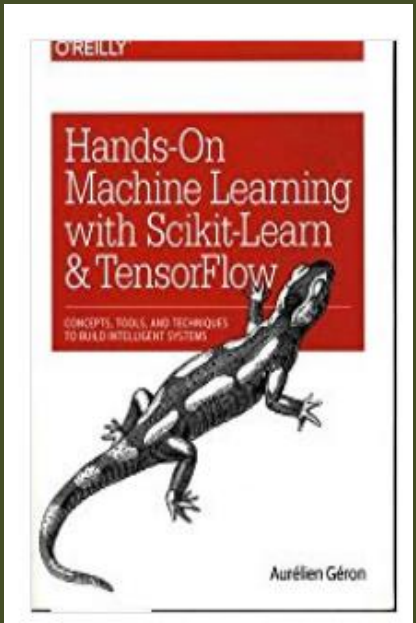
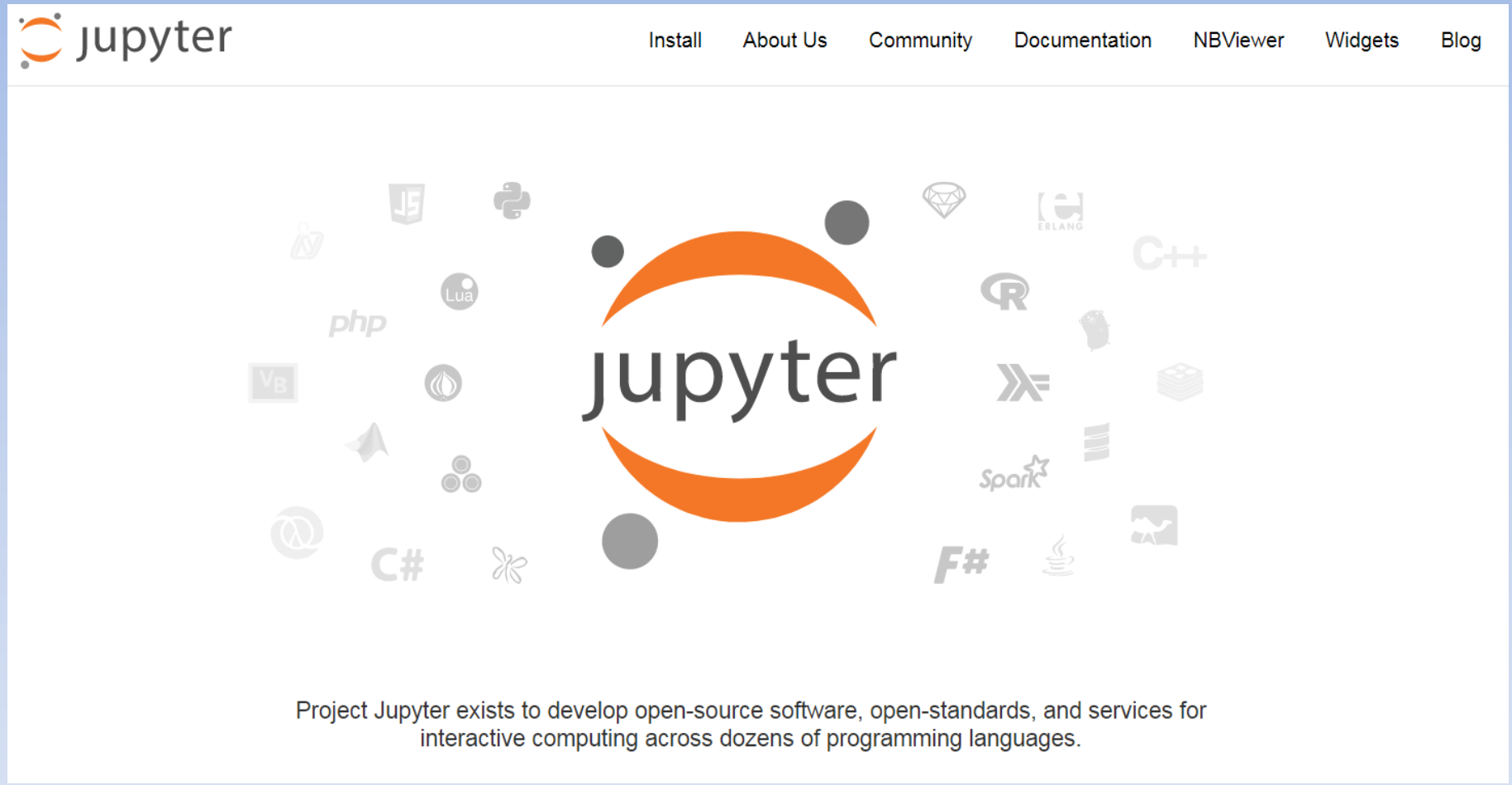


CHAPTER 2: PYTHON STACK

Python Stack --- FREE

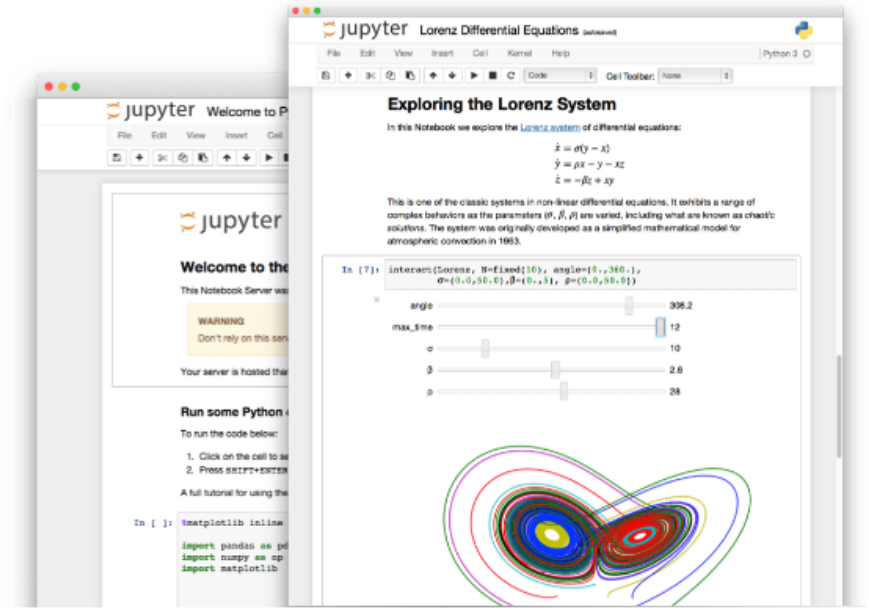
- In order to use text code, you'll need a stack of tools.
- "You will need a number of Python modules: Jupyter, NumPy, Pandas, Matplotlib, and Scikit-Learn."

Jupyter Notebook



jupyter

Install About Us Community Documentation NBViewer Widgets Blog



The Jupyter Notebook

The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more.

Language of choice

The Notebook has support for over 40 programming languages, including Python, R, Julia, and Scala.

Share notebooks

Notebooks can be shared with others using email, Dropbox, GitHub and the Jupyter Notebook Viewer.

Interactive output

Your code can produce rich, interactive output: HTML, images, videos, LaTeX, and custom MIME types.

Big data integration

Leverage big data tools, such as Apache Spark, from Python, R and Scala. Explore that same data with pandas, scikit-learn, ggplot2, TensorFlow.

Steering Council

The role of the Jupyter Steering Council is to ensure, through working with and serving the broader Jupyter community, the long-term well-being of the project, both technically and as a community. The Jupyter Steering Council currently consists of the following members (in alphabetical order).

Damian Avila

Continuum Analytics
@damianavila on GitHub

Matthias Bussonnier

UC Berkeley
@carreau on GitHub

Sylvain Corlay

QuantStack
@sylvaincorlay on GitHub

Jonathan Frederic

Cal Poly, San Luis Obispo
@jdfreder on GitHub

Brian Granger

Cal Poly, San Luis Obispo
@ellisonbg on GitHub

Jason Grout

Bloomberg
@jasongrout on GitHub

Jessica Hamrick

UC Berkeley
@jhamrick on GitHub

Paul Ivanov

Bloomberg
@ivanov on GitHub

Thomas Kluyver

University of Southampton
@takluyver on GitHub

Kyle Kelley

Netflix
@rgbkrk on GitHub

Peter Parente

MaxPoint
@parente on GitHub

Fernando Perez

UC Berkeley
@fperez on GitHub

Min Ragan-Kelley

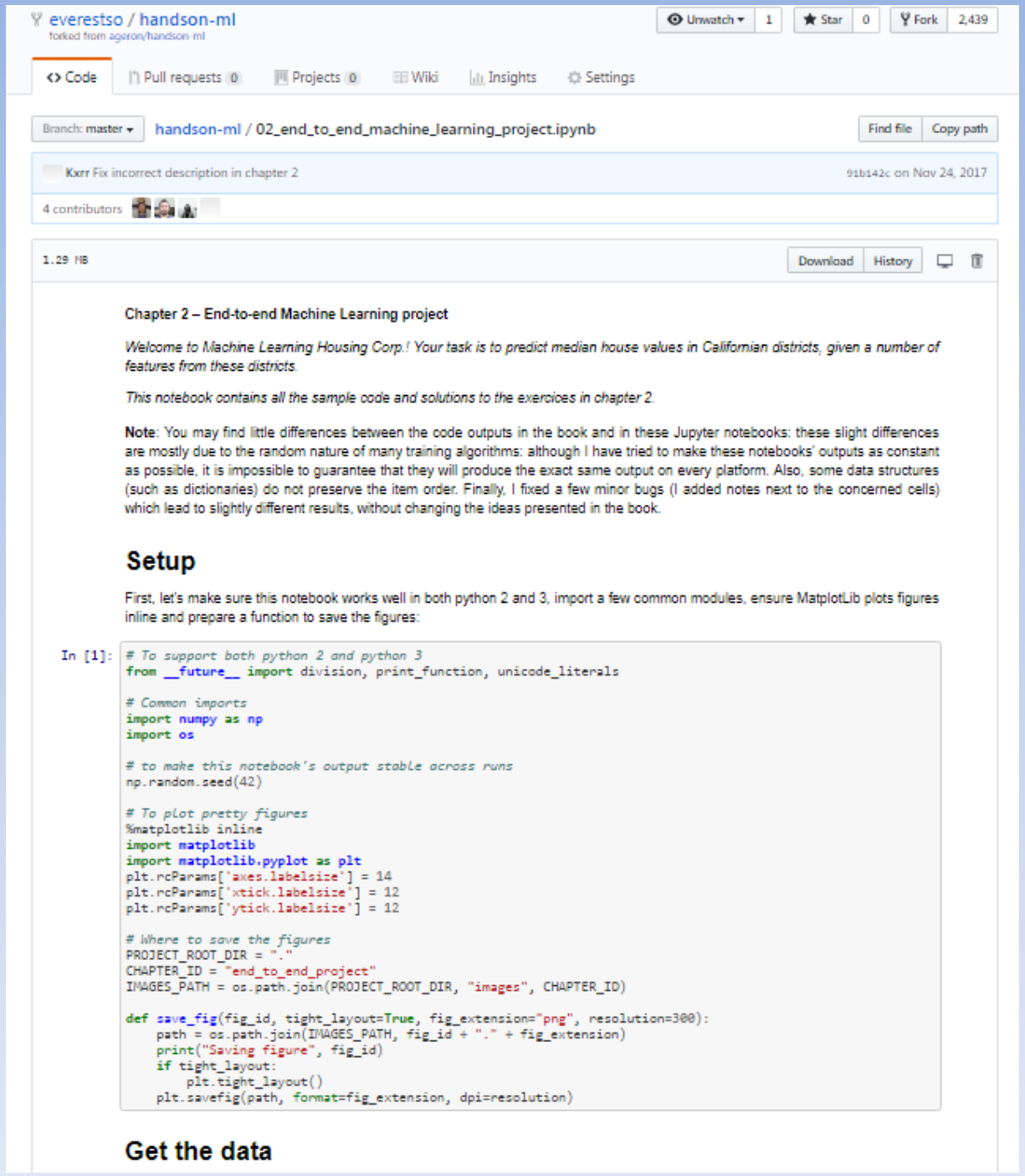
Simula Research Lab
@minrk on GitHub

Steven Silvester

Continuum Analytics
@blink1073 on GitHub

Carol Willing

Cal Poly
@willingc on GitHub



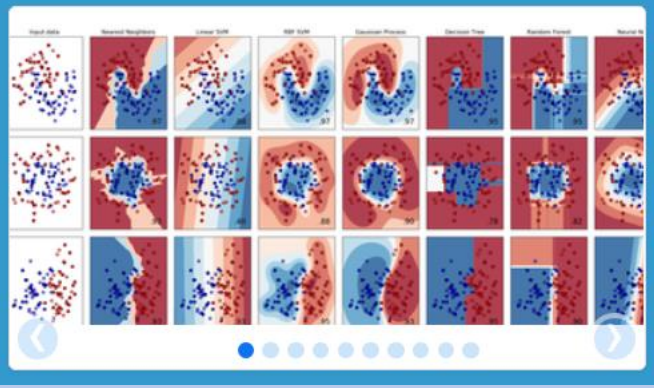
- Course Material in Notebooks on Github

scikit learn

Home Installation Documentation Examples

Google Custom Search Search

Fork me on GitHub



scikit-learn

Machine Learning in Python

- Simple and efficient tools for data mining and data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license

Classification

Identifying to which category an object belongs to.

Applications: Spam detection, Image recognition.
Algorithms: SVM, nearest neighbors, random forest, … — Examples

Regression

Predicting a continuous-valued attribute associated with an object.

Applications: Drug response, Stock prices.
Algorithms: SVR, ridge regression, Lasso, … — Examples

Clustering

Automatic grouping of similar objects into sets.

Applications: Customer segmentation, Grouping experiment outcomes
Algorithms: k-Means, spectral clustering, mean-shift, … — Examples

Dimensionality reduction

Reducing the number of random variables to consider.

Applications: Visualization, Increased efficiency
Algorithms: PCA, feature selection, non-negative matrix factorization. — Examples

Model selection

Comparing, validating and choosing parameters and models.

Goal: Improved accuracy via parameter tuning
Modules: grid search, cross validation, metrics. — Examples

Preprocessing

Feature extraction and normalization.

Application: Transforming input data such as text for use with machine learning algorithms.
Modules: preprocessing, feature extraction.

— Examples

SciPy.org

Sponsored By ENTHOUGHT


Install


Getting Started


Documentation


Report Bugs


Blogs

SciPy (pronounced "Sigh Pie") is a Python-based ecosystem of open-source software for mathematics, science, and engineering. In particular, these are some of the core packages:

NumPy

Base N-dimensional array package

SciPy library

Fundamental library for scientific computing

Matplotlib

Comprehensive 2D Plotting

IP[y]: IPython

IPython

Enhanced Interactive Console

Sympy

Symbolic mathematics

pandas

Data structures & analysis

NumPy

Scipy.org

NumPy

NumPy is the fundamental package for scientific computing with Python. It contains among other things:

- a powerful N-dimensional array object
- sophisticated (broadcasting) functions
- tools for integrating C/C++ and Fortran code
- useful linear algebra, Fourier transform, and random number capabilities

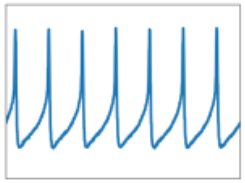
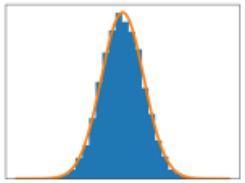
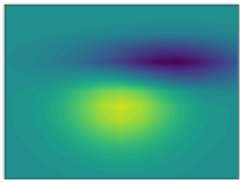
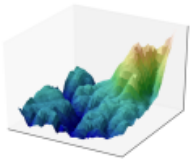
Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined. This allows NumPy to seamlessly and speedily integrate with a wide variety of databases.

NumPy is licensed under the BSD license, enabling reuse with few restrictions.

matplotlib

Introduction

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shell, the jupyter notebook, web application servers, and four graphical user interface toolkits.

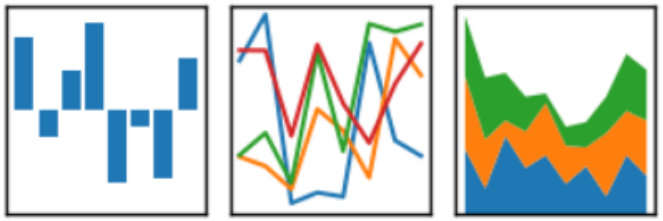





Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, errorcharts, scatterplots, etc., with just a few lines of code. For examples, see the sample plots and thumbnail gallery.

For simple plotting the `pyplot` module provides a MATLAB-like interface, particularly when combined with `IPython`. For the power user, you have full control of line styles, font properties, axes properties, etc, via an object oriented interface or via a set of functions familiar to MATLAB users.

pandas

$$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$$



Python Data Analysis Library

pandas is an open source, BSD-licensed library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language.

pandas is a NumFOCUS sponsored project. This will help ensure the success of development of *pandas* as a world-class open-source project, and makes it possible to donate to the project.

A Fiscally Sponsored Project of

NUMFOCUS

OPEN CODE = BETTER SCIENCE

VERSIONS

Release
0.22.0 - December 2017
download // docs // pdf

Development
0.23.0 - 2018
github // docs

Now Onto Our Notebook

In [1]:

```
# To support both python 2 and python 3
from __future__ import division, print_function, unicode_literals

# Common imports
import numpy as np
import os

# to make this notebook's output stable across runs
np.random.seed(42)

# To plot pretty figures
%matplotlib inline
import matplotlib
import matplotlib.pyplot as plt
plt.rcParams['axes.labelsize'] = 14
plt.rcParams['xtick.labelsize'] = 12
plt.rcParams['ytick.labelsize'] = 12

# Where to save the figures
PROJECT_ROOT_DIR = "."
CHAPTER_ID = "end_to_end_project"
IMAGES_PATH = os.path.join(PROJECT_ROOT_DIR, "images", CHAPTER_ID)

def save_fig(fig_id, tight_layout=True, fig_extension="png", resolution=300):
    path = os.path.join(IMAGES_PATH, fig_id + "." + fig_extension)
    print("Saving figure", fig_id)
    if tight_layout:
        plt.tight_layout()
    plt.savefig(path, format=fig_extension, dpi=resolution)
```

Get Data!

Get the data

In [2]:
```
import os
import tarfile
from six.moves import urllib

DOWNLOAD_ROOT = "https://raw.githubusercontent.com/ageron/handson-ml/master/"
HOUSING_PATH = os.path.join("datasets", "housing")
HOUSING_URL = DOWNLOAD_ROOT + "datasets/housing/housing.tgz"

def fetch_housing_data(housing_url=HOUSING_URL, housing_path=HOUSING_PATH):
    if not os.path.isdir(housing_path):
        os.makedirs(housing_path)
    tgz_path = os.path.join(housing_path, "housing.tgz")
    urllib.request.urlretrieve(housing_url, tgz_path)
    housing_tgz = tarfile.open(tgz_path)
    housing_tgz.extractall(path=housing_path)
    housing_tgz.close()
```

In [3]:
```
fetch_housing_data()
```

Pandas for Data!

In [4]:
```
import pandas as pd

def load_housing_data(housing_path=HOUSING_PATH):
    csv_path = os.path.join(housing_path, "housing.csv")
    return pd.read_csv(csv_path)
```

In [5]:
```
housing = load_housing_data()
housing.head()
```

Out[5]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	media
0	-122.23	37.88	41.0	880.0	129.0	322.0	126.0	8.3252	45260
1	-122.22	37.86	21.0	7099.0	1106.0	2401.0	1138.0	8.3014	35850
2	-122.24	37.85	52.0	1467.0	190.0	496.0	177.0	7.2574	35210
3	-122.25	37.85	52.0	1274.0	235.0	558.0	219.0	5.6431	34130
4	-122.25	37.85	52.0	1627.0	280.0	565.0	259.0	3.8462	34220

IO Tools (Text, CSV, HDF5, ...)

The pandas I/O API is a set of top level `reader` functions accessed like `pd.read_csv()` that generally return a `pandas` object. The corresponding `writer` functions are object methods that are accessed like `df.to_csv()`

Format Type	Data Description	Reader	Writer
text	CSV	read_csv	to_csv
text	JSON	read_json	to_json
text	HTML	read_html	to_html
text	Local clipboard	read_clipboard	to_clipboard
binary	MS Excel	read_excel	to_excel
binary	HDF5 Format	read_hdf	to_hdf
binary	Feather Format	read_feather	to_feather
binary	Parquet Format	read_parquet	to_parquet
binary	Msgpack	read_msgpack	to_msgpack
binary	Stata	read_stata	to_stata
binary	SAS	read_sas	
binary	Python Pickle Format	read_pickle	to_pickle
SQL	SQL	read_sql	to_sql
SQL	Google Big Query	read_gbq	to_gbq

Here is an informal performance comparison for some of these IO methods.

Note: For examples that use the `StringIO` class, make sure you import it according to your Python version, i.e. `from StringIO import StringIO` for Python 2 and `from io import StringIO` for Python 3.

CSV & Text files

The two workhorse functions for reading text files (a.k.a. flat files) are `read_csv()` and `read_table()`. They both use the same parsing code to intelligently convert tabular data into a DataFrame object. See the cookbook for some advanced strategies.

Some Pandas Basics

pandas 0.22.0 documentation »

Table Of Contents

Essential Basic Functionality

Here we discuss a lot of the essential functionality common to the pandas data structures. Here's how to create some of the objects used in the examples from the previous section:

```
In [1]: index = pd.date_range('1/1/2000', periods=8)

In [2]: s = pd.Series(np.random.randn(5), index=['a', 'b', 'c', 'd', 'e'])

In [3]: df = pd.DataFrame(np.random.randn(8, 3), index=index,
   ...:                   columns=['A', 'B', 'C'])
   ...:

In [4]: wp = pd.Panel(np.random.randn(2, 5, 4), items=['Item1', 'Item2'],
   ...:               major_axis=pd.date_range('1/1/2000', periods=5),
   ...:               minor_axis=['A', 'B', 'C', 'D'])
   ...:
```

Head and Tail

To view a small sample of a Series or DataFrame object, use the `head()` and `tail()` methods. The default number of elements to display is five, but you may pass a custom number.

```
In [5]: long_series = pd.Series(np.random.randn(1000))

In [6]: long_series.head()
Out[6]:
0    0.229453
1    0.304418
2    0.736135
3   -0.859631
4   -0.424100
dtype: float64
```

read_csv()
head()

In [4]:
```
import pandas as pd

def load_housing_data(housing_path=HOUSING_PATH):
    csv_path = os.path.join(housing_path, "housing.csv")
    return pd.read_csv(csv_path)
```

In [5]:
```
housing = load_housing_data()
housing.head()
```

Out[5]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	media
0	-122.23	37.88	41.0	880.0	129.0	322.0	126.0	8.3252	45260
1	-122.22	37.86	21.0	7099.0	1106.0	2401.0	1138.0	8.3014	35850
2	-122.24	37.85	52.0	1467.0	190.0	496.0	177.0	7.2574	35210
3	-122.25	37.85	52.0	1274.0	235.0	558.0	219.0	5.6431	34130
4	-122.25	37.85	52.0	1627.0	280.0	565.0	259.0	3.8462	34220