```
import pandas as pd


def test_run():
    """Function called by Test Run."""
    df = pd.read_csv("data/AAPL.csv")
    print df.head()


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

Read and Slicing data

```
# Reading in a CSV file
# You can read in the contents of a CSV (comma-separated values) file into a Pandas dataframe using:
df = pd.read_csv(<filename>)

# Selecting rows from a dataframe:
First 5 rows: df.head()
Last 5 rows: df.tail()

# Similarly, last n rows:
df.tail(n)

# Reading Rows
# from row 10 to 20:
df[10:21]
```

Return the statistics

```
# calculate the maximun

"""Compute max price"""

import pandas as pd

def get_max_close(symbol):
    """Return the max closing value 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
    return df['close'].max()
     # return df['Volume'].mean()


def test_run():
    """Function called by Test Run."""
    for symbol in ['AAPL', 'IBM']:
        print "Max close"
        print symbol, get_max_close(symbol)


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

Plot data

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

import pandas as pd
import matplotlib.pyplot as plt

def test_run():
    df = pd.read_csv("data/IBM.csv")
    df['High'].plot()
    # df['Adj Close'].plot()
    plt.show()  # must be called to show plots

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

Plot two columns

```
df[['Adj Close','High']]].plot()
```

Create an empty dataframe

```
import pandas as pd

def test_run():
    start_date='2010-01-01'
    end_date='2012-12-31'
    dates=pd.date_range(start_date, end_date)

    # Create an empty data frame
    df1=pd.DataFrame(index=dates)

    # Read the new data
    dfSPY=pandas.read_csv("data/SPY.csv", index_col="Date",
    parse_date=True, usecols=['Date','Adj Close'],na_values=['nan'
    ])

    # Join the two dataframe using dataframe.join()
    df1=df1.join(dfSPY)

    # Drop the n.a.
    df1=df1.dropna()
    print df1

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

Alternatively, last join and drop function can be shorted as

```
df1=df1.join(dfSPY, how='inner')
```

Such that, the last code block could be wrote as

```
import pandas as pd

def test_run():
    start_date='2010-01-01'
    end_date='2012-12-31'
    dates=pd.date_range(start_date, end_date)

    # Create an empty data frame
    df1=pd.DataFrame(index=dates)
```

```
10
11      # Read the new data
12      dfSPY=pandas.read_csv("data/SPY.csv", index_col="Date",
        parse_date=True, usecols=['Date','Adj Close'],na_values=['nan'
        ])
13
14      # Join the two dataframe using dataframe.join() and drop the
        nan
15      df1=df1.join(dfSPY, how='inner')
16
17      # Rename column name to avoid clash
18      dfSPY=dfSPY.rename(columns={'Adj Close':'SPY'})
19
20      # Read in more stocks
21      symbols=['IBM','GOOG','GLD']
22      for symbol in symbols:
23          df_temp=pandas.read_csv("data/{}.csv".format(symbol),
        index_col="Date", parse_date=True, usecols=['Date','Adj Close'
        ],na_values=['nan'])
24
25          # Rename column name to avoid clash
26          dfSPY=dfSPY.rename(columns={'Adj Close': symbol})
27          df=df1.join(df_temp)
28
29      print df
30
31  if __name__ == "__main__":
32      test_run()
```

Utility Function for reading data:

```
1   """Utility functions"""
2
3   import os
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
        ", parse_dates=True, usecols=['Date','Adj Close'],na_values=['
        nan'])
19          df_temp=df_temp.rename(columns={'Adj Close': symbol})
20          df=df.join(df_temp)
21          df=df.dropna()
22
23      return df
24
```

```
def test_run():
    # Define a date range
    dates = pd.date_range('2010-01-22', '2010-01-26')

    # Choose stock symbols to read
    symbols = ['GOOG', 'IBM', 'GLD']

    # Get stock data
    df = get_data(symbols, dates)
    print df


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

Slicing DataFrame

```
# Slice by row ranges:
df.ix['2001-01-01':'2001-01-31']

# Slice by column:
df['GOOG']
df[['GOOG','IBM']]

# Slice by row and column:
df.ix['2001-01-01':'2001-01-31',['GOOG','IBM']]
```

Plotting multiple stocks

```
import matplotlib.pyplot as plt

def plot_date(df,title="Stock Prices"):
    '''plot stock prices'''
    ax=df.plot(title=title,fontsize=2)
    ax.set_xlable("Date")
    ax.set_ylable("Price")
    plt.show()
```

Slice and Plot

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

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


def plot_selected(df, columns, start_index, end_index):
    """Plot the desired columns over index values in the given
    range."""
    plot_data(df.ix[start_index:end_index,columns], title='Selected
     Data')

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
        symbols.insert(0, 'SPY')

    for symbol in symbols:
        df_temp = pd.read_csv(symbol_to_path(symbol), index_col='Date',
                parse_dates=True, usecols=['Date', 'Adj Close'], na_values=['nan'])
        df_temp = df_temp.rename(columns={'Adj Close': symbol})
        df = df.join(df_temp)
        if symbol == 'SPY':  # drop dates SPY did not trade
            df = df.dropna(subset=["SPY"])

    return df


def plot_data(df, title="Stock prices"):
    """Plot stock prices with a custom title and meaningful axis labels."""
    ax = df.plot(title=title, fontsize=12)
    ax.set_xlabel("Date")
    ax.set_ylabel("Price")
    plt.show()


def test_run():
    # Define a date range
    dates = pd.date_range('2010-01-01', '2010-12-31')

    # Choose stock symbols to read
    symbols = ['GOOG', 'IBM', 'GLD']  # SPY will be added in get_data()

    # Get stock data
    df = get_data(symbols, dates)

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


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

Normalizing data

```
def normalize_data(df):
    """Normalizing the stock price using the 1st row"""
    return df/df.ix[0,:]
```