Chapter 05 Task 01

Introducing matplotlib

```
fb = pd.read_csv(
    'data/fb_stock_prices_2018.csv', index_col='date', parse_dates=True
)

plt.plot(fb.index, fb.open)     # plot open price against index (date)
plt.show()                      # display the graph

# add this to avoid having to do plt.show() in the future
%matplotlib inline


plt.plot('high', 'low', 'or', data=fb.head(20))     # red scatter plot

quakes = pd.read_csv('data/earthquakes.csv')
plt.hist(quakes.query('magType == "ml"').mag)       # histogram using magtype ml
```

Marker	Linestyle	Color	Format String	Result
	-	b	-b	blue solid line
.		k	.k	black points
	--	r	--r	red dashed line
o	-	g	o-g	green solid line with circles
	:	m	:m	magenta dotted line
x	-.	c	x-.c	cyan dot-dashed line with x's

```
# multiple plots
fig, axes = plt.subplots(1, 2)
# pick size
fig, axes = plt.subplots(1, 2, figsize=(10, 4))

# can add plots in plots using figure and axes
fig = plt.figure(figsize=(3, 3))
outside = fig.add_axes([0.1, 0.1, 0.9, 0.9])
inside = fig.add_axes([0.7, 0.7, 0.25, 0.25])

# plots can be saved as images
fig.savefig('empty.png')
# exit properly
plt.close('all')

# udpate default settings on plot size
mpl.rcParams['figure.figsize'] = (300, 10)
```

Plotting with pandas

```
fb.plot(
    kind='line',         # line plot
    y='open',            # y = open
    figsize=(10, 5),     # 10in x 5in
    style='-b',          # blue line
    # color='blue',
    # linestyle='solid',
    legend=False,        # no legend
    title='Evolution of Facebook Open Price'
)

fb.first('1W').plot(
    y=['open', 'high', 'low', 'close'],     # each line aligns with style below
    style=['o-b', '--r', ':k', '.-g'],
    title='Facebook OHLC Prices during 1st Week of Trading 2018'
).autoscale()

# not providing a column has it use all of them
fb.plot(
    kind='line',
    subplots=True,
    layout=(3, 2),
    figsize=(15, 10),
    title='Facebook Stock 2018'
)
```

```
fig, axes = plt.subplots(1, 3, figsize=(15, 5))
# grouping the subplots based on index can help display data better
new_cases_rolling_average[['China']].plot(ax=axes[0], style='-.c')
new_cases_rolling_average[['Italy', 'Spain']].plot(
    ax=axes[1], style=['-', '--'],
    title='7-day rolling average of new COVID-19 cases\n(source: ECDC)'
)
# line styles help with b&w graphs
new_cases_rolling_average[['Brazil', 'India', 'USA']]\
    .plot(ax=axes[2], style=['--', ':', '-'])
```

log transform on the x-axis since the scales of the axes are very different. With `pandas`, we simply pass in `logx=True`

```
fb.assign(
    max_abs_change=fb.high - fb.low
).plot(
    kind='scatter', x='volume', y='max_abs_change',
    title='Facebook Daily High - Low vs. Volume Traded'
```

```
    logx=True, alpha=0.25
    # logx   = change scale
    # alplha = transparanecy
)


# histogram in pandas
fb.volume.plot(
    kind='hist',
    title='Histogram of Daily Volume Traded in Facebook Stock'
)
plt.xlabel('Volume traded') # label the x-axis (discussed in chapter 6)

# kernel density estimation on top of histogram
ax = fb.high.plot(kind='hist', density=True, alpha=0.5)
fb.high.plot(
    ax=ax, kind='kde', color='blue',
    title='Distribution of Facebook Stock\'s Daily High Price in 2018'
)
plt.xlabel('Price ($)') # label the x-axis (discussed in chapter 6)


# box plots
fb.iloc[:,:4].plot(kind='box', title='Facebook OHLC Prices Box Plot')
plt.ylabel('price ($)') # label the y-axis (discussed in chapter 6)

# bar chart
quakes.parsed_place.value_counts().iloc[14::-1,].plot(
# quakes.groupby('parsed_place').tsunami.sum().sort_values().iloc[-10:,].plot(
    kind='barh', figsize=(10, 5),
    title='Top 15 Places for Earthquakes '
          '(September 18, 2018 - October 13, 2018)'
)
plt.xlabel('earthquakes') # label the x-axis (discussed in chapter 6)
```

Pandas plotting module

```
# scatter plots between all columns
from pandas.plotting import scatter_matrix
scatter_matrix(fb, figsize=(10, 10))
# default diagonal is histogram
scatter_matrix(fb, figsize=(10, 10), diagonal='kde')

# see how the variable correlates with past observations of itself
from pandas.plotting import lag_plot
np.random.seed(0) # make this repeatable
```

```
lag_plot(pd.Series(np.random.random(size=200)))
# Data with some level of correlation to itself (autocorrelation) may have patterns
lag_plot(fb.close)

# bootstrap plot helps understand the uncertainty in our summary statistics
from pandas.plotting import bootstrap_plot
fig = bootstrap_plot(fb.volume, fig=plt.figure(figsize=(10, 6)))
```