Data Input and Output

This notebook is the reference code for getting input and output, pandas can read a variety of file types using its pd.read_ methods. Let's take a look at the most common data types:

In [1]:
```
import numpy as np
import pandas as pd
```

CSV

CSV Input

In [2]:
```
df = pd.read_csv('example')
df
```

Out[2]:

	a	b	c	d
0	0	1	2	3
1	4	5	6	7
2	8	9	10	11
3	12	13	14	15

CSV Output

In [3]:
```
df.to_csv('example',index=False)
```

Excel

Pandas can read and write excel files, keep in mind, this only imports data. Not formulas or images, having images or macros may cause this read_excel method to crash.

Excel Input

In [35]:
```
pd.read_excel('Excel_Sample.xlsx',sheetname='Sheet1')
```
Out[35]:

	a	b	c	d
0	0	1	2	3
1	4	5	6	7
2	8	9	10	11
3	12	13	14	15

Excel Output

In [33]:
```
df.to_excel('Excel_Sample.xlsx',sheet_name='Sheet1')
```

HTML

You may need to install htmllib5,lxml, and BeautifulSoup4. In your terminal/command prompt run:

```
conda install lxml
conda install html5lib
conda install BeautifulSoup4
```

Then restart Jupyter Notebook. (or use pip install if you aren't using the Anaconda Distribution)

Pandas can read table tabs off of html. For example:

HTML Input

Pandas read_html function will read tables off of a webpage and return a list of DataFrame objects:

In [5]:
```
df = pd.read_html('http://www.fdic.gov/bank/individual/failed/banklis
```

In [7]:
```
df[0]
```

529	Alamo	Alamo	TN	9961	No Acquirer	8, 2002	2005	none
530	AmTrade International BankEn Espanol	Atlanta	GA	33784	No Acquirer	September 30, 2002	September 11, 2006	none
531	Universal Federal Savings Bank	Chicago	IL	29355	Chicago Community Bank	June 27, 2002	April 9, 2008	none
532	Connecticut Bank of Commerce	Stamford	CT	19183	Hudson United Bank	June 26, 2002	February 14, 2012	none
533	New Century Bank	Shelby Township	MI	34979	No Acquirer	March 28, 2002	March 18, 2005	none
534	Net 1st National Bank	Boca Raton	FL	26652	Bank Leumi USA	March 1, 2002	April 9, 2008	none
535	NextBank, NA	Phoenix	AZ	22314	No Acquirer	February 7, 2002	February 5, 2015	none

SQL (Optional)

- Note: If you are completely unfamiliar with SQL you can check out my other course: "Complete SQL Bootcamp" to learn SQL.

The pandas.io.sql module provides a collection of query wrappers to both facilitate data retrieval and to reduce dependency on DB-specific API. Database abstraction is provided by SQLAlchemy if installed. In addition you will need a driver library for your database. Examples of such drivers are psycopg2 for PostgreSQL or pymysql for MySQL. For SQLite this is included in Python's standard library by default. You can find an overview of supported drivers for each SQL dialect in the SQLAlchemy docs.

If SQLAlchemy is not installed, a fallback is only provided for sqlite (and for mysql for backwards compatibility, but this is deprecated and will be removed in a future version). This mode requires a Python database adapter which respect the Python DB-API.

See also some cookbook examples for some advanced strategies.

The key functions are:

- read_sql_table(table_name, con[, schema, ...])
  - Read SQL database table into a DataFrame.
- read_sql_query(sql, con[, index_col, ...])
  - Read SQL query into a DataFrame.
- read_sql(sql, con[, index_col, ...])
  - Read SQL query or database table into a DataFrame.
- DataFrame.to_sql(name, con[, flavor, ...])
  - Write records stored in a DataFrame to a SQL database.

In [36]:
```
from sqlalchemy import create_engine
```

In [37]:
```
engine = create_engine('sqlite:///:memory:')
```

In [40]:
```
df.to_sql('data', engine)
```

In [42]:
```
sql_df = pd.read_sql('data',con=engine)
```

In [43]:
```
sql_df
```

Out[43]:

	index	a	b	c	d
0	0	0	1	2	3
1	1	4	5	6	7
2	2	8	9	10	11
3	3	12	13	14	15

Great Job!