PYTHON FOR FINANCE

Analyze Big Financial Data

Introduction

- How information influences the movement of stock
- Building software to analyze & visualize the relationships
- Details about how stock exchange works
- Learn how to build machine learning algorithms

Three parts to the course

- Manipulating Financial Data in Python
  - How read data into Python and manipulate using power statistical algorithms
- Computational Investing
  - Algorithms, methods & models used by hedge funds and IBs to manipulate and work with financial data
- Learning Algorithms for Trading
  - We pull everything together, the data and use it with machine learning algorithm like Q-Learning and Random Forest.

Goal

- After completion of the ‘Series’ of workshops:
  - You’ll be equipped to join trading system development team.
    - Hedge fund or Investment Bank
  - Begin equity trading, and earn some money !
  - Build up your own portfolio of stocks and other financial instruments.

Textbooks

- Python for Finance by Hilpish
- What Hedge Funds Really Do by Philip J. Romero
- Machine Learning by Mitchell

Python for Financial Applications:

- Set up quick prototype algorithms
- Computational speed
- Features:
  - Strong scientific libraries
  - Strongly maintained
  - Fast
    - If you stick to metrics notation, because lower levels are returned in C.
- Alternatives: R and MATLAB

Prerequisites

- You need to have strong programming skills
- Basic knowledge of finance
- Statistics/Econometrics knowledge

Prepare for the challenge !

READING & PLOTTING OF STOCK DATA

Lesson outline

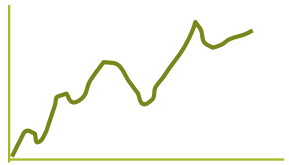
- In this lesson you will learn how to read data, select subsets of it and generate useful plots, using **pandas** and **matplotlib**.
- Read stock data from CSV files:
  - **pandas.DataFrame**
  - **pandas.read_csv**
- Select desired rows and columns:
  - **Indexing and Slicing Data**
  - Gotchas: **Label-based slicing conventions**
- Visualize data by generating plots:
  - **Plotting**
  - **pandas.DataFrame.plot**
  - **matplotlib.pyplot.plot**

What CSV file looks

```
Date,Open,High,Low,Close,Adj Close,
2006-11-30,5486.000000,5684.399902,5411.000000,5669.899902,5669.899902,0
2006-12-31,5672.700195,5829.299805,5499.000000,5773.399902,5773.399902,0
2007-01-31,5781.299805,6052.100098,5781.299805,5832.500000,5832.500000,0
2007-02-28,5840.600098,5998.899902,5641.100098,5995.000000,5995.000000,0
2007-03-31,5994.000000,6265.200195,5915.799805,6166.000000,6166.000000,0
2007-04-30,6163.200195,6384.100098,6134.200195,6313.500000,6313.500000,0
2007-05-31,6321.000000,6409.200195,6170.200195,6274.899902,6274.899902,0
2007-06-30,6282.000000,6436.700195,6044.299805,6144.200195,6144.200195,0
2007-07-31,6138.600098,6247.200195,5483.299805,6247.200195,6247.200195,0
2007-08-31,6253.700195,6594.399902,6146.500000,6567.799805,6567.799805,0
2007-09-30,6567.600098,6800.200195,6542.899902,6754.100098,6754.100098,0
2007-10-31,6778.399902,6851.500000,6312.600098,6533.100098,6533.100098,0
2007-11-30,6535.200195,6684.399902,6105.600098,6339.799805,6339.799805,0
2007-12-31,6336.600098,6385.700195,5186.799805,5650.299805,5650.299805,0
2008-01-31,5672.600098,6022.000000,5490.399902,5572.100098,5572.100098,0
2008-02-29,5534.299805,5535.200195,5039.600098,5355.700195,5355.700195,0
2008-03-31,5363.399902,5672.700195,5301.000000,5595.399902,5595.399902,0
2008-04-30,5592.600098,5980.799805,5538.700195,5654.700195,5654.700195,0
2008-05-31,5671.899902,5681.500000,5144.700195,5215.299805,5215.299805,0
```

Header

Data



Quiz:

- Which fields you would expect to see in a CSV file of stock data?

❑# of employees

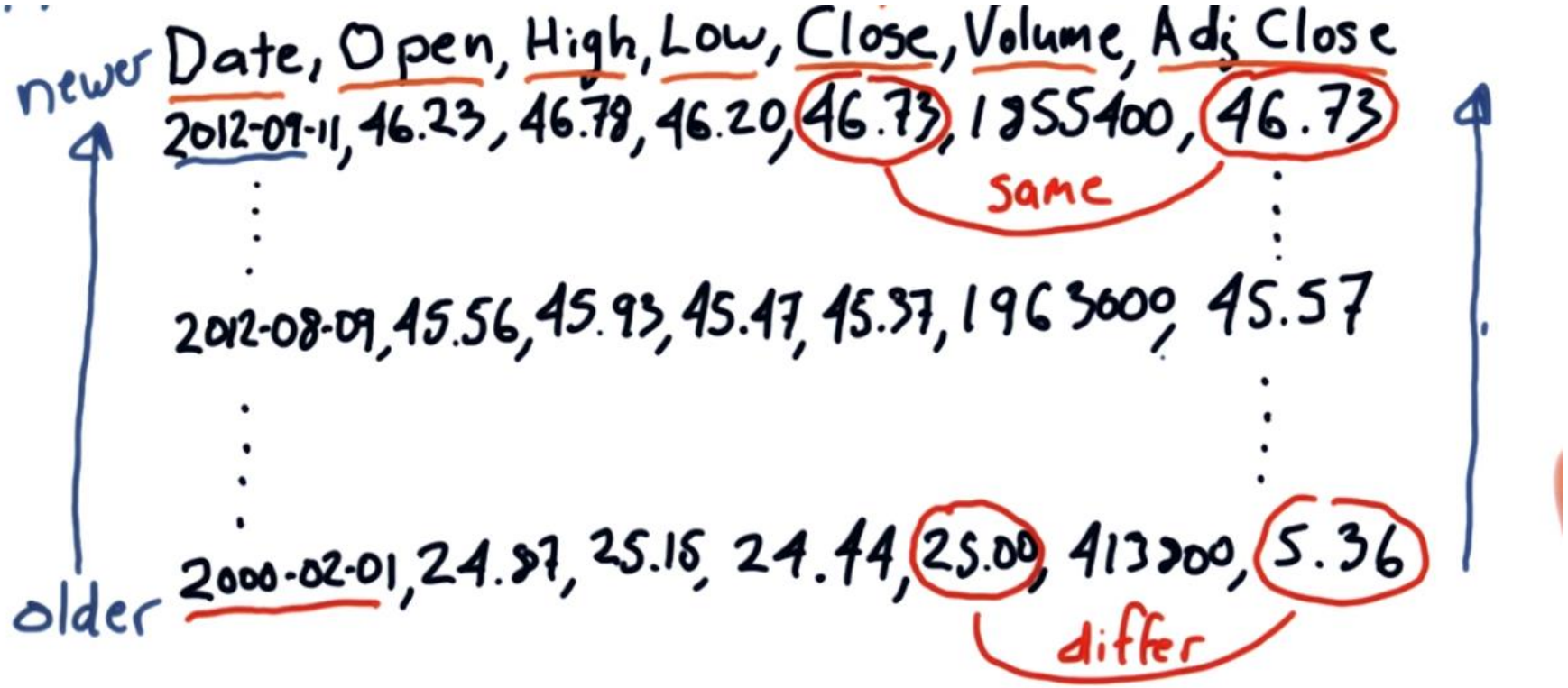
❑Date/time

❑Company name

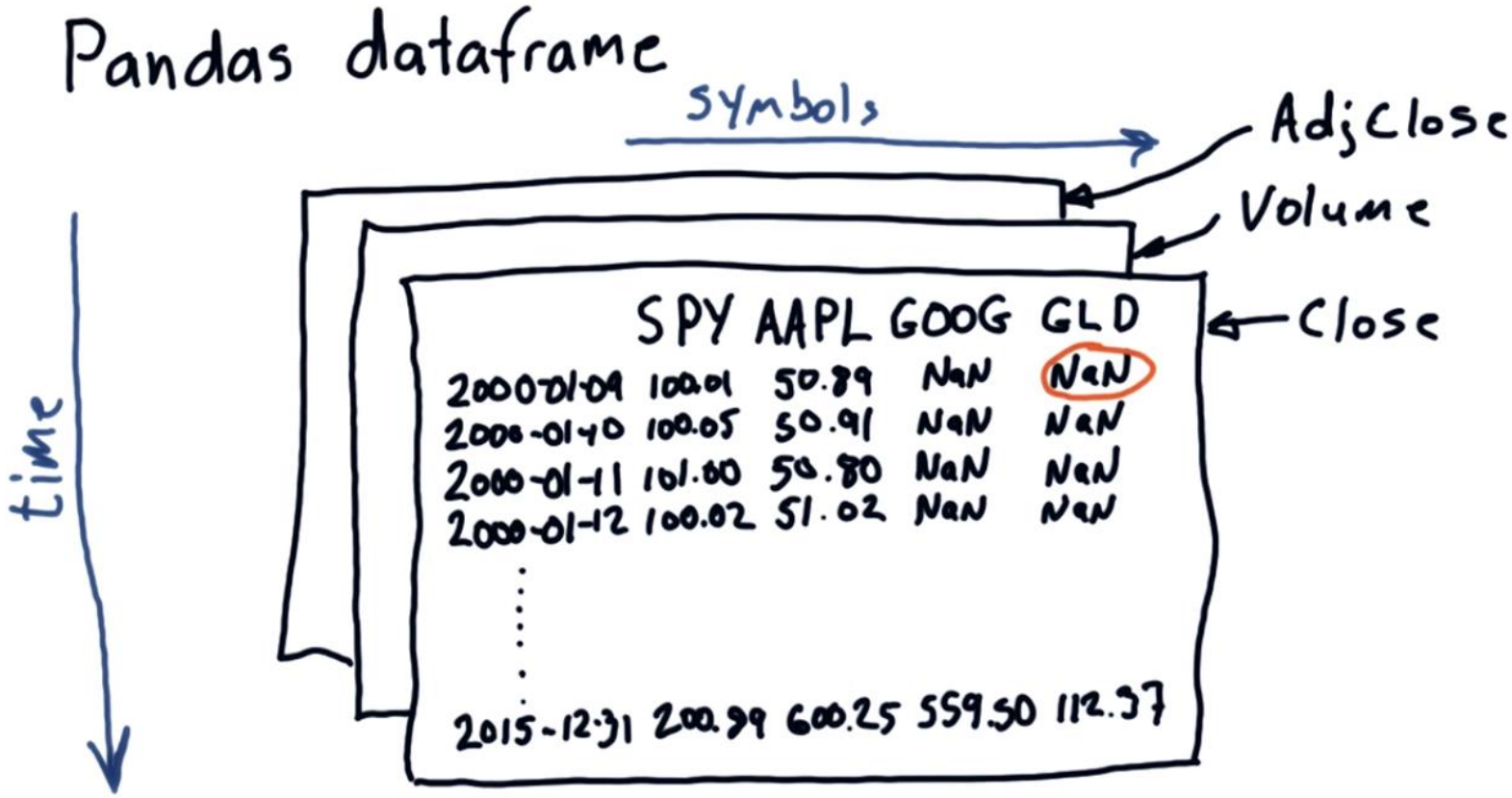
❑Price of the stock

❑Company's hometown

Yahoo Database



Pandas DataFrame



Read CSV file

```
1  import pandas as pd
2
3  def test_run():
4      df=pd.read_csv("C:\Users\Fahad\Desktop\Data\AAPL.csv")
5      print df
6      print df.head()
7
8      print df[0:5]
9
10 if __name__=="__main__":
11     test_run()
```

Quiz: Print last 5 rows of the data frame

```
1   import pandas as pd
2
3
4   def test_run():
5       """Function called by Test Run."""
6       df = pd.read_csv("data/AAPL.csv")
7       # TODO: Print last 5 rows of the data frame
8
9
10  if __name__ == "__main__":
11      test_run()
12
```

Slicing

```
1  import pandas as pd
2
3
4  def test_run():
5      df = pd.read_csv("data/AAPL.csv")
6      print df[10:21]#rows between index 10 and 20
7
8
9  if __name__ == "__main__":
10     test_run()
```

Computing max closing price

```
1  import pandas as pd
2
3  def get_max_close(symbol):
4      """Return the maximum closing value for stock indicated by symbol.
5
6      Note: Data for a stock is stored in file: data/<symbol>.csv
7      """
8      df = pd.read_csv("data/{}.csv".format(symbol))  # read in data
9      return df['Close'].max()  # compute and return max
10
11
12 def test_run():
13     """Function called by Test Run."""
14     for symbol in ['AAPL', 'IBM']:
15         print "Max close"
16         print symbol, get_max_close(symbol)
17
18
19
20 if __name__ == "__main__":  # if run standalone
21     test_run()
```

Quiz: Compute mean volume

```
"""Compute mean volume"""

import pandas as pd

def get_mean_volume(symbol):
    """Return the mean volume for stock indicated by symbol.

    Note: Data for a stock is stored in file: data/<symbol>.csv
    """
    df = pd.read_csv("data/{}.csv".format(symbol))  # read in data
    # TODO: Compute and return the mean volume for this stock


def test_run():
    """Function called by Test Run."""
    for symbol in ['AAPL', 'IBM']:
        print "Mean Volume"
        print symbol, get_mean_volume(symbol)


if __name__ == "__main__":
    test_run()
```

Plotting stock price data

```
1  import pandas as pd
2  import matplotlib.pyplot as plt
3
4  def test_run():
5      df = pd.read_csv("data/AAPL.csv")
6      print df['Adj Close']
7      df['Adj Close'].plot()
8      plt.show()  # must be called to show plots
9
10
11 if __name__ == "__main__":
12     test_run()
```

Quiz: Plot High prices for IBM

```
"""Plot High prices for IBM"""

import pandas as pd
import matplotlib.pyplot as plt

def test_run():
    df = pd.read_csv("data/IBM.csv")
    # TODO: Your code here
    plt.show()  # must be called to show plots


if __name__ == "__main__":
    test_run()

```

Plot two columns

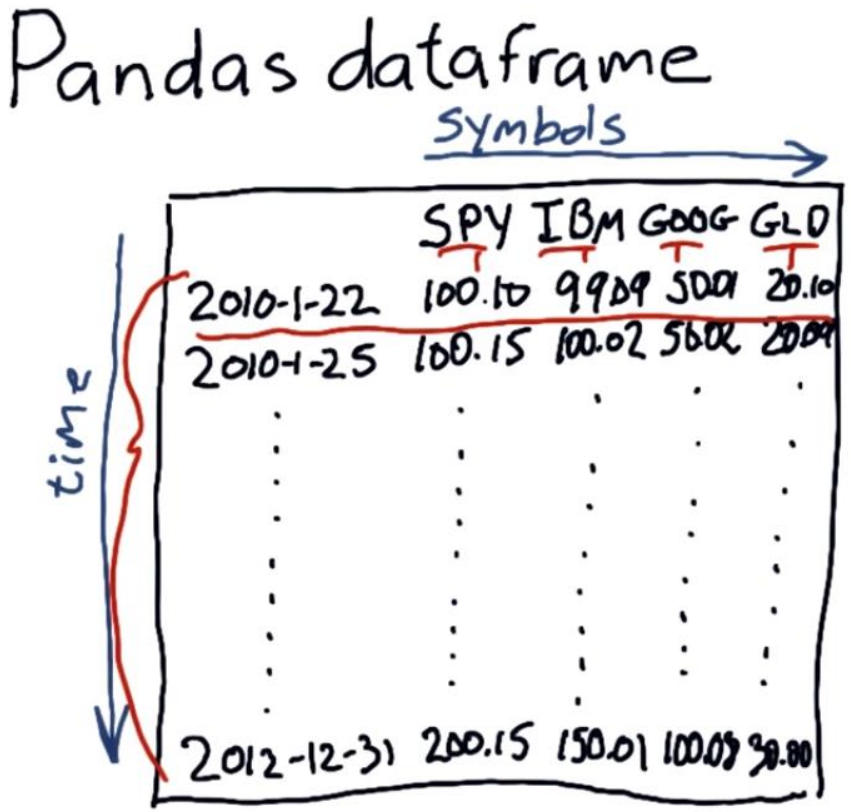
```
1  import pandas as pd
2  import matplotlib.pyplot as plt
3
4  def test_run():
5      df = pd.read_csv("data/AAPl.csv")
6      df[['Close','Adj Close']].plot()
7      plt.show()  # must be called to show plots
8
9
10 if __name__ == "__main__":
11     test_run()
```

WORKING WITH MULTIPLE STOCKS

Lesson Outline

- Here's an overview of what you'll learn to do in this lesson. Documentation links are for reference.
- **Read in multiple stocks:**
- Create an empty **pandas.DataFrame** with dates as index: **pandas.date_range**
- Drop missing date rows: **pandas.DataFrame.dropna**
- Incrementally join data for each stock: **pandas.DataFrame.join**
- **Manipulate stock data:**
- **Index and select data** by row (dates) and column (symbols)
- Plot multiple stocks at once (still using **pandas.DataFrame.plot**)
- Carry out arithmetic operations across stocks

Problems to solve



Problems to solve

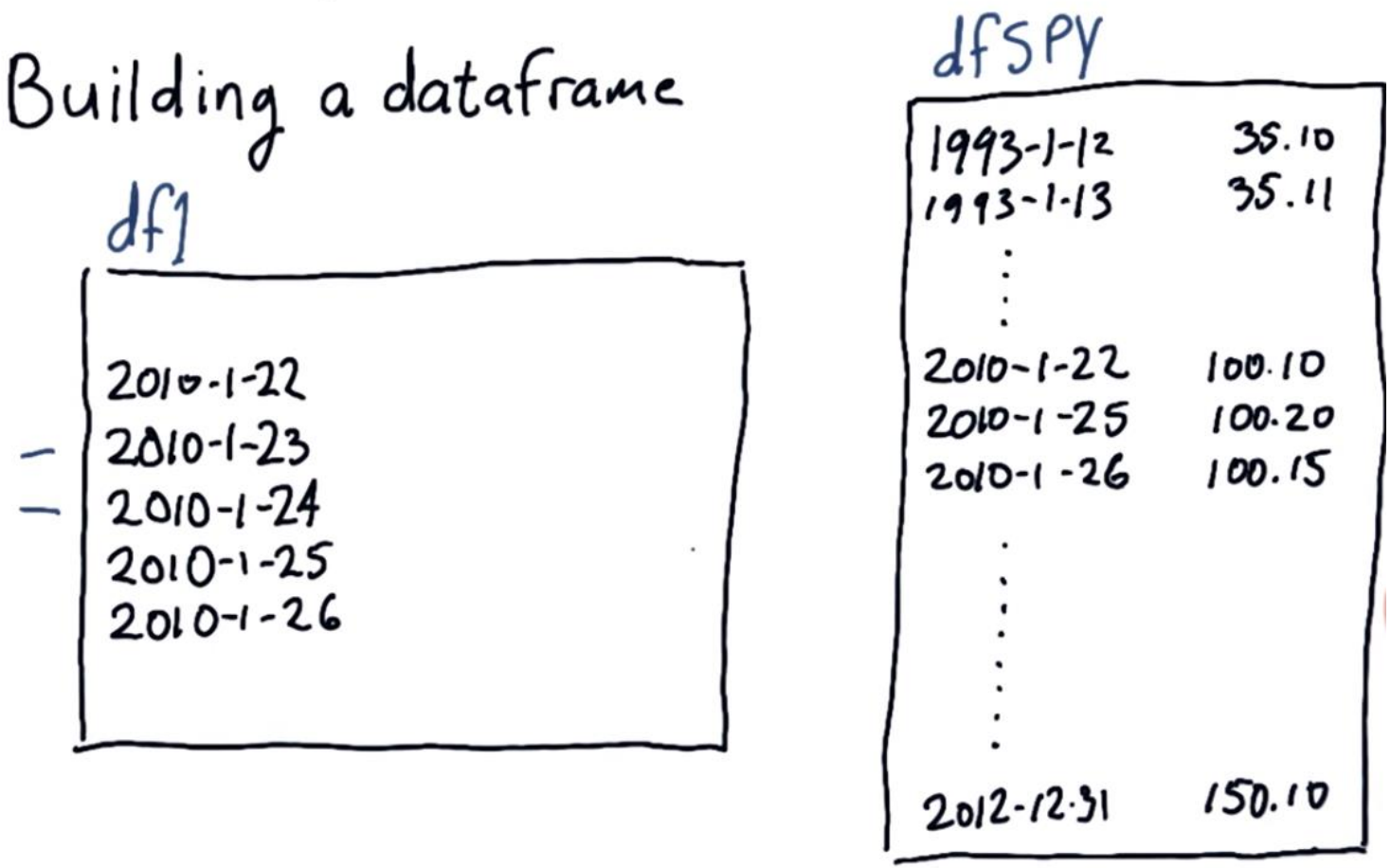
- Date ranges
- Mutiple stocks
- Align dates
- Proper date order

Quiz: NYSE trading days

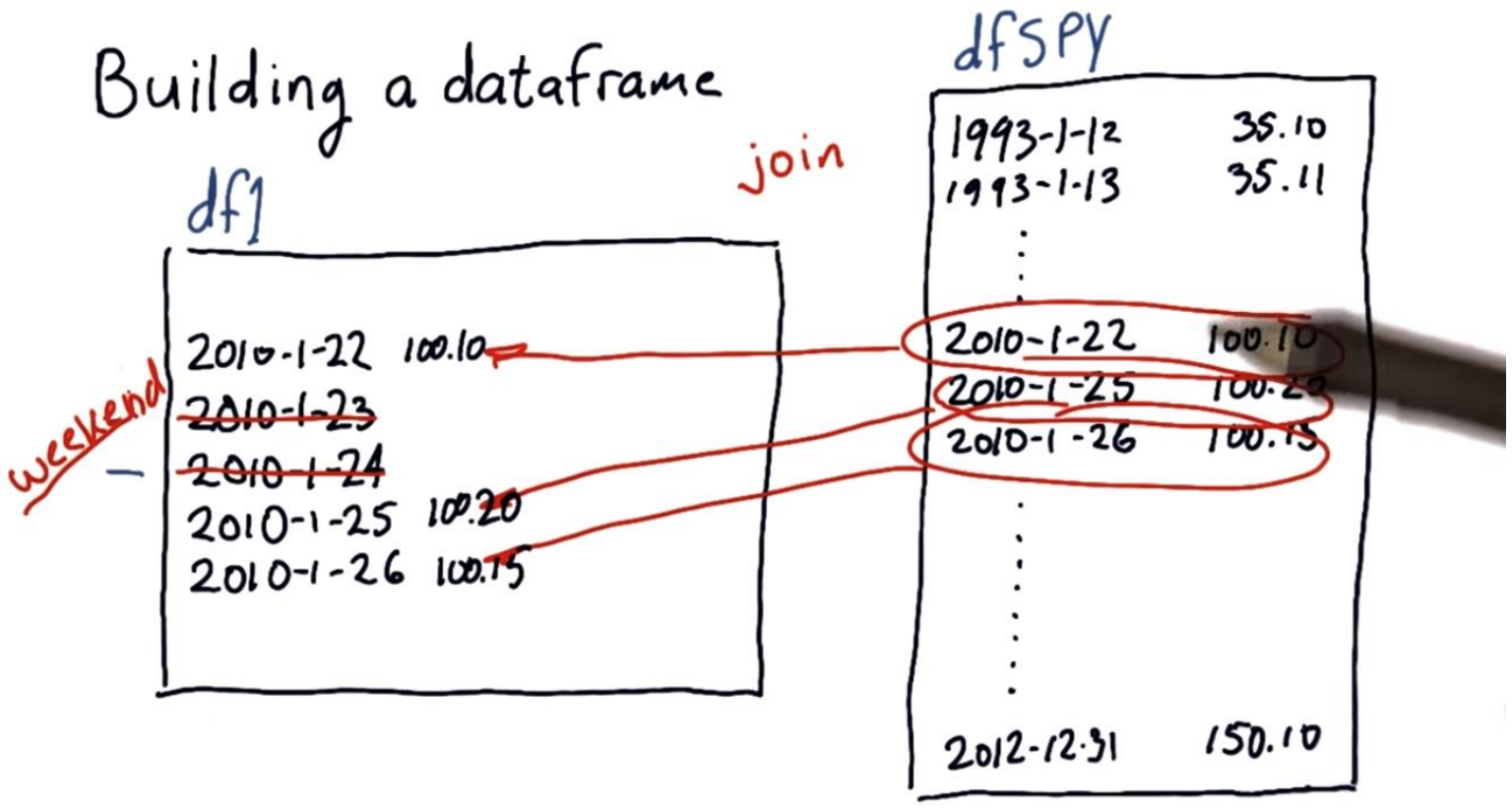
Quiz: How many days were US stocks traded at NYSE in 2014?

- 365
- 260
- 252

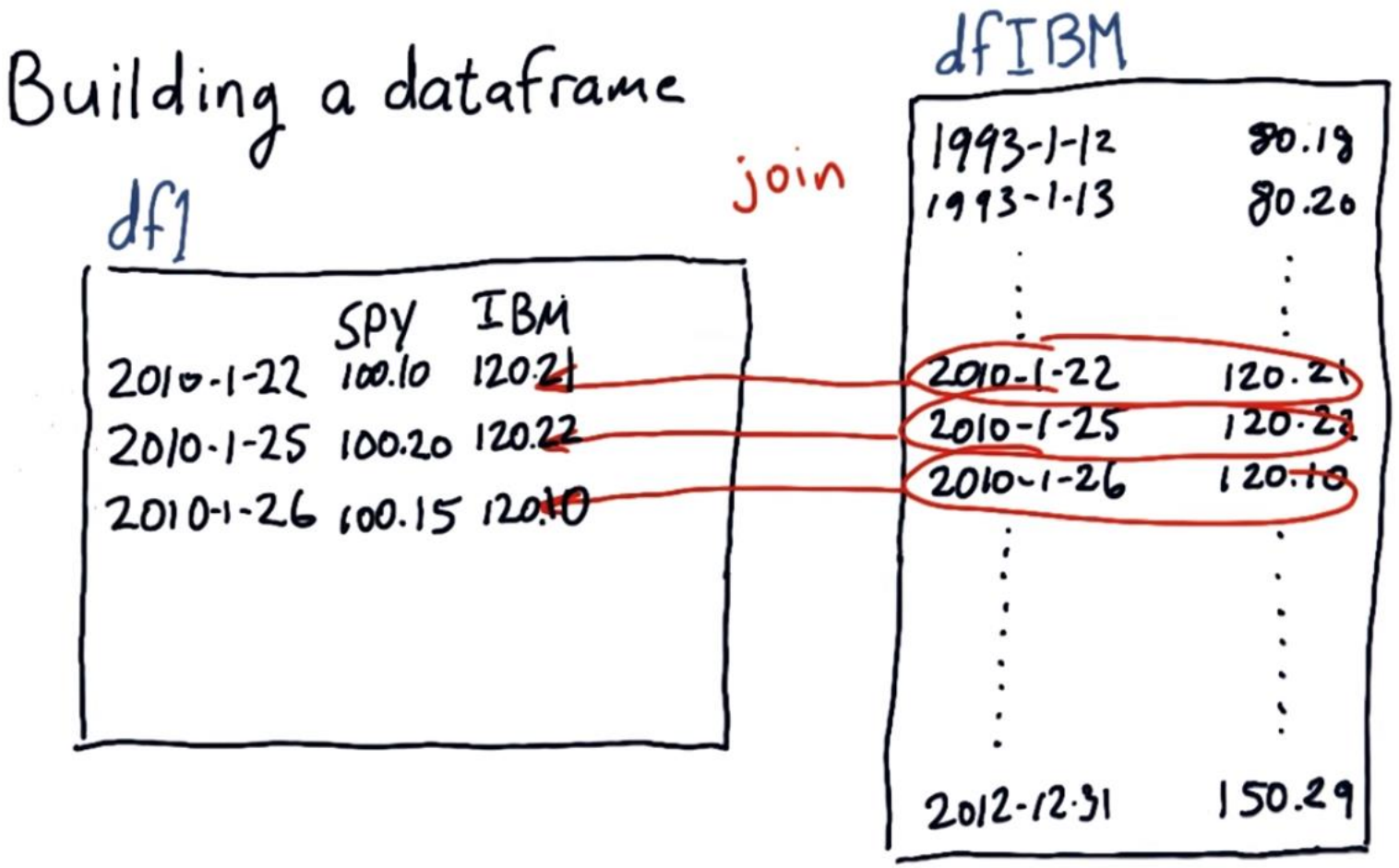
Building a DataFrame



Building a DataFrame



Building a DataFrame: 'JOIN'



Create an empty data frame

```
 1  import pandas as pd
 2  def test_run():
 3      start_date='2016-10-26'
 4      end_date='2016-11-09'
 5      dates=pd.date_range(start_date,end_date)
 6      print dates[0]
 7      df1=pd.DataFrame(index=dates)
 8      print df1
 9  if __name__ == "__main__":
10      test_run()
11
12
13
```

Join SPY data

```
4  def test_run():
5      #Define date range
6      start_date='2010-01-22'
7      end_date='2010-01-26'
8      dates=pd.date_range(start_date,end_date)
9
10     #Create an empty dataframe
11     df1=pd.DataFrame(index=dates)
12
13     #Read SPY data into temporary dataframe
14     dfSPY = pd.read_csv("data/SPY.csv",index_col="Date",
15                         parse_dates=True,usecols=['Date','Adj Close'],
16                         na_values=['nan'])
17
18     #Join the two dataframes using DataFrame.join()
19     df1=df1.join(dfSPY)
20
21     # Drop NaN Values
22     df1 = df1.dropna()
23     print df1
```

Read in more stocks

```
12
13      #Read SPY data into temporary dataframe
14      dfSPY = pd.read_csv("data/SPY.csv",index_col="Date",
15                          parse_dates=True,usecols=['Date','Adj Close'],
16                          na_values=['nan'])
17
18      #Rename 'Adj Close' column to 'SPY' to prevent clash
19      dfSPY = dfSPY.rename(columns={'Adj Close':'SPY'})
20
21      #Join the two dataframes using DataFrame.join(), with how='inner'
22      df1=df1.join(dfSPY,how='inner')
23
24      #Read in more stocks
25      symbols = ['GOOG','IBM','GLD']
26      for symbol in symbols:
27          df_temp=pd.read_csv("data/{}.csv".format(symbol), index_col='Date',
28                              parse_dates=True,usecols=['Date','Adj Close']
29                                                      ,na_values=['nan'])
30           # rename to prevent clash
31          df_temp = df_temp.rename(columns={'Adj Close': symbol})
32          df1=df1.join(df_temp) #use default how='left'
33
```

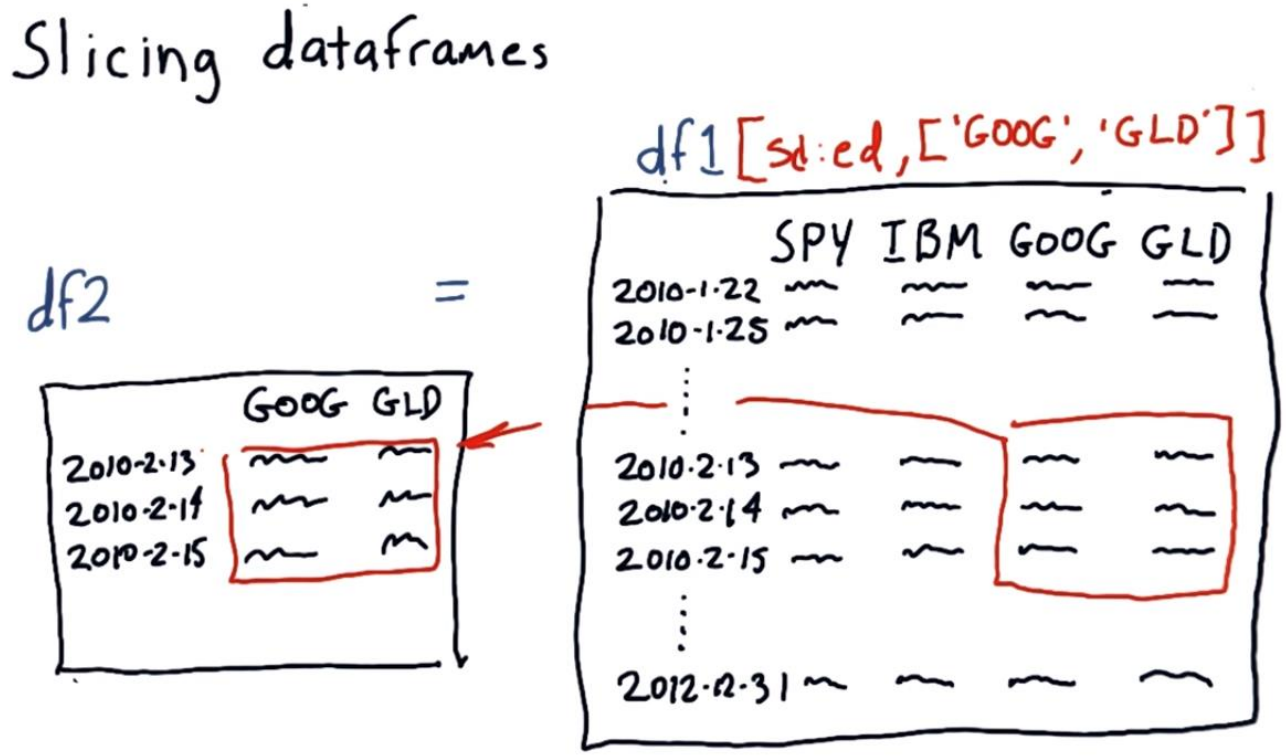
Quiz: Utility functions for reading data

```
 6 def symbol_to_path(symbol, base_dir="data"):
 7     """Return CSV file path given ticker symbol."""
 8     return os.path.join(base_dir, "{}.csv".format(str(symbol)))
 9
10
11 def get_data(symbols, dates):
12     """Read stock data (adjusted close) for given symbols from CSV files."""
13     df = pd.DataFrame(index=dates)
14     if 'SPY' not in symbols:  # add SPY for reference, if absent
15         symbols.insert(0, 'SPY')
16
17     for symbol in symbols:
18         # TODO: Read and join data for each symbol
19
20     return df
21
22
23 def test_run():
24     # Define a date range
25     dates = pd.date_range('2010-01-22', '2010-01-26')
26
27     # Choose stock symbols to read
28     symbols = ['GOOG', 'IBM', 'GLD']
29
30     # Get stock data
31     df = get_data(symbols, dates)
32     print df
33
34
```

Solution

```
4  import pandas as pd
5
6  def symbol_to_path(symbol, base_dir="data"):
7      """Return CSV file path given ticker symbol."""
8      return os.path.join(base_dir, "{}.csv".format(str(symbol)))
9
10
11 def get_data(symbols, dates):
12     """Read stock data (adjusted close) for given symbols from CSV files."""
13     df = pd.DataFrame(index=dates)
14     if 'SPY' not in symbols:  # add SPY for reference, if absent
15         symbols.insert(0, 'SPY')
16
17     for symbol in symbols:
18         df_temp = pd.read_csv(symbol_to_path(symbol), index_col='Date',
19                 parse_dates=True, usecols=['Date', 'Adj Close'], na_values=['nan'])
20         df_temp = df_temp.rename(columns = {'Adj Close': symbol})
21         df = df.join(df_temp)
22         if symbol == 'SPY': #drop dates SPY did not trade
23             df = df.dropna(subset=["SPY"])
24
```

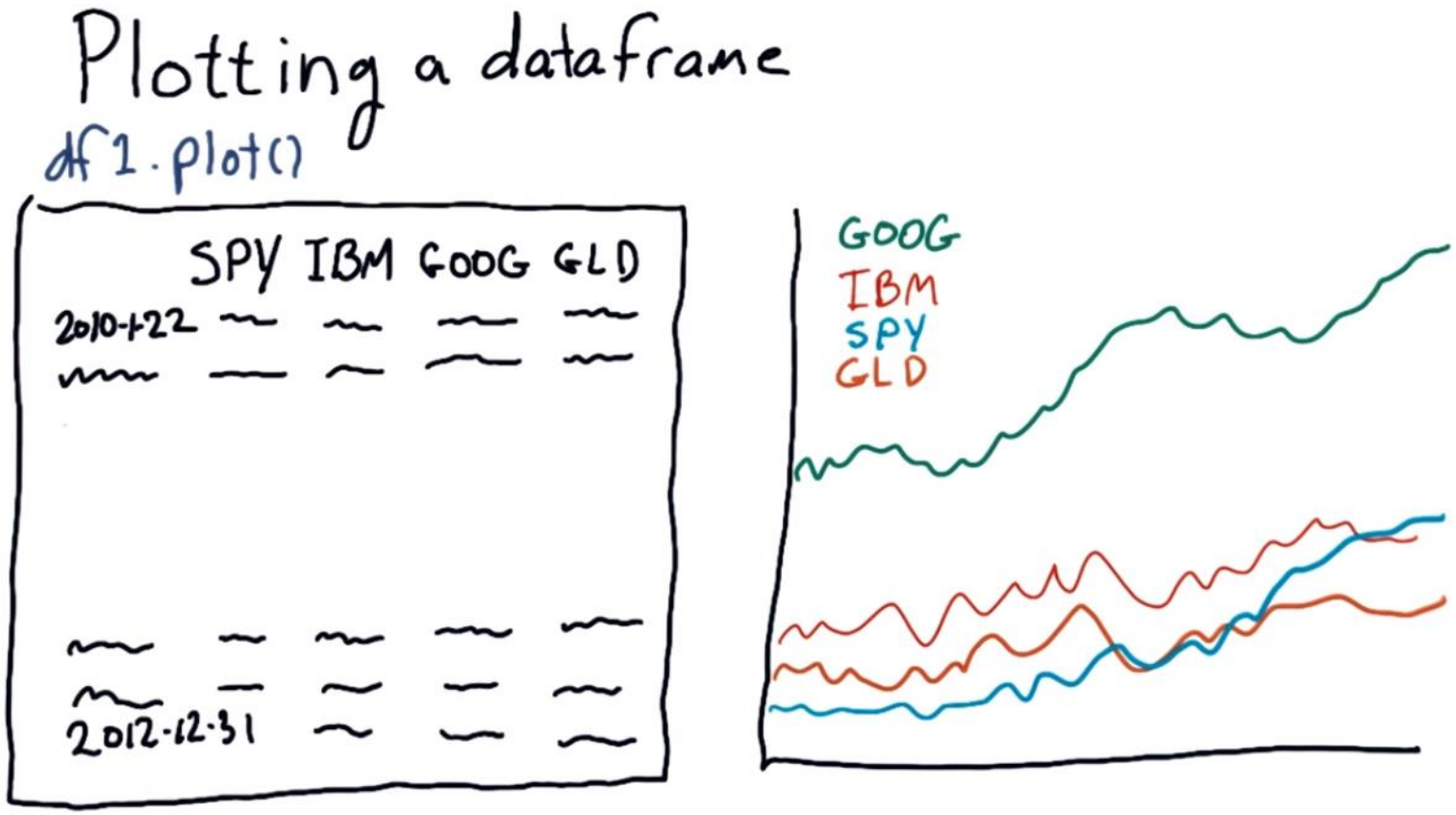
Slicing DataFrame



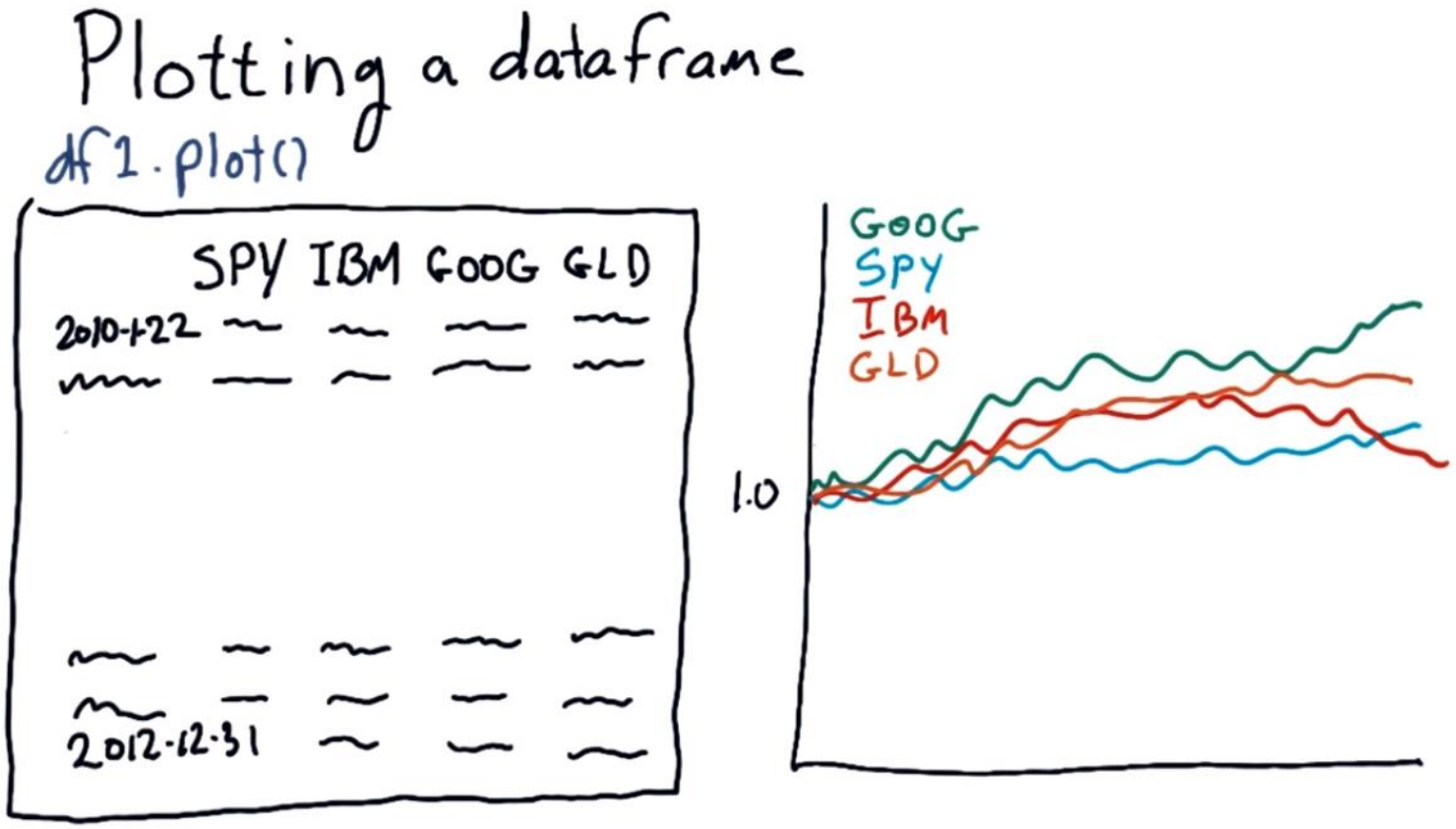
More Slicing

```
31
32      # Get stock data
33      df = get_data(symbols, dates)
34      print df
35      # More slicing
36      print
37      print 'More slicing'
38      print df.ix['2012-10-30': '2012-11-04']
39      print
40      print df['IBM']  #print one label
41      print
42      print df[['IBM', 'AAPL']] #print 2 columns
43      #Slice by row & column:
44      print 'Slice by row & column:'
45      print df.ix['2012-10-28':'2012-11-04',['IBM', 'AAPL']]
46      #normalize
47      df_n = df / df.ix[0]
48      print df_n
49
```

Problems with plotting



Normalization



Quiz: How to plot on "equal footing"?

Quiz: What is the best way to normalize price data so that all prices start at 1.0?

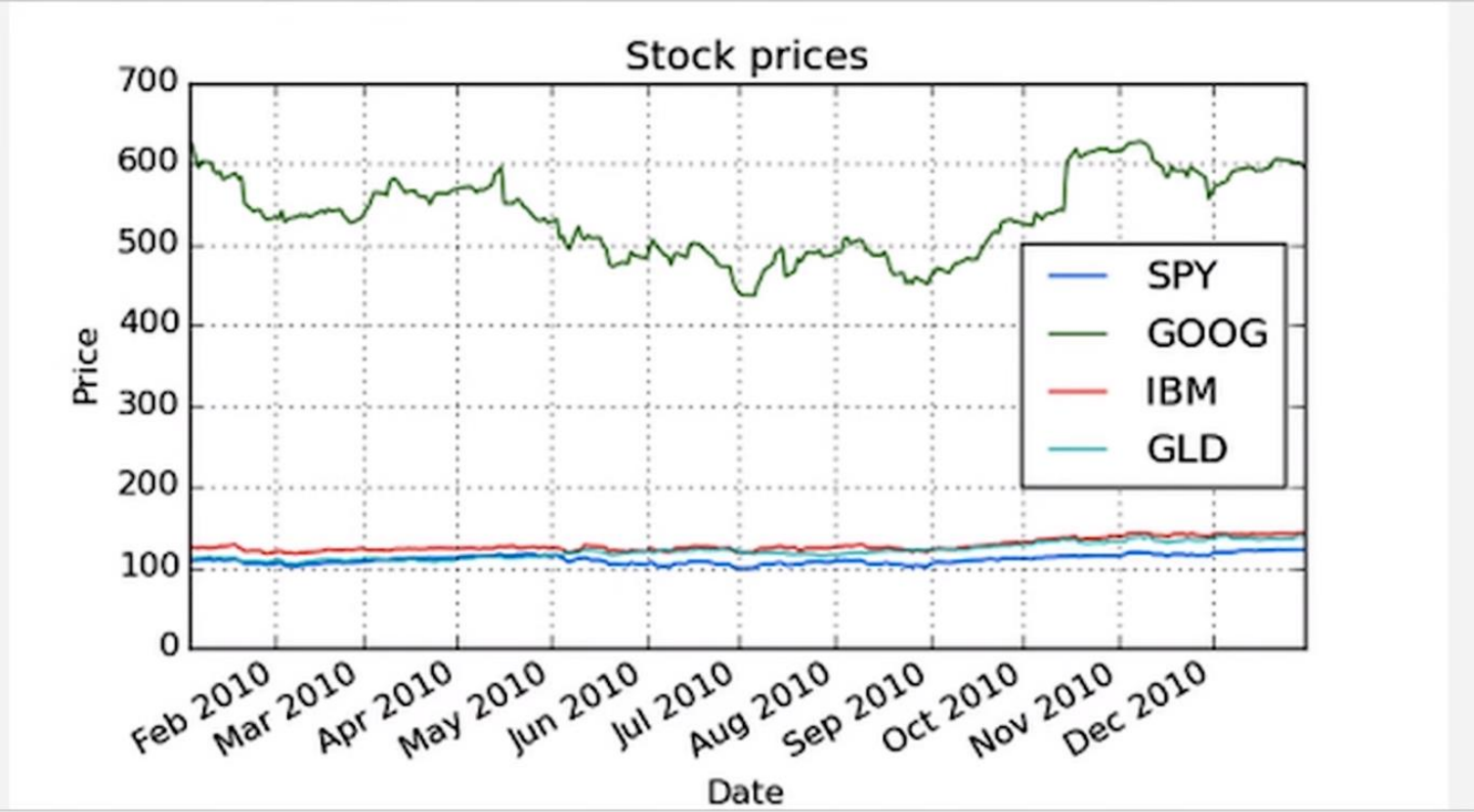
- ☆
```
for date in df1.index:
    for s in symbols:
        df1[date,s] = df1[date,s]/df1[0,s]
```

- ☆
```
df1 = df1/df1[0]
```

Plot Function

```
15      if 'SPY' not in symbols:  # add SPY for reference, if absent
16          symbols.insert(0, 'SPY')
17
18      for symbol in symbols:
19          df_temp = pd.read_csv(symbol_to_path(symbol), index_col='Date',
20                  parse_dates=True, usecols=['Date', 'Adj Close'], na_values=['nan'])
21          df_temp = df_temp.rename(columns={'Adj Close': symbol})
22          df = df.join(df_temp)
23          if symbol == 'SPY':  # drop dates SPY did not trade
24              df = df.dropna(subset=["SPY"])
25
26      return df
27
28  def plot_data(df,title="Stock prices"):
29      '''Plot stock prices'''
30      ax = df.plot(title=title,fontsize=2)
31      ax.set_xlabel("Date")
32      ax.set_ylabel("Price")
33      plt.show() #must be called to show plots in some environments
34
35
36
```

Plotting multiple stocks



Quiz: Slice and plot two stocks

```
"""Slice and plot"""

import os
import pandas as pd
import matplotlib.pyplot as plt


def plot_selected(df, columns, start_index, end_index):
    # TODO: Your code here
    # Note: DO NOT modify anything else!


def symbol_to_path(symbol, base_dir="data"):
    """Return CSV file path given ticker symbol."""
    return os.path.join(base_dir, "{}.csv".format(str(symbol)))


def get_data(symbols, dates):
    """Read stock data (adjusted close) for given symbols from CSV files."""
    df = pd.DataFrame(index=dates)
    if 'SPY' not in symbols:  # add SPY for reference, if absent
```

```
# Slice and plot
plot_selected(df, ['SPY', 'IBM'], '2010-03-01', '2010-04-01')
```

Normalization

```
22          df_temp = df_temp.rename(columns={'Adj Close': symbol})
23          df = df.join(df_temp)
24          if symbol == 'SPY':  # drop dates SPY did not trade
25              df = df.dropna(subset=["SPY"])
26
27      return df
28
29
30  def normalize_data(df):
31      """Normalize stock prices using the first row of the dataframe."""
32      return df/ df.ix[0,:]
33
34
35  def plot_data(df, title="Stock prices"):
36      """Plot stock prices with a custom title and meaningful axis labels."""
37      ax = df.plot(title=title, fontsize=12)
38      ax.set_xlabel("Date")
39      ax.set_ylabel("Price")
40      plt.show()
41
42
43  def test_run():
```

HAPPY CODING !!