

	close	date	high	low	open	volume
0	106.989998	1539005400	107.230003	105.570000	106.629997	2723400
1	106.660004	1539091800	106.930000	105.940002	106.300003	2604400
2	103.570000	1539178200	107.010002	103.519997	106.959999	4555600
3	101.580002	1539264600	104.339996	101.550003	103.220001	6069300
4	103.000000	1539351000	104.919998	101.709999	104.309998	4855900
5	102.620003	1539610200	103.559998	102.209999	102.849998	2780900
6	104.269997	1539696600	104.470001	102.690002	103.070000	3121000
7	104.339996	1539783000	105.290001	103.919998	104.330002	3792400
8	102.839996	1539869400	104.760002	102.290001	104.529999	4538200
9	106.730003	1539955800	107.550003	104.059998	104.059998	5726300
10	104.510002	1540215000	106.959999	104.449997	106.610001	5003100
11	104.379997	1540301400	104.519997	101.839996	102.410004	4223800
12	101.839996	1540387800	104.949997	101.510002	104.430000	4056700
13	103.599998	1540474200	104.169998	101.800003	102.480003	3378900
14	101.250000	1540560600	102.660004	100.139999	102.540001	5395700
15	101.190002	1540819800	103.250000	100.040001	102.470001	4238700
16	102.080002	1540906200	102.389999	100.410004	101.599998	3778200
17	102.730003	1540992600	103.709999	102.550003	103.059998	4511300
18	104.040001	1541079000	104.269997	103.019997	103.260002	2786800
19	103.709999	1541165400	105.050003	102.889999	104.930000	4322200

quotesdf

Download Data Directly

- How to easily and rapidly fetch historical data of companies from financial websites?

Time Period: Oct 05, 2018 - Oct 05, 2019 Show: Historical Prices Frequency: Daily Apply

Currency in USD

Download Data

Date	Open	High	Low	Close	Adj Close	Volume
2018/10/5	108.06	108.47	106.72	107.23	105.6803	2399800
2018/10/8	106.63	107.23	105.57	106.99	105.4437	2723400
2018/10/9	106.3	106.93	105.94	106.66	105.1185	2604400
2018/10/10	106.96	107.01	103.52	103.57	102.0732	4555600
2018/10/11	103.22	104.34	101.55	101.58	100.1119	6069300
2018/10/12	104.31	104.92	101.71	103	101.5114	4855900

File

```
# Filename: quotes_fromcsv.py
import pandas as pd
quotesdf = pd.read_csv('axp.csv')
print(quotesdf)
```

Read and Write of csv Format

- Store the basic stock information of American Express in the past year into stockAXP.csv.

File

```
# Filename: to_csv.py
import pandas as pd
…
quotes = retrieve_quotes_historical('AXP')
df = pd.DataFrame(quotes)
df.to_csv('stockAXP.csv')
```

Read and Write of Excel Data

File

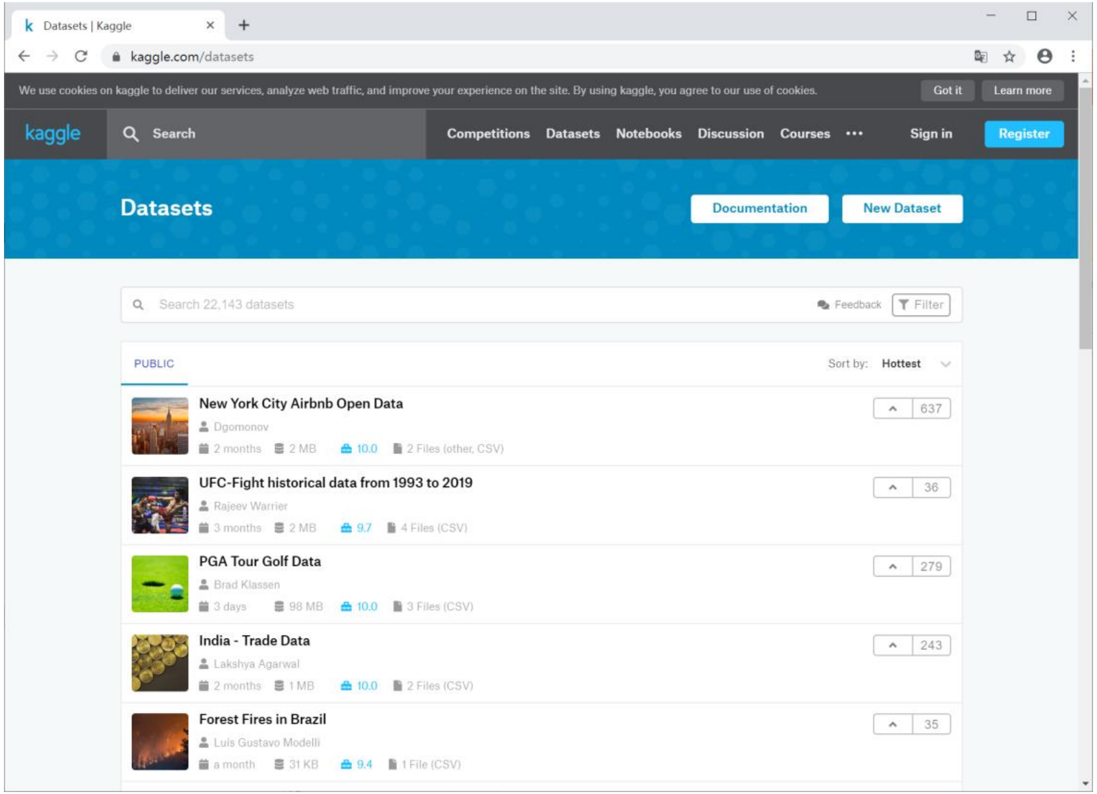
```python
# Filename: to_excel.py
…
quotes = retrieve_quotes_historical('AXP')
df = pd.DataFrame(quotes)
df.to_excel('stockAXP.xlsx', sheet_name = 'AXP')
```

File

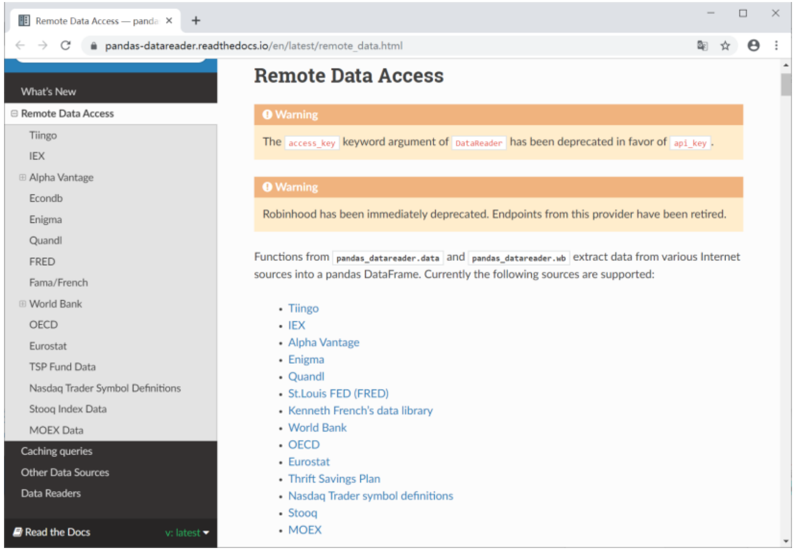
```python
# Filename: read_excel.py
…
df = pd.read_excel('stockAXP.xlsx', index_col = 'date')
print(df['close'][:3])
```

```
0     106.989998
1     106.660004
2     103.570000
Name: close, dtype: float64
```

Download Data Directly



Get Data Using API



Source

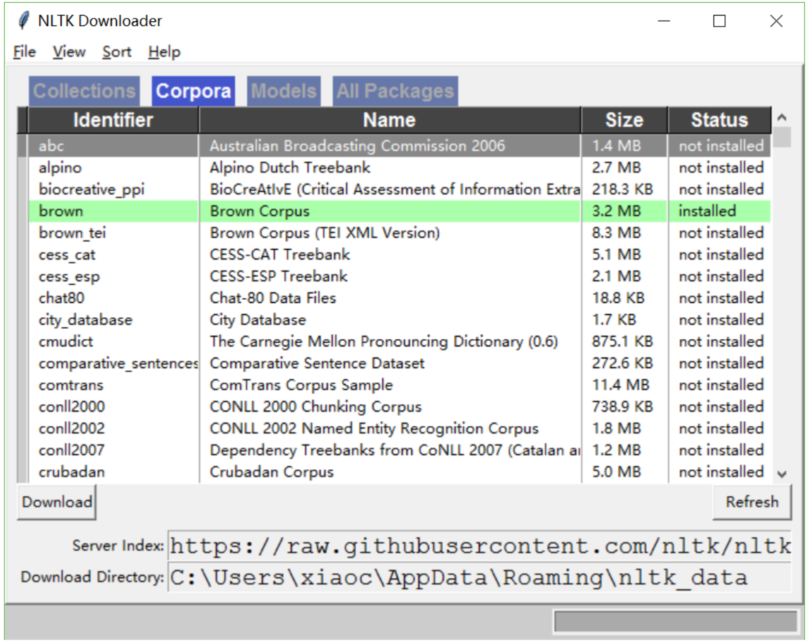
```
>>> import pandas_datareader.data as web
>>> f = web.DataReader('AXP', 'stooq')
>>> f.head(5)
Open    High    Low   Close   Volume
Date
2019-10-04  112.62  114.530  112.60  114.41  2753195
2019-10-03  112.52  112.955  111.06  112.55  3549232
2019-10-02  115.76  115.810  112.75  112.86  4931560
2019-10-01  118.70  119.500  116.61  116.70  2857528
2019-09-30  119.05  119.240  118.14  118.28  2353731
```

Using Datasets Module in Sklearn

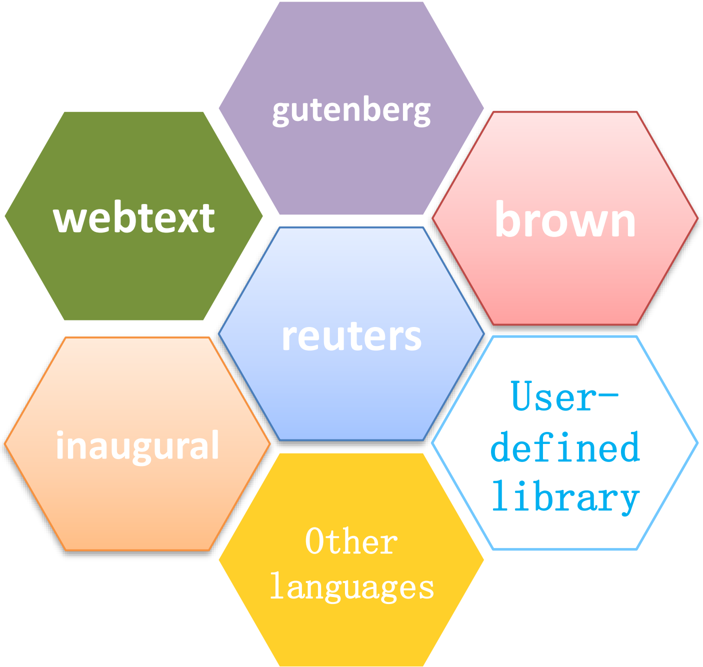
Source

```
>>> from sklearn import datasets
>>> iris = datasets.load_iris()
>>> iris.feature_names
['sepal length (cm)',
 'sepal width (cm)',
 'petal length (cm)',
 'petal width (cm)']
>>> iris.data
array([[5.1, 3.5, 1.4, 0.2],
       [4.9, 3. , 1.4, 0.2],
       [4.7, 3.2, 1.3, 0.2],
       ...
       [6.5, 3. , 5.2, 2. ],
       [6.2, 3.4, 5.4, 2.3],
       [5.9, 3. , 5.1, 1.8]])
```

NLTK library



Identifier	Name	Size	Status
abc	Australian Broadcasting Commission 2006	1.4 MB	not installed
alpino	Alpino Dutch Treebank	2.7 MB	not installed
biocreative_ppi	BioCreAtIvE (Critical Assessment of Information Extra	218.3 KB	not installed
brown	Brown Corpus	3.2 MB	installed
brown_tei	Brown Corpus (TEI XML Version)	8.3 MB	not installed
cess_cat	CESS-CAT Treebank	5.1 MB	not installed
cess_esp	CESS-ESP Treebank	2.1 MB	not installed
chat80	Chat-80 Data Files	18.8 KB	not installed
city_database	City Database	1.7 KB	not installed
cmudict	The Carnegie Mellon Pronouncing Dictionary (0.6)	875.1 KB	not installed
comparative_sentences	Comparative Sentence Dataset	272.6 KB	not installed
comtrans	ComTrans Corpus Sample	11.4 MB	not installed
conll2000	CONLL 2000 Chunking Corpus	738.9 KB	not installed
conll2002	CONLL 2002 Named Entity Recognition Corpus	1.8 MB	not installed
conll2007	Dependency Treebanks from CoNLL 2007 (Catalan a	1.2 MB	not installed
crubadan	Crubadan Corpus	5.0 MB	not installed



Easier Approach to Data

Source

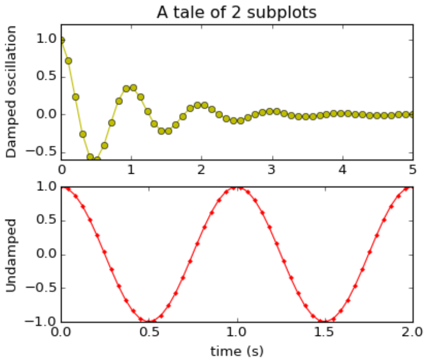
```
>>> from nltk.corpus import gutenberg
>>> import nltk
>>> print(gutenberg.fileids())
['austen-emma.txt', 'austen-persuasion.txt', 'austen-sense.txt', 'bible-kjv.txt', 'blake-poems.txt', 'bryant-stories.txt', 'burgess-busterbrown.txt', 'carroll-alice.txt', 'chesterton-ball.txt', 'chesterton-brown.txt', 'chesterton-thursday.txt', 'edgeworth-parents.txt', 'melville-moby_dick.txt', 'milton-paradise.txt', 'shakespeare-caesar.txt', 'shakespeare-hamlet.txt', 'shakespeare-macbeth.txt', 'whitman-leaves.txt']
>>> texts = gutenberg.words('shakespeare-hamlet.txt')
>>> print(texts)
['[', 'The', 'Tragedie', 'of', 'Hamlet', 'by', ...]
```

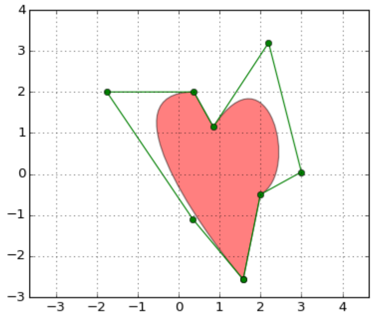
brown

Data Processing Using Python

2 FUNDAMENTALS OF PYTHON PLOTTING

Matplotlib Plotting





- **Matplotlib Plotting**

Most famous Python 2D plotting library

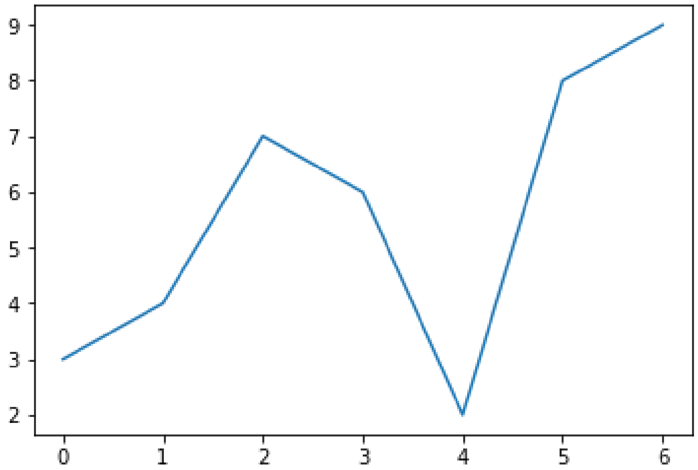
- High quality
- Convenient plotting modules
  - Plotting API——pyplot module

Line Chart

Source

```
>>> import matplotlib.pyplot as plt
>>> plt.plot([3, 4, 7, 6, 2, 8, 9])
```

```
plt.plot(range(7), [3, 4, 7, 6, 2, 8, 9])
```

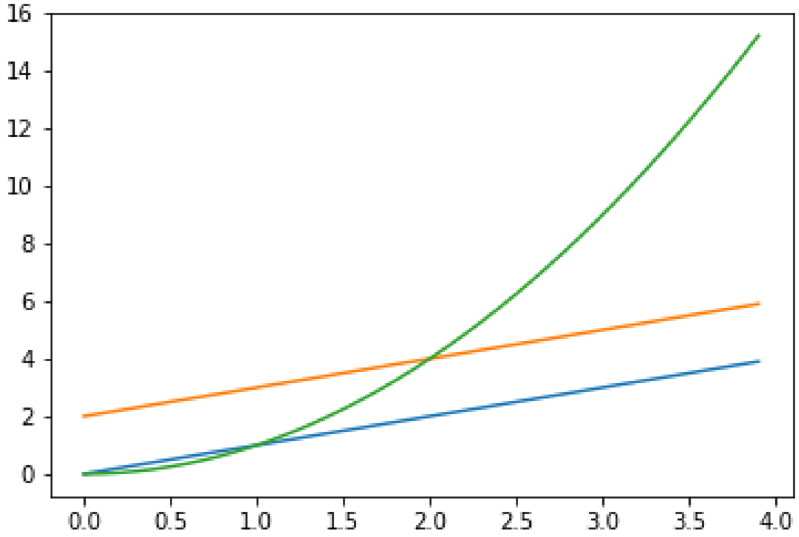


Line Chart – for groups of data

- **NumPy** array can also be used as a parameter of **Matplotlib**
- **Groups data** plotting

Source

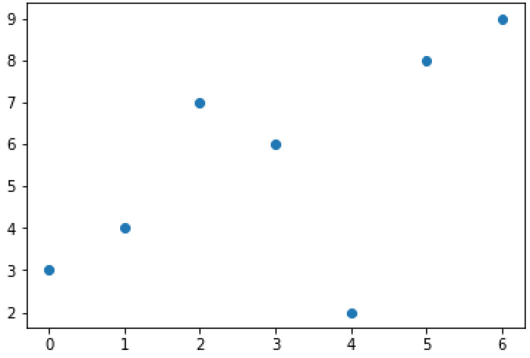
```
>>> import numpy as np
>>> import matplotlib.pyplot as plt
>>> t=np.arange(0.,4.,0.1)
>>> plt.plot(t, t, t, t+2, t, t**2)
```

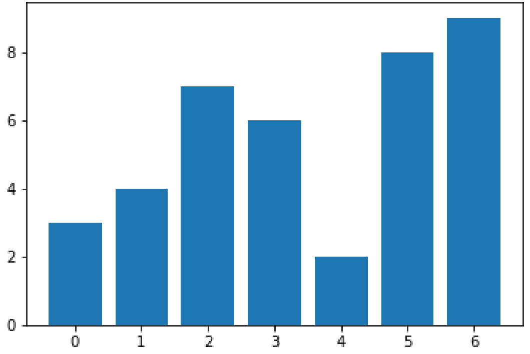


Different plot forms

Source

```
>>> import matplotlib.pyplot as plt
>>> plt.scatter(range(7), [3, 4, 7, 6, 2, 8, 9])
>>> plt.bar(range(7), [3, 4, 7, 6, 2, 8, 9])
```





Matplotlib Attributes

……

Character attributes

Grid attributes

axes

subplots

Color and style

Line width

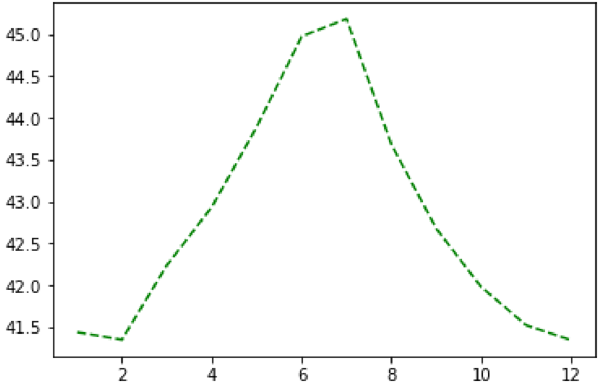
Point per inch

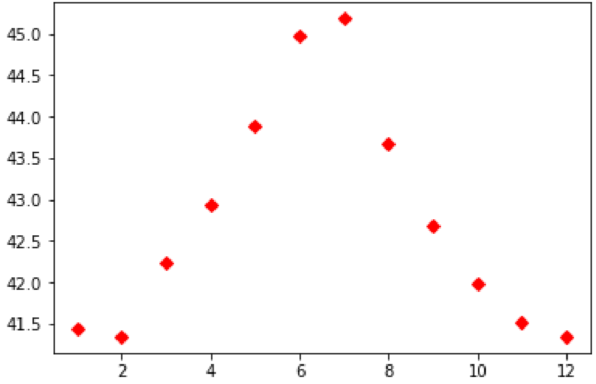
Graph size

Default attributes Matplotlib can control

Color and Style

- Could color, line or style of graph be modified?






```
plt.plot(x, y, 'g--')
```

```
plt.plot(x, y, 'rD')
```

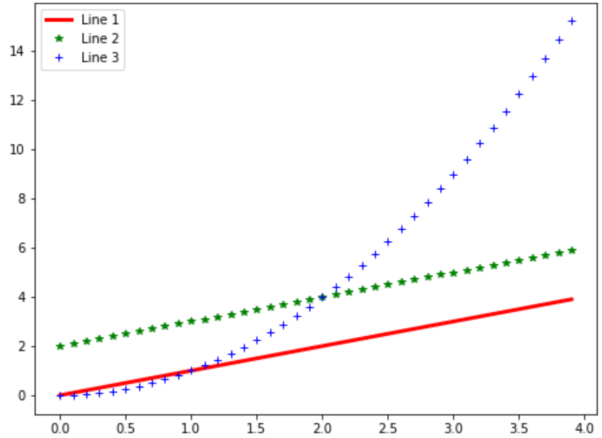
Color and Style

Character	Color
b	blue
g	green
r	red
c	cyan
m	magenta
Y	yellow
k	black
w	white

Type	Description
'-'	solid
'--'	dashed
'-.'	dash_dot
':'	dotted
'None'	draw nothing
' '	draw nothing
''	draw nothing

Mark	Description
"o"	circle
"v"	triangle_down
"s"	square
"p"	pentagon
"*"	star
"h"	hexagon1
"+"	plus
"D"	diamond
…	…

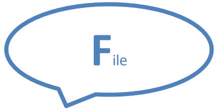
Other Attributes



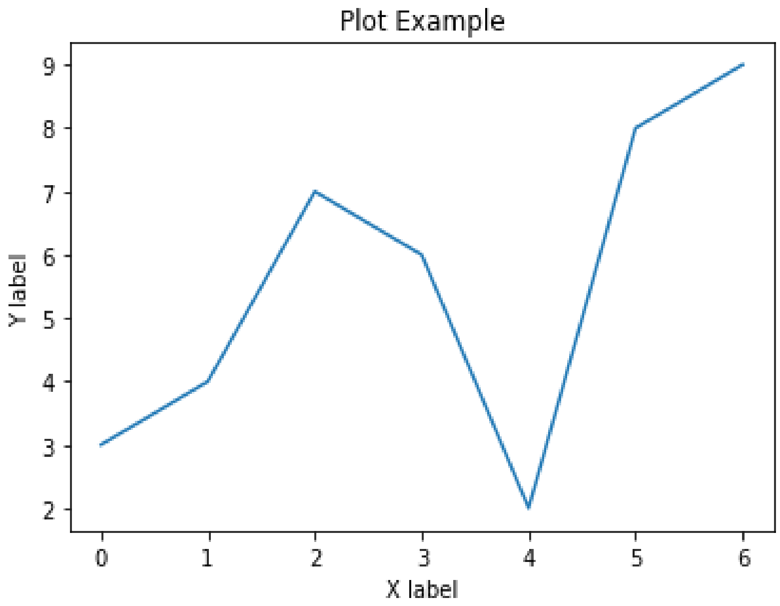

```
# Filename: multilines.py
import matplotlib.pyplot as plt
import numpy as np

plt.figure(figsize = (8, 6), dpi = 100)
t = np.arange(0., 4., 0.1)
plt.plot(t, t, color='red', linestyle='-', linewidth=3, label='Line 1')
plt.plot(t, t+2, color='green', linestyle='', marker='*', linewidth=3, label='Line 2')
plt.plot(t, t**2, color='blue', linestyle='', marker='+', linewidth=3, label='Line 3')
plt.legend(loc = 'upper left')
```

Words

Add titles： graph, vertical axis and horizontal axis

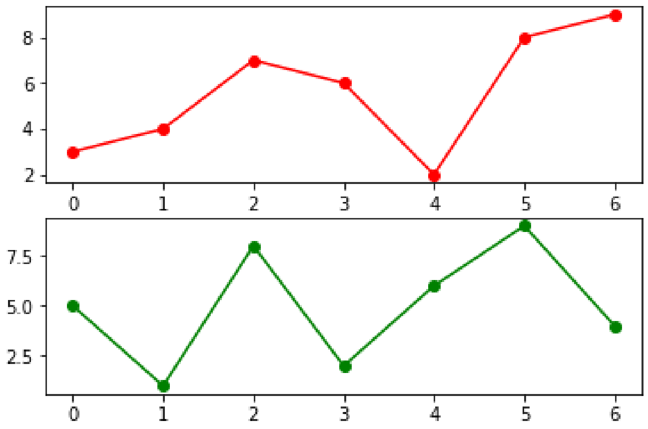


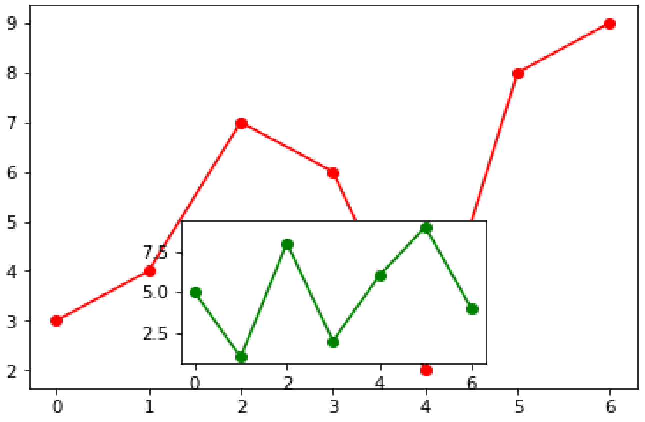

```
# Filename: title.py
import matplotlib.pyplot as plt

plt.title('Plot Example')
plt.xlabel('X label')
plt.ylabel('Y label')
plt.plot(range(7), [3, 4, 7, 6, 2, 8, 9])
```



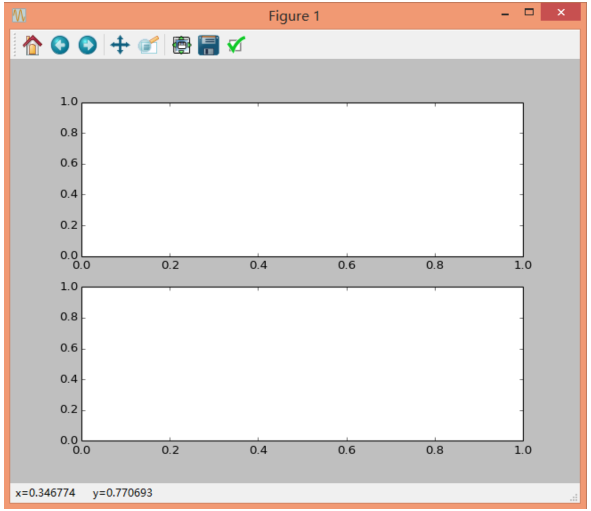
Subplots

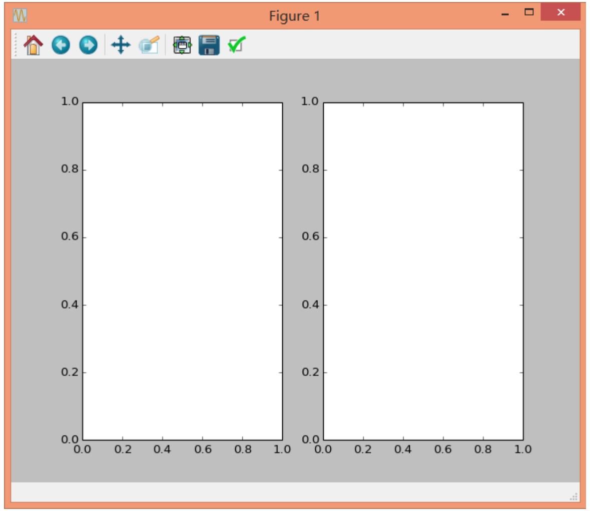
- The plotting is carried out in the current figure and the current coordinate system (axes) in Matplotlib. By default, the plotting is in a figure No. 1. We can plot in multiple areas of a figure.
- Using subplot()/subplots() and axes() functions respectively.

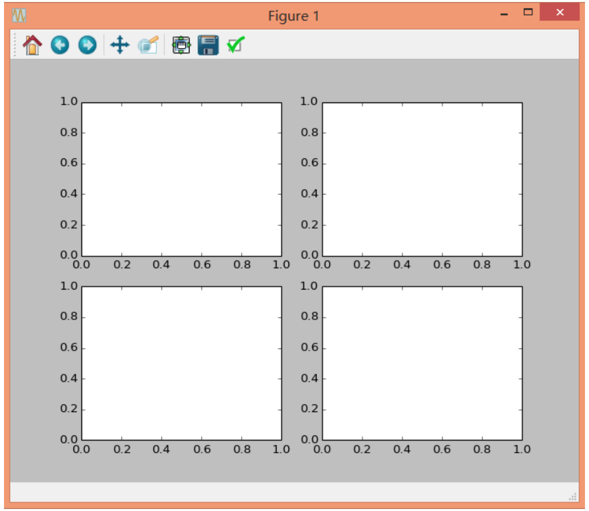




subplots








```
plt.subplot(211)
plt.subplot(212)
```

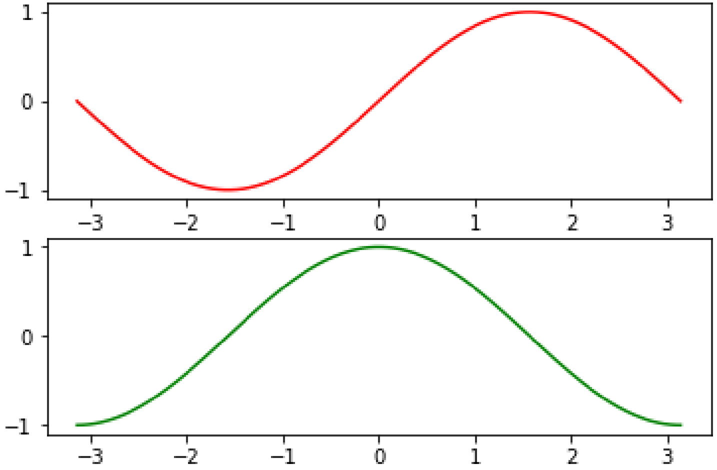
```
plt.subplot(121)
plt.subplot(122)
```

```
plt.subplot(221)
plt.subplot(222)
plt.subplot(223)
plt.subplot(224)
```

subplot()

File

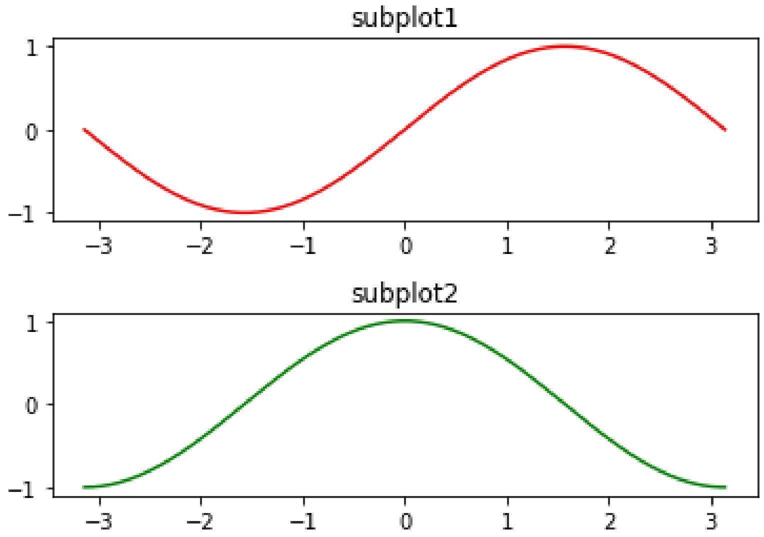
```python
# Filename: subplot.py
import numpy as np
import matplotlib.pyplot as plt

x = np.linspace(-np.pi, np.pi, 300)
plt.figure(1)          # default
plt.subplot(211)       # first subplot
plt.plot(x, np.sin(x), color = 'r')
plt.subplot(212)       # second subplot
plt.plot(x, np.cos(x), color = 'g')
```



subplots()

File

```
# Filename: subplots.py
import numpy as np
import matplotlib.pyplot as plt

x = np.linspace(-np.pi, np.pi, 300)
fig, (ax0, ax1) = plt.subplots(2, 1)
ax0.plot(x, np.sin(x), color = 'r')
ax0.set_title('subplot1')
plt.subplots_adjust(hspace = 0.5)
ax1.plot(x, np.cos(x), color = 'g')
ax1.set_title('subplot2')
```

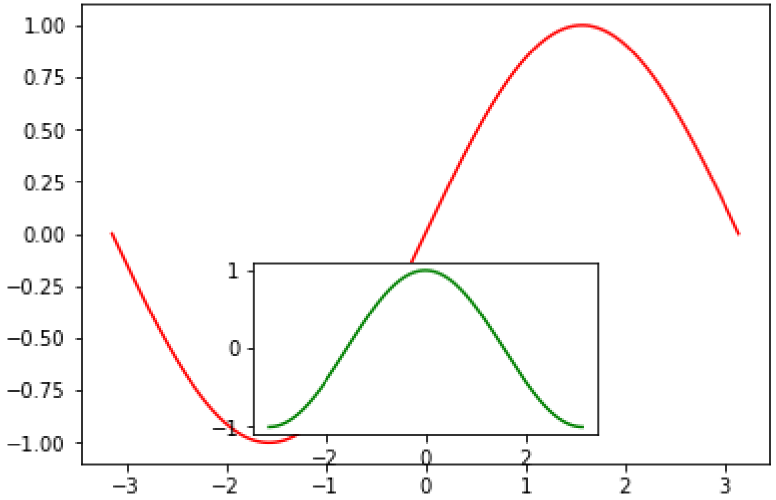


subplots-axes

axes([left,bottom,width,height]) Range of parameter: (0,1)

File

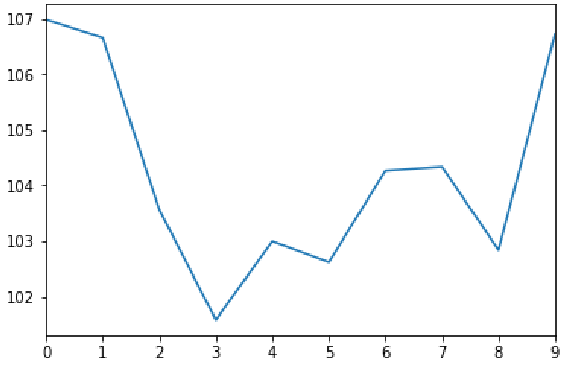
```
# Filename: axes.py
import numpy as np
import matplotlib.pyplot as plt

x = np.linspace(-np.pi, np.pi, 300)
plt.axes([.1, .1, 0.8, 0.8])
plt.plot(x, np.sin(x), color = 'r')
plt.axes([.3, .15, 0.4, 0.3])
plt.plot(x, np.cos(x), color = 'g')
```



pandas plotting

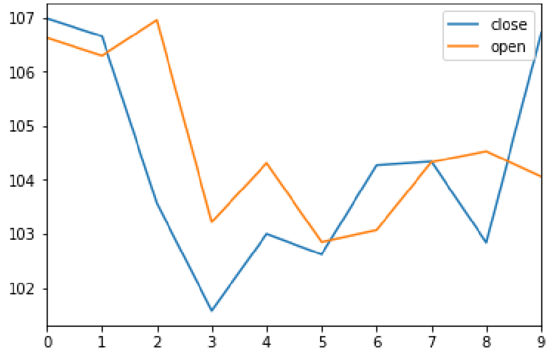
Source

```
>>> quotesdf.loc[:9, 'close'].plot()
```



Source

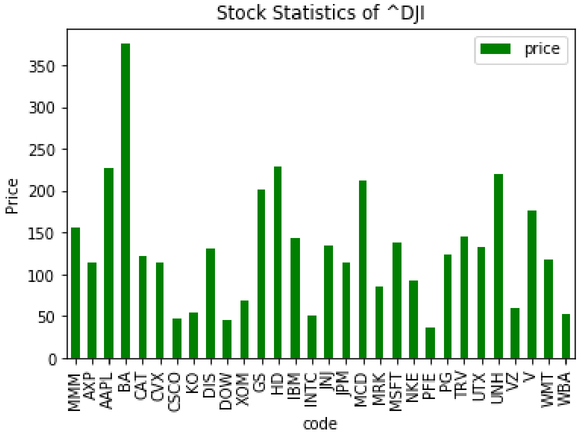
```
>>> quotesdf.loc[:9, ['close', 'open']].plot()
```



pandas plotting

Source

```
>>> ax = djidf.plot(kind = 'bar', x = 'code', y = 'price', color = 'g');
ax.set(ylabel='Price', title = 'Stock Statistics of ^DJI')
```



Data Processing Using Python

3 DATA CLEAN OF DATA EXPLORATION AND PREPROCESSING

Data Exploration

- check data errors
- understand data distribution characteristics and inherent regularities

Data preprocessing

- Data cleaning
- Data integration
- Data transformation
- Data reduction

Missing Value Handling

How to deal with?

- drop
- fill

fill	fixed value
	mean, median/mode value
	up and down data
	interpolation function
	most likely value

Missing value handling—DataFrame


```
quotesdf_nan = pd.read_csv('AXP_NaN.csv', index_col = 'Date')
```

judge missing value: df.isnull()

drop missing value: df.dropna()

fill missing value: df.fillna()

How to fill missing value with mean value?

```
quotesdf_nan.fillna(method='ffill', inplace = True)
```

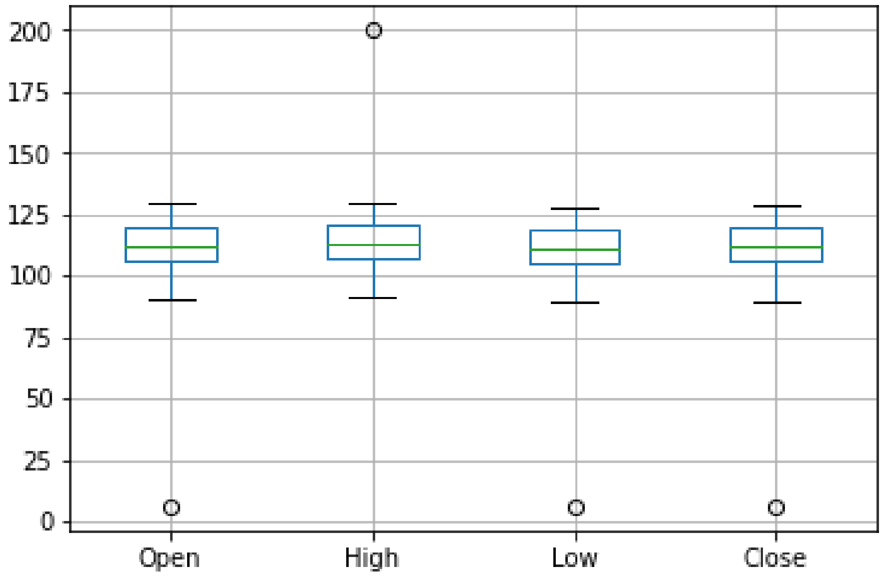
Outliers

How to observe?

- simple statistics
- plotting
- density-based, knn or cluster algorithm

How to deal with?

- same as missing values
- calculate the local mean (binning)
- do nothing



Data Processing Using Python

4 DATA TRANSFORMATION OF DATA PRECESSING

Data Transformation

transform data into the suitable form

common way	Normalization
	Discretization of continuous features
	Binarization

Normalization

What impacts are solved?

- different dimension
- wide range of values

common method

- Min-Max normalization
- Z-Score normalization
- Normalization by decimal scaling

Boston Housing Datasets


```
>>> boston = datasets.load_boston()
>>> boston.feature_names
array(['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD',
     'TAX', 'PTRATIO', 'B', 'LSTAT'], dtype='<U7')
>>> boston.target
array([24. , 21.6, 34.7, 33.4, 36.2, 28.7, 22.9, 27.1, 16.5, ..., ]])
>>> boston_df = pd.DataFrame(boston.data[:, 4:7])
>>> boston_df.columns = boston.feature_names[4:7]
>>> boston_df
       NOX     RM    AGE
0      0.538  6.575  65.2
1      0.469  6.421  78.9
2      0.469  7.185  61.1
...
504  0.573  6.794  89.3
505  0.573  6.030  80.8
```

4: NOX - nitric oxides concentration (parts per 10 million)
5: RM - average number of rooms per dwelling
6: AGE - proportion of owner-occupied units built prior to 1940
MEDV - Median value of owner-occupied homes in $1000's

Min-Max normalization

$$x' = \frac{x - min}{\max - min}$$

```
(df-df.min())/(df.max()-df.min())
```

	NOX	RM	AGE
0	0.538	6.575	65.2
1	0.469	6.421	78.9
2	0.469	7.185	61.1
3	0.458	6.998	45.8
4	0.458	7.147	54.2
5	0.458	6.430	58.7

Problems:

- If the number in the future exceeds min and max one, it needs to be redefined.
- If a certain number is large, the normalized values are close and all are close to 0.

	NOX	RM	AGE
0	0.314815	0.577505	0.641607
1	0.172840	0.547998	0.782698
2	0.172840	0.694386	0.599382
3	0.150206	0.658555	0.441813
4	0.150206	0.687105	0.528321
5	0.150206	0.549722	0.574665

Min-Max normalization

```
from sklearn import preprocessing

min_max_scaler = preprocessing.minmax_scale(df)    # [0,1]
```

Z-Score normalization

$$x' = \frac{x - \bar{x}}{\sigma}$$

```
(df-df.mean())/df.std()
```

	NOX	RM	AGE
0	0.538	6.575	65.2
1	0.469	6.421	78.9
2	0.469	7.185	61.1
3	0.458	6.998	45.8
4	0.458	7.147	54.2
5	0.458	6.430	58.7

	NOX	RM	AGE
0	-0.144075	0.413263	-0.119895
1	-0.739530	0.194082	0.366803
2	-0.739530	1.281446	-0.265549
3	-0.834458	1.015298	-0.809088
4	-0.834458	1.227362	-0.510674
5	-0.834458	0.206892	-0.350810

Features:

- Most frequently used.
- The mean of the processed data is 0, and the standard deviation is 1.

Z-Score normalization


```
scaler = preprocessing.scale(df)
```

```
array([[-0.14421743,  0.41367189, -0.12001342],
       [-0.74026221,  0.19427445,  0.36716642],
       [-0.74026221,  1.28271368, -0.26581176],
       ...,
       [ 0.15812412,  0.98496002,  0.79744934],
       [ 0.15812412,  0.72567214,  0.73699637],
       [ 0.15812412, -0.36276709,  0.43473151]])
```

Normalization by decimal scaling

$$x' = \frac{x}{10^{j}}$$

```
df/10**np.ceil(np.log10(df.abs().max()))
```

Features:

- Move the decimal point position. The number of moves depends on the maximum value of the features' absolute value.
- Fall between [- 1, 1] commonly.

	NOX	RM	AGE
0	0.538	6.575	65.2
1	0.469	6.421	78.9
2	0.469	7.185	61.1
3	0.458	6.998	45.8
4	0.458	7.147	54.2
5	0.458	6.430	58.7

	NOX	RM	AGE
0	0.538	0.6575	0.652
1	0.469	0.6421	0.789
2	0.469	0.7185	0.611
3	0.458	0.6998	0.458
4	0.458	0.7147	0.542
5	0.458	0.6430	0.587

Discretization of Continuous Features

Method

- Binning: equal-width, equal frequency
- Clustering

```
pd.cut(df.AGE, 5, labels = range(5))
pd.qcut(df.AGE, 5, labels = range(5))
```

0	65.2
1	78.9
2	61.1
3	45.8
4	54.2
5	58.7

0	3
1	3
2	2
3	2
4	2
5	2

0	1
1	2
2	1
3	1
4	1
5	1

Feature Binarization

rating	Label
6	1
4	0
7	1
7	1
8	1
5	0
3	0
3	0
9	1
4	0
6	1
7	1
5	0
9	1
9	1
8	1
2	0
5	0
3	0
3	0

Source

```
>>> from sklearn.preprocessing import Binarizer
>>> X = boston.target.reshape(-1,1)
>>> Binarizer(threshold = 20.0).fit_transform(X)
```

Data Processing Using Python

5 DATA REDUCTION OF DATA PREPROCCESSING

Data Reduction

Purpose:

- The features and values are normalized to obtain a much smaller specification representation than the original dataset, but still close to the integrity of the original data. Mining on the dataset after the specification can produce almost the same analysis results.

Way	Feature reduction: forward selection, backward elimination, decision tree, PCA
	Value reduction: Parametric method (regression, log linear model), nonparametric method(histogram, clustering, sampling)

Feature Reduction - PCA

Source

```
>>> from sklearn.decomposition import PCA
>>> X = preprocessing.scale(boston.data)
>>> pca = PCA(n_components=5)
>>> pca.fit(X)
>>> pca.explained_variance_ratio_
array([0.47129606, 0.11025193, 0.0955859 , 0.06596732, 0.06421661])
```

Value Reduction - histogram

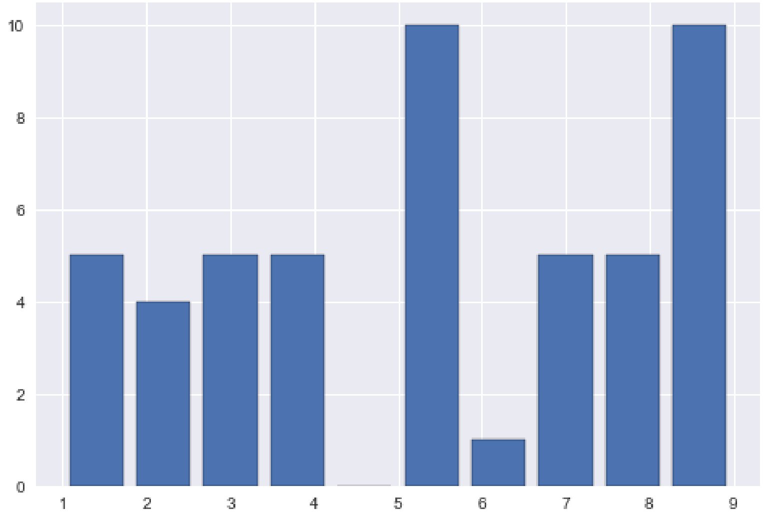
Features:

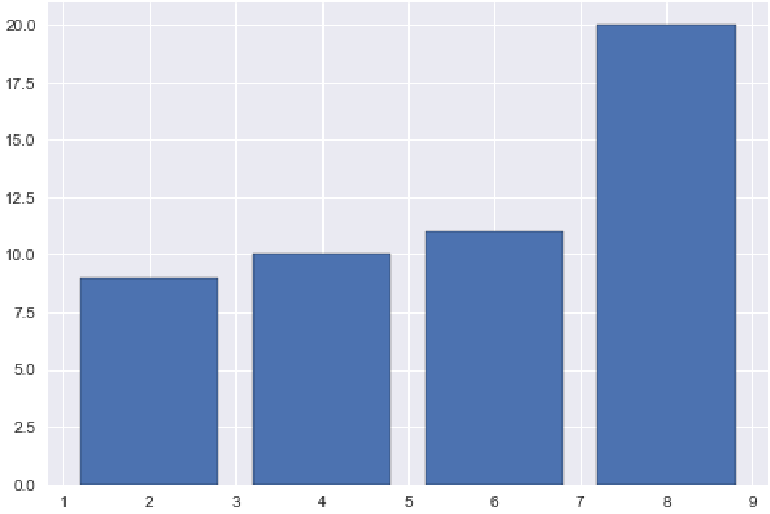
- Show data distribution by forming bins.
- Each bin shows the frequency of data value.

```
array([4, 8, 9, 8, 7, 2, 8, 7, 5, 3, 1, 4, 5, 8, 7, 9, 5, 9, 9, 5, 9, 1, 9, 7, 1, 2, 9, 5, 5,
5, 9, 4, 3, 5, 5, 4, 7, 4, 9, 8, 2, 6, 3, 5, 3, 2, 9, 1, 3, 1])
```

```
data = np.random.randint(1,10,50)
```

Value Reduction - histogram






```
array([4, 8, 9, 8, 7, 2, 8, 7, 5, 3, 1, 4, 5, 8, 7, 9, 5, 9, 9, 5, 9, 1, 9, 7, 1, 2, 9, 5, 5, 5, 9,
4, 3, 5, 5, 4, 7, 4, 9, 8, 2, 6, 3, 5, 3, 2, 9, 1, 3, 1])
```

plt.hist(data, bins=…)

Value Reduction - sampling

<table>
<tr><td rowspan="3">Sampling</td><td>Random Sampling:
without replacement
with replacement</td></tr>
<tr><td>Cluster sampling</td></tr>
<tr><td>Stratified Sampling</td></tr>
</table>

Some features:

- Without replacement sampling: Take n samples from N samples of the original dataset D, and get different data each time.
- With replacement sampling: Take n samples from the N samples in the original dataset D, record them and put them back. It is possible to extract the same data.
- Stratified sampling: Dataset D is divided into disjoint parts(layers), and each layer is randomly sampled to get the final result.

Random Sampling

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
24	4.8	3.4	1.9	0.2
81	5.5	2.4	3.7	1.0
145	6.7	3.0	5.2	2.3
115	6.4	3.2	5.3	2.3
136	6.3	3.4	5.6	2.4
98	5.1	2.5	3.0	1.1
79	5.7	2.6	3.5	1.0
143	6.8	3.2	5.9	2.3
87	6.3	2.3	4.4	1.3
107	7.3	2.9	6.3	1.8

Without Replacement:

iris_df.sample(n = 10)

iris_df.sample(frac = 0.3)

With Replacement:

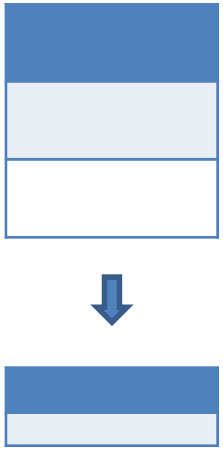
iris_df.sample(n = 10, replace = True)

iris_df.sample(frac = 0.3, replace = True)

Stratified Sampling

Source

```
>>> A = iris_df[iris_df.target == 0].sample(frac = 0.3)
>>> B = iris_df[iris_df.target == 1].sample(frac = 0.2)
>>> A.append(B)
```



Summary

Data Processing Using Python Week 5 Data retrieval and preprocessing of Python

- **5.1 Convenient and Fast Data Acquisition**
  - download data directly, get data using API, datasets and corpus in Python
- **5.2 Fundamentals of Python Plotting**
  - Matplotlib plotting and set attributes, pandas plotting
- **5.3 Data Clean of Data Exploration and Preprocessing**
  - missing value detection and processing, outliers detection and processing
- **5.4 Data Transformation of Data Precessing**
  - normalization, discretization of continuous features, feature binarization
- **5.5 Data Reduction of Data Preproccessing**
  - feature reduction of PCA, value reduction of histogram and sampling