

# Index

## Symbols

- ! (exclamation point) for shell commands, [Interacting with the Operating System](#)
- != (not equal to), [Binary operators and comparisons](#)
- " (double quote)
  - multiline strings, [Strings](#)
  - string literals declared, [Strings](#)
- # (hash mark) for comment, [Comments](#)
- \$ (dollar sign) for shell environment, [Shell Commands and Aliases](#)
- % (percent)
  - datetime formatting, [Converting Between String and Datetime](#)
  - IPython magic commands, [About Magic Commands](#)-[About Magic Commands](#)
    - %bookmark, [Directory Bookmark System](#)
    - Ctrl-C to interrupt running code, [Interrupting running code](#)
    - %debug, [Interactive Debugger](#)-[Other ways to use the debugger](#)
    - executing code from clipboard, [Executing Code from the Clipboard](#)
    - %lprun, [Profiling a Function Line by Line](#)-[Profiling a Function Line by Line](#)
    - operating system commands, [Interacting with the Operating System](#)
    - %prun, [Basic Profiling: %prun and %run -p](#)-[Basic Profiling: %prun and %run -p](#)
    - %run, [The Python Interpreter](#), [The %run Command](#)
    - %run -p, [Basic Profiling: %prun and %run -p](#)-[Basic Profiling: %prun and %run -p](#)
    - %time and %timeit, [Timing Code: %time and %timeit](#)-[Timing Code: %time and %timeit](#)
  - & (ampersand)
    - AND, [Binary operators and comparisons](#)
    - NumPy ndarrays, [Boolean Indexing](#)

- intersection of two sets, [Set](#)
- ' (single quote)
  - multiline strings, [Strings](#)
  - string literals declared, [Strings](#)
- () parentheses
  - calling functions and object methods, [Function and object method calls](#)
  - intervals open (exclusive), [Discretization and Binning](#)
  - method called on results, [Detecting and Filtering Outliers](#)
  - tuples, [Tuple-Tuple methods](#)
    - tuples of exception types, [Errors and Exception Handling](#)
- \* (asterisk)
  - multiplication, [Binary operators and comparisons](#)
    - tuple by integer, [Tuple](#)
    - two-dimensional arrays, [Linear Algebra](#)
  - namespace search in IPython, [Introspection](#)
  - raising to the power, [Binary operators and comparisons](#)
  - rest in tuples, [Unpacking tuples](#)
- + (plus) operator, [Binary operators and comparisons](#)
  - adding strings, [Strings](#)
  - lists, [Concatenating and combining lists](#)
  - Patsy's formulas, [Creating Model Descriptions with Patsy](#)
    - I() wrapper for addition, [Data Transformations in Patsy Formulas](#)
  - timedelta type, [Dates and times](#)
  - tuple concatenation, [Tuple](#)
- - (minus) operator, [Binary operators and comparisons](#)
  - sets, [Set](#)
  - timedelta type, [Dates and times](#)
- / (slash) division operator, [Binary operators and comparisons](#)
  - floor (/), [Numeric types](#)
- :(colon)
  - arrays returned in reverse order, [More About Sorting](#)
  - code blocks, [Indentation, not braces](#)
  - dictionaries, [Dictionary](#)
- ; (semicolon) for multiple statements, [Indentation, not braces](#)
- < (less than) operator, [Binary operators and comparisons](#)

- set elements, [Set](#)
- = (equal sign)
  - test for equality, [Binary operators and comparisons](#)
  - variable assignment, [Variables and argument passing](#)
    - unpacking tuples, [Unpacking tuples](#)
- > (greater than) operator, [Binary operators and comparisons](#)
  - set elements, [Set](#)
- ? (question mark)
  - namespace search in IPython, [Introspection](#)
  - object introspection in IPython, [Introspection](#)
- @ (at) infix operator, [Transposing Arrays and Swapping Axes](#)
- [ ] (square brackets)
  - arrays returned in reverse order, [More About Sorting](#)
    - (see also arrays)
  - intervals closed (inclusive), [Discretization and Binning](#)
  - list definitions, [List](#)
    - slicing lists, [Slicing](#)
  - loc and iloc operators, [Indexing, Selection, and Filtering](#), [Indexing, Selection, and Filtering](#)
  - series indexing, [Indexing, Selection, and Filtering](#)
  - string element index, [String Functions in pandas](#)
  - string slicing, [String Functions in pandas](#)
  - tuple elements, [Tuple](#)
- \ (backslash) escaping in strings, [Strings](#)
- \n (newline character) counted, [Strings](#)
- ^ (caret) operator, [Binary operators and comparisons](#)
  - symmetric difference between sets, [Set](#)
- \_ (underscore)
  - data types with trailing underscores, [NumPy Data Type Hierarchy](#)
  - tab completion and, [Tab Completion](#)
  - unwanted variables, [Unpacking tuples](#)
- { } (curly braces)
  - code blocks surrounded by, [Indentation, not braces](#)
  - dictionaries, [Dictionary](#)
  - formatting strings, [Strings](#)
  - sets, [Set](#)
- | (vertical bar)

- OR, [Binary operators and comparisons](#)
  - NumPy ndarrays, [Boolean Indexing](#)
  - union of two sets, [Set](#)
- ~ (tilde) as NumPy negation operator, [Boolean Indexing](#)

## A

- accumulate() (NumPy ufunc), [ufunc Instance Methods](#)
- add() (NumPy ufunc), [Universal Functions: Fast Element-Wise Array Functions](#)
- add() (sets), [Set](#)
- addition (see plus (+) operator)
- add\_constant() (statsmodels), [Estimating Linear Models](#)
- add\_subplot() (matplotlib), [Figures and Subplots](#)
  - AxesSubplot objects returned, [Figures and Subplots](#)
- aggregate() or agg(), [Data Aggregation, Group Transforms and “Unwrapped” GroupBys](#)
- aggregation of data, [Data Aggregation](#)
  - about, [Data Aggregation and Group Operations](#)
  - agg(), [Data Aggregation, Group Transforms and “Unwrapped” GroupBys](#)
  - column-wise, [Column-Wise and Multiple Function Application](#)
  - describe(), [Data Aggregation, Apply: General split-apply-combine](#)
  - downsampling, [Downsampling-Open-high-low-close \(OHLC\) resampling](#)
- groupby() methods, [Data Aggregation](#)
  - (see also group operations)
  - example of group operation, [How to Think About Group Operations](#)
- indexing a GroupBy object, [Selecting a Column or Subset of Columns](#)
- moving window functions, [Moving Window Functions-User-Defined Moving Window Functions](#)
- multiple functions, [Column-Wise and Multiple Function Application](#)
- NumPy array methods, [Mathematical and Statistical Methods](#)

- open-high-low-close resampling, [Open-high-low-close \(OHLC\) resampling](#)
- performance and, [Data Aggregation](#)
- performance of aggregation functions, [Group Transforms and “Unwrapped” GroupBys](#)
- pivot tables, [Pivot Tables and Cross-Tabulation](#)
  - about, [Pivot Tables and Cross-Tabulation](#)
  - cross-tabulations, [Cross-Tabulations: Crosstab](#)
  - hierarchical indexing, [Hierarchical Indexing](#)
- returning without row indexes, [Returning Aggregated Data Without Row Indexes](#)
- upsampling, [Upsampling and Interpolation](#)
- %alias command, [Shell Commands and Aliases](#)
- Altair for visualization, [Other Python Visualization Tools](#)
- ampersand (&)
  - AND, [Binary operators and comparisons](#)
  - intersection of two sets, [Set](#)
  - NumPy ndarrays, [Boolean Indexing](#)
- anchored offsets in frequencies, [Frequencies and Date Offsets](#)
  - shifting dates with offsets, [Shifting dates with offsets](#)
- annotate() (matplotlib), [Annotations and Drawing on a Subplot](#)
- anonymous (lambda) functions, [Anonymous \(Lambda\) Functions](#)
- Apache Parquet
  - read\_parquet(), [Binary Data Formats](#)
  - remote servers for processing data, [Using HDF5 Format](#)
- APIs
  - importing, [Imports](#)
    - (see also modules)
  - matplotlib primer, [A Brief matplotlib API Primer-matplotlib Configuration](#)
  - web API interactions, [Interacting with Web APIs-Interacting with Web APIs](#)
- apply() (GroupBy), [Apply: General split-apply-combine](#)
- apply() (moving window functions), [User-Defined Moving Window Functions](#)
- arange() (NumPy), [NumPy Basics: Arrays and Vectorized Computation, Creating ndarrays](#)

- `argsort()` (NumPy), [Indirect Sorts: argsort and lexsort](#)
- `array()` (NumPy), [Creating ndarrays](#)
- arrays
  - aggregations defined, [Data Aggregation](#)
    - (see also aggregation of data)
  - associative arrays, [Dictionary](#)
    - (see also dictionaries)
  - dates, [Converting Between String and Datetime](#)
  - moving window functions, [Moving Window Functions](#)-[User-Defined Moving Window Functions](#)
    - binary, [Binary Moving Window Functions](#)
    - expanding window mean, [Moving Window Functions](#)
    - exponentially weighted functions, [Exponentially Weighted Functions](#)
    - rolling operator, [Moving Window Functions](#), [Moving Window Functions](#)
    - user-defined, [User-Defined Moving Window Functions](#)
  - NumPy ndarrays, [The NumPy ndarray: A Multidimensional Array Object](#)
    - (see also ndarrays)
  - pandas Series, [pandas](#), [Series](#)
    - (see also Series)
  - PandasArray, [Series](#)
  - PeriodIndex created from, [Creating a PeriodIndex from Arrays](#)
  - striding information of ndarrays, [ndarray Object Internals](#)
  - structured ndarrays, [Structured and Record Arrays](#)
    - (see also structured ndarrays)
- `asfreq()`, [Period Frequency Conversion, Upsampling and Interpolation](#)
- associative arrays, [Dictionary](#)
  - (see also dictionaries)
- asterisk (\*)
  - multiplication, [Binary operators and comparisons](#)
    - tuple by integer, [Tuple](#)
    - two-dimensional arrays, [Linear Algebra](#)
  - namespace search in IPython, [Introspection](#)
  - raising to the power, [Binary operators and comparisons](#)
  - rest in tuples, [Unpacking tuples](#)

- astype() for extension data types, [Extension Data Types](#), [Categorical Extension Type in pandas](#)
- at (@) infix operator, [Transposing Arrays and Swapping Axes](#)
- attributes of Python objects, [Attributes and methods](#)
- AxesSubplot objects, [Figures and Subplots](#)

## B

- baby names in US dataset, [US Baby Names 1880–2010-Boy names that became girl names \(and vice versa\)](#)
  - naming trends analysis, [Analyzing Naming Trends-Boy names that became girl names \(and vice versa\)](#)
    - gender and naming, [Boy names that became girl names \(and vice versa\)](#)
    - increase in diversity, [Measuring the increase in naming diversity](#)
    - last letter revolution, [The “last letter” revolution](#)
- backslash () escaping in strings, [Strings](#)
- bang (!) for shell commands, [Interacting with the Operating System](#)
- bar plots
  - pandas, [Bar Plots-Bar Plots](#)
  - seaborn, [Bar Plots](#)
  - value\_counts() for, [Bar Plots](#)
- barplot() (seaborn), [Bar Plots](#)
- base frequencies, [Frequencies and Date Offsets](#)
- Beazley, David, [Python Language Basics, IPython, and Jupyter Notebooks](#)
- binary data formats
  - about non-pickle formats, [Binary Data Formats](#)
  - HDF5 format, [Using HDF5 Format-Using HDF5 Format](#), [HDF5 and Other Array Storage Options](#)
  - Microsoft Excel files, [Reading Microsoft Excel Files](#)
  - ndarrays saved, [File Input and Output with Arrays](#)
    - memory-mapped files, [Memory-Mapped Files](#)
  - Parquet (Apache)
    - read\_parquet(), [Binary Data Formats](#)
    - remote servers for processing data, [Using HDF5 Format](#)

- pickle format, [Binary Data Formats](#)
  - caution about, [Binary Data Formats](#)
- plots saved to files, [Saving Plots to File](#)
- binary operators of Python, [Binary operators and comparisons](#)
- binary ufuncs, [Universal Functions: Fast Element-Wise Array Functions](#)
  - methods, [Universal Functions: Fast Element-Wise Array Functions](#), [ufunc Instance Methods](#)-[ufunc Instance Methods](#)
- binding of variables, [Variables and argument passing](#)
- binning data, [Discretization and Binning](#), [Quantile and Bucket Analysis](#)-[Quantile and Bucket Analysis](#), [numpy.searchsorted: Finding Elements in a Sorted Array](#)
- Bitly links with .gov or .mil dataset, [Bitly Data from 1.USA.gov-Counting Time Zones with pandas](#)
  - counting time zones in Python, [Counting Time Zones in Pure Python](#)
  - counting time zones with pandas, [Counting Time Zones with pandas](#)-[Counting Time Zones with pandas](#)
- Bokeh for visualization, [Other Python Visualization Tools](#)
- book website, [Installation and Setup](#), [Data for Examples](#)
- %bookmark command, [Directory Bookmark System](#)
- bool scalar type, [Scalar Types](#), [Booleans](#)
  - type casting, [Type casting](#)
- BooleanDtype extension data type, [Extension Data Types](#)
- box plots, [Facet Grids and Categorical Data](#)
- braces (see curly braces)
- brackets (see square brackets)
- break keyword
  - for loops, [for loops](#)
  - while loops, [while loops](#)
- breakpoint in code, [Other ways to use the debugger](#)
- broadcasting, [Broadcasting-Setting Array Values by Broadcasting](#)
  - about, [Arithmetic with NumPy Arrays](#), [Operations between DataFrame and Series](#)
    - performance tip, [Performance Tips](#)
  - broadcasting rule, [Broadcasting](#)
  - over other axes, [Broadcasting over Other Axes](#)

- setting array values via, [Setting Array Values by Broadcasting](#)
- build\_design\_matrices() (Patsy), [Data Transformations in Patsy](#)
- bytes scalar type, [Scalar Types, Bytes and Unicode](#)

## C

- C/C#/C++ languages
  - about legacy libraries, [Python as Glue](#)
  - HDF5 C library, [Using HDF5 Format](#)
  - NumPy arrays, [NumPy](#), [NumPy Basics: Arrays and Vectorized Computation](#), [Data Types for ndarrays](#)
  - performance tip, [Performance Tips](#)
    - Cython for C-like performance, [Performance Tips](#)
- card deck draw example, [Example: Random Sampling and Permutation](#)
- caret (^) operator, [Binary operators and comparisons](#)
  - symmetric difference between sets, [Set](#)
- casting and conversions of variables, [Dynamic references, strong types](#)
- cat (Unix) to print file to screen, [Reading and Writing Data in Text Format](#), [Writing Data to Text Format](#)
- categorical data
  - any immutable data types, [Categorical Extension Type in pandas](#)
  - background, [Background and Motivation](#)
    - category codes, [Background and Motivation](#)
  - Categorical extension data type, [Categorical Extension Type in pandas](#)
    - cut() and groupby(), [Quantile and Bucket Analysis-Quantile and Bucket Analysis](#)
    - modeling nonnumeric column, [Interfacing Between pandas and Model Code](#)
    - sequences into, [Categorical Extension Type in pandas](#)
  - computations with, [Computations with Categoricals](#)
    - performance improvement with, [Better performance with categoricals](#)
  - dict() for code and category mapping, [Categorical Extension Type in pandas](#)

- meaningful order, [Categorical Extension Type in pandas](#)
- methods, [Categorical Methods-Creating dummy variables for modeling](#)
- Patsy's formulas, [Categorical Data and Patsy](#)-[Categorical Data and Patsy](#)
- visualizing with facet grids, [Facet Grids and Categorical Data](#)-[Facet Grids and Categorical Data](#)
- Categorical extension data type, [Categorical Extension Type in pandas](#)
  - computations with, [Computations with Categoricals](#)
    - performance improvement with, [Better performance with categoricals](#)
  - cut() and groupby(), [Quantile and Bucket Analysis](#)-[Quantile and Bucket Analysis](#)
  - modeling nonnumeric column, [Interfacing Between pandas and Model Code](#)
  - sequences into, [Categorical Extension Type in pandas](#)
- CategoricalDtype extension data type, [Extension Data Types](#)
- catplot() (seaborn), [Facet Grids and Categorical Data](#)
- center() (Patsy), [Data Transformations in Patsy Formulas](#)
- chain() (itertools), [itertools module](#)
- Circle() (matplotlib), [Annotations and Drawing on a Subplot](#)
- clear() for sets, [Set](#)
- clipboard code executed, [Executing Code from the Clipboard](#)
- close() an open file, [Files and the Operating System](#)
- closed property, [Files and the Operating System](#)
- collections
  - built-in sequence functions, [Built-In Sequence Functions](#)
  - dictionaries, [Dictionary](#)-[Valid dictionary key types](#)
  - list comprehensions, [List](#), [Set](#), and [Dictionary Comprehensions](#)
  - sets, [Set](#)-[Set](#)
- colon (:)
  - arrays returned in reverse order, [More About Sorting](#)
  - code blocks, [Indentation, not braces](#)
  - dictionaries, [Dictionary](#)
- colors for plot(), [Colors, Markers, and Line Styles](#)
- combinations() (itertools), [itertools module](#)
- combine\_first(), [Combining Data with Overlap](#)

- command history of IPython, [Using the Command History-Input and Output Variables](#)
  - input and output variables, [Input and Output Variables](#)
  - searching and reusing, [Searching and Reusing the Command History](#)
- comment preceded by hash mark (#), [Comments](#)
- comprehensions for lists, sets, dictionaries, [List, Set, and Dictionary Comprehensions](#)
  - nested, [Nested list comprehensions](#)
- concat() (pandas), [Concatenating Along an Axis-Concatenating Along an Axis](#)
- concatenate() for ndarrays, [Concatenating Along an Axis, Concatenating and Splitting Arrays](#)
  - r\_ and c\_ objects, [Stacking helpers: r\\_ and c\\_](#)
- conda
  - about Miniconda, [Installation and Setup](#)
    - (see also Miniconda)
  - activate to activate environment, [Installing Necessary Packages](#)
  - create to create new environment, [Installing Necessary Packages](#)
  - install, [Installing Necessary Packages](#)
    - necessary packages, [Installing Necessary Packages](#)
    - recommended over pip, [Installing Necessary Packages](#)
  - updating packages, [Installing Necessary Packages](#)
- configuration of IPython, [Profiles and Configuration](#)
- configuration of Jupyter notebooks, [Profiles and Configuration](#)
- console output to variable, [Shell Commands and Aliases](#)
- contiguous memory importance, [The Importance of Contiguous Memory](#)
- continue keyword in for loops, [for loops](#)
- control flow in Python, [Control Flow-range](#)
  - NumPy array vectorized version, [Expressing Conditional Logic as Array Operations](#)
- corr(), [Correlation and Covariance, Example: Group Weighted Average and Correlation, Binary Moving Window Functions](#)
- correlation and covariance with pandas, [Correlation and Covariance-Correlation and Covariance](#)

- binary moving window functions, [Binary Moving Window Functions](#)
- group weighted average and correlation, [Example: Group Weighted Average and Correlation](#)-[Example: Group Weighted Average and Correlation](#)
- count()
  - newlines in multiline string, [Strings](#)
  - nonnull values in groups, [How to Think About Group Operations](#)
  - substring occurrences, [Python Built-In String Object Methods](#)
  - tuple method, [Tuple methods](#)
- cov(), [Correlation and Covariance](#)
- covariance and correlation with pandas, [Correlation and Covariance-Correlation and Covariance](#)
- cProfile module, [Basic Profiling: %prun and %run -p-Basic Profiling: %prun and %run -p](#)
- cross-validation in model training, [Introduction to scikit-learn](#)
- crosstab(), [Cross-Tabulations: Crosstab](#)
  - frequency table via, [Bar Plots](#)
- cross\_val\_score() (scikit-learn), [Introduction to scikit-learn](#)
- CSV (comma-separated values) files
  - reading CSV files
    - about Python files, [itertools module](#)
    - defining format and delimiter, [Working with Other Delimited Formats](#)
  - other delimited formats, [Working with Other Delimited Formats](#)
  - reading in pieces, [Reading Text Files in Pieces](#)
  - read\_csv(), [Reading and Writing Data in Text Format](#)-[Reading and Writing Data in Text Format](#)
  - read\_csv() arguments, [Reading and Writing Data in Text Format](#)
- writing CSV files, [Writing Data to Text Format](#)
  - other delimited formats, [Working with Other Delimited Formats](#)
- csv module
  - Dialect, [Working with Other Delimited Formats](#)
    - option possibilities chart, [Working with Other Delimited Formats](#)
  - importing, [Working with Other Delimited Formats](#)

- opening file with single-character delimiter, [Working with Other Delimited Formats](#)
- Ctrl-C to interrupt running code, [Interrupting running code](#)
- Ctrl-D to exit Python shell, [GNU/Linux](#), [The Python Interpreter](#)
- curly braces ({ })
  - code blocks surrounded by, [Indentation, not braces](#)
  - dictionaries, [Dictionary](#)
  - formatting strings, [Strings](#)
  - sets, [Set](#)
- cut() to divide data into bins, [Discretization and Binning](#)
  - groupby() with, [Quantile and Bucket Analysis-Quantile and Bucket Analysis](#)
  - qcut() per quantiles, [Discretization and Binning](#)
- Cython for C-like performance, [Performance Tips](#)

## D

- data
  - about structured data, [What Kinds of Data?](#)
  - aggregation (see aggregation of data)
  - arrays (see arrays)
  - combining and merging datasets
    - about, [Combining and Merging Datasets](#)
    - combining with overlap, [Combining Data with Overlap](#)
    - concatenating along an axis, [Concatenating Along an Axis-Concatenating Along an Axis](#)
    - joins, [Database-Style DataFrame Joins-Database-Style DataFrame Joins](#)
    - merge key(s) in index, [Merging on Index-Merging on Index](#)
  - datasets on GitHub, [Data for Examples](#)
    - zip files, [Data for Examples](#)
  - dictionaries, [Dictionary-Valid dictionary key types](#)
  - example datasets
    - about, [Data Analysis Examples](#)
    - Bitly links with .gov or .mil, [Bitly Data from 1.USA.gov-Counting Time Zones with pandas](#)

- Federal Election Commission (2012), [2012 Federal Election Commission Database-Donation Statistics by State](#)
- MovieLens, [MovieLens 1M Dataset-Measuring Rating Disagreement](#)
- US baby names, [US Baby Names 1880–2010-Boy names that became girl names \(and vice versa\)](#)
- USDA food database, [USDA Food Database-USDA Food Database](#)
- feature engineering in modeling, [Interfacing Between pandas and Model Code](#)
- Kaggle competition dataset, [Introduction to scikit-learn](#)
- lists, [List-Slicing](#)
- loading, [Data Loading, Storage, and File Formats](#)
  - (see also reading data from a file)
- missing data (see missing data)
- preparation (see data preparation)
- repeated instances of values, [Background and Motivation](#)
  - (see also categorical data)
- reshaping and pivoting
  - hierarchical indexing for, [Hierarchical Indexing, Reshaping with Hierarchical Indexing-Reshaping with Hierarchical Indexing](#)
  - pivoting long to wide format, [Pivoting “Long” to “Wide” Format-Pivoting “Long” to “Wide” Format](#)
  - pivoting wide to long format, [Pivoting “Wide” to “Long” Format-Pivoting “Wide” to “Long” Format](#)
- scalars, [Scalar Types-Dates and times](#)
  - type casting, [Type casting](#)
- sequence functions built in, [Built-In Sequence Functions](#)
- sets, [Set-Set](#)
- shifting through time, [Shifting \(Leading and Lagging\) Data-Shifting dates with offsets](#)
- time series, [Time Series](#)
  - (see also time series)
- tuples, [Tuple-Tuple methods](#)
- variables as objects, [Variables and argument passing](#)
  - dynamic references, strong types, [Dynamic references, strong types](#)

- data analysis
  - about Python, [Why Python for Data Analysis?](#)
    - drawbacks, [Why Not Python?](#)
  - about the book, [What Is This Book About?](#)
  - categorical data background, [Background and Motivation](#)
    - (see also categorical data)
  - email lists, [Community and Conferences](#)
  - example datasets
    - about, [Data Analysis Examples](#)
    - Bitly links with .gov or .mil, [Bitly Data from 1.USA.gov-Counting Time Zones with pandas](#)
    - Federal Election Commission (2012), [2012 Federal Election Commission Database-Donation Statistics by State](#)
    - MovieLens, [MovieLens 1M Dataset-Measuring Rating Disagreement](#)
    - US baby names, [US Baby Names 1880–2010-Boy names that became girl names \(and vice versa\)](#)
    - USDA food database, [USDA Food Database-USDA Food Database](#)
- data loading, [Data Loading, Storage, and File Formats](#)
  - (see also reading data from a file)
- data preparation
  - about, [Data Cleaning and Preparation](#)
  - categorical data
    - background, [Background and Motivation](#)
    - Categorical extension data type, [Categorical Extension Type in pandas](#)
    - computations with, [Computations with Categoricals](#)
    - meaningful order, [Categorical Extension Type in pandas](#)
    - methods, [Categorical Methods-Creating dummy variables for modeling](#)
    - performance improvement with, [Better performance with categoricals](#)
  - combining and merging datasets
    - about, [Combining and Merging Datasets](#)
    - combining with overlap, [Combining Data with Overlap](#)
    - concatenating along an axis, [Concatenating Along an Axis- Concatenating Along an Axis](#)

- joins, [Database-Style DataFrame Joins-Database-Style DataFrame Joins](#)
- merge key(s) in index, [Merging on Index-Merging on Index](#)
- data transformation
  - aggregations defined, [Data Aggregation](#)
    - (see also aggregation of data)
  - axis index map(), [Renaming Axis Indexes](#)
  - categorical variable into dummy matrix, [Computing Indicator/Dummy Variables](#)
  - discretization and binning, [Discretization and Binning](#), [Quantile and Bucket Analysis-Quantile and Bucket Analysis](#),  
[numpy.searchsorted: Finding Elements in a Sorted Array](#)
  - duplicates removed, [Removing Duplicates](#)
  - mapping or function for, [Transforming Data Using a Function or Mapping](#)
  - outlier detection and filtering, [Detecting and Filtering Outliers](#)
  - permutation, [Permutation and Random Sampling](#)
  - random sampling, [Permutation and Random Sampling](#)
  - replacing values, [Replacing Values](#)
- extension data types, [Extension Data Types](#), [String Functions in pandas](#)
- missing data, [Handling Missing Data](#)
  - (see also missing data)
- reshaping and pivoting
  - hierarchical indexing for, [Hierarchical Indexing](#), [Reshaping with Hierarchical Indexing-Reshaping with Hierarchical Indexing](#)
  - pivoting long to wide format, [Pivoting “Long” to “Wide” Format-Pivoting “Long” to “Wide” Format](#)
  - pivoting wide to long format, [Pivoting “Wide” to “Long” Format-Pivoting “Wide” to “Long” Format](#)
- string manipulation
  - built-in string object methods, [Python Built-In String Object Methods](#)
  - regular expressions, [Regular Expressions-Regular Expressions](#)
  - string functions in pandas, [String Functions in pandas-String Functions in pandas](#)

- Data Science from Scratch: First Principles with Python (Grus), [Conclusion](#)
- data types
  - about NumPy, [Data Types for ndarrays](#)
  - DataFrame to\_numpy(), [DataFrame](#)
  - date and time, [Date and Time Data Types and Tools](#)
  - dtype property, [The NumPy ndarray: A Multidimensional Array Object](#), [Creating ndarrays](#), [Data Types for ndarrays](#)-[Data Types for ndarrays](#), [ndarray Object Internals](#)
  - extension data types, [Extension Data Types](#), [String Functions in pandas](#)
  - NaN for missing data, [Handling Missing Data](#)
  - NumPy ndarrays, [Creating ndarrays](#), [Data Types for ndarrays](#)-[Data Types for ndarrays](#), [ndarray Object Internals](#)
    - hierarchy of data types, [NumPy Data Type Hierarchy](#)
    - type casting, [Data Types for ndarrays](#)
    - ValueError, [Data Types for ndarrays](#)
  - structured ndarrays, [Structured and Record Arrays](#)
    - (see also structured ndarrays)
  - time and date, [Date and Time Data Types and Tools](#)
  - trailing underscores in names, [NumPy Data Type Hierarchy](#)
  - type casting, [Type casting](#)
    - NumPy ndarrays, [Data Types for ndarrays](#)
  - type inference in reading text data, [Reading and Writing Data in Text Format](#)
- database interactions, [Interacting with Databases](#)-[Interacting with Databases](#)
- DataFrames (pandas), [DataFrame](#)-[DataFrame](#)
  - about, [pandas](#), [pandas](#)
  - apply() function, [Function Application and Mapping](#)
  - applymap() for element-wise functions, [Function Application and Mapping](#)
  - arithmetic, [Arithmetic and Data Alignment](#)
    - with fill values, [Arithmetic methods with fill values](#)
    - arithmetic methods chart, [Arithmetic methods with fill values](#)
    - arithmetic with Series, [Operations between DataFrame and Series](#)

- axis indexes with duplicate labels, [Axis Indexes with Duplicate Labels](#)
- categorical data from column, [Categorical Extension Type in pandas](#)
- chained indexing pitfalls, [Pitfalls with chained indexing](#)
- columns retrieved as Series, [DataFrame](#)
- concatenating along an axis, [Concatenating Along an Axis- Concatenating Along an Axis](#)
- constructing, [DataFrame](#)
  - possible data inputs chart, [DataFrame](#)
- dropping entries from an axis, [Dropping Entries from an Axis](#)
- duplicate rows removed, [Removing Duplicates](#)
- get() HTTP request data, [Interacting with Web APIs](#)
- get\_dummies(), [Computing Indicator/Dummy Variables](#)
- HDF5 binary data format, [Using HDF5 Format](#)
- head() and tail(), [DataFrame](#)
- hierarchical indexing, [DataFrame](#), [Hierarchical Indexing](#)
  - indexing with columns, [Indexing with a DataFrame's columns](#)
  - reordering and sorting levels, [Reordering and Sorting Levels](#)
  - reshaping data, [Hierarchical Indexing](#), [Reshaping with Hierarchical Indexing](#)-[Reshaping with Hierarchical Indexing](#)
  - summary statistics by level, [Summary Statistics by Level](#)
- importing into local namespace, [Getting Started with pandas](#)
- Index objects, [Index Objects](#)
  - map() to transform data, [Renaming Axis Indexes](#)
- indexes for row and column, [DataFrame](#)
  - hierarchical indexing, [DataFrame](#)
- indexing options chart, [Selection on DataFrame with loc and iloc](#)
- integer indexing pitfalls, [Integer indexing pitfalls](#)
- JSON data to and from, [JSON Data](#)
- Jupyter notebook display of, [DataFrame](#)
- missing data
  - dropna() to filter out, [Filtering Out Missing Data](#)-[Filtering Out Missing Data](#)
  - fillna() to fill in, [Filling In Missing Data](#)-[Filling In Missing Data](#)
- NumPy ufuncs and, [Function Application and Mapping](#)
- objects that have different indexes, [Arithmetic and Data Alignment](#)

- outlier detection and filtering, [Detecting and Filtering Outliers](#)
- ranking, [Sorting and Ranking](#)
- reading data from a file (see reading data from a file)
- reindexing, [Reindexing](#)
- selection with loc and iloc, [Selection on DataFrame with loc and iloc](#)
  - row retrieval, [DataFrame](#)
- sorting, [Sorting and Ranking](#)
- SQL query results into, [Interacting with Databases-Interacting with Databases](#)
- statistical methods
  - correlation and covariance, [Correlation and Covariance-Correlation and Covariance](#)
  - summary statistics, [Summarizing and Computing Descriptive Statistics-Summarizing and Computing Descriptive Statistics](#)
  - summary statistics by level, [Summary Statistics by Level](#)
- to\_numpy(), [DataFrame](#)
- unique and other set logic, [Unique Values, Value Counts, and Membership-Unique Values, Value Counts, and Membership](#)
- writing data to a file (see writing data to a file)
- date offset, [Frequencies and Date Offsets](#)
  - periods, [Periods and Period Arithmetic](#)
- date type, [Dates and times-Dates and times, Date and Time Data Types and Tools](#)
- datetime module, [Dates and times-Dates and times, Date and Time Data Types and Tools](#)
- datetime type, [Dates and times-Dates and times, Date and Time Data Types and Tools](#)
  - converting between string and, [Converting Between String and Datetime-Converting Between String and Datetime](#)
  - formatting as string, [Dates and times, Converting Between String and Datetime-Converting Between String and Datetime](#)
  - immutable, [Dates and times](#)
  - ISO 8601 format parsed, [Converting Between String and Datetime](#)
  - locale-specific formatting options, [Converting Between String and Datetime](#)
  - shifting dates with offsets, [Shifting dates with offsets](#)

- time series basics, [Time Series Basics](#)
- DatetimeIndex, [Time Series Basics](#)
- DatetimeTZDtype extension data type, [Extension Data Types](#)
- date\_range(), [Generating Date Ranges](#)
  - normalize option, [Generating Date Ranges](#)
- %debug command, [Interactive Debugger](#)-[Other ways to use the debugger](#)
- debug() to call debugger, [Other ways to use the debugger](#)
- debugger in IPython, [Interactive Debugger](#)-[Other ways to use the debugger](#)
  - chart of commands, [Interactive Debugger](#)
- deck of card random sampling example, [Example: Random Sampling and Permutation](#)
- def to declare a function, [Functions](#)
- delimiters separating text fields
  - reading a CSV file, [Reading and Writing Data in Text Format](#)
  - reading other delimited formats, [Working with Other Delimited Formats](#)
  - writing a CSV file, [Writing Data to Text Format](#)
  - writing other delimited formats, [Working with Other Delimited Formats](#)
- density plots and histograms, [Histograms and Density Plots](#)-  
[Histograms and Density Plots](#)
- describe() in aggregated data, [Data Aggregation](#), [Apply: General split-apply-combine](#)
- descriptive statistics with pandas, [Summarizing and Computing Descriptive Statistics](#)-[Summarizing and Computing Descriptive Statistics](#)
- DesignMatrix instances of Patsy, [Creating Model Descriptions with Patsy](#)
- development environment, [Integrated Development Environments and Text Editors](#)
  - (see also software development tools)
- dict(), [Creating dictionaries from sequences](#)
  - categorical data, [Categorical Extension Type in pandas](#)
- dictionaries (dict), [Dictionary-Valid dictionary key types](#)
  - DataFrame as dictionary of Series, [DataFrame](#)

- constructing a DataFrame, [DataFrame](#)
- defaultdict(), [Default values](#)
- dictionary comprehensions, [List, Set, and Dictionary Comprehensions](#)
  - reading data from delimited file, [Working with Other Delimited Formats](#)
- get() HTTP request data, [Interacting with Web APIs](#)
- get() versus pop() when key not present, [Default values](#)
- grouping via, [Grouping with Dictionaries and Series](#)
- HDF5 binary data format, [Using HDF5 Format](#)
- keys() and values(), [Dictionary](#)
- merging, [Dictionary](#)
- pandas Series as, [Series](#)
  - Series from and to dictionary, [Series](#)
- sequences into, [Creating dictionaries from sequences](#)
- setdefault(), [Default values](#)
- valid key types, [Valid dictionary key types](#)
- difference() for sets, [Set](#)
  - DataFrame method, [Index Objects](#)
- difference\_update() for sets, [Set](#)
- dimension tables, [Background and Motivation](#)
- division
  - division operator (/), [Binary operators and comparisons](#)
  - floor (//), [Binary operators and comparisons](#), [Numeric types](#)
  - integer division, [Numeric types](#)
- dmatrices() (Patsy), [Creating Model Descriptions with Patsy](#)
- documentation online
  - IPython, [Tab Completion](#)
  - pandas, [Time Series](#)
  - Python
    - formatting strings, [Strings](#)
    - itertools functions, [itertools module](#)
- dollar sign (\$) for shell environment, [Shell Commands and Aliases](#)
- dot() (NumPy) for matrix multiplication, [Linear Algebra](#)
- double quote (")
  - multiline strings, [Strings](#)
  - string literals declared, [Strings](#)

- downsampling, [Resampling and Frequency Conversion](#), [Downsampling-Open-high-low-close \(OHLC\) resampling](#)
  - target period as subperiod, [Resampling with Periods](#)
- dropna() filter for missing data, [Handling Missing Data-Filtering Out Missing Data](#)
- drop\_duplicates() for DataFrame rows, [Removing Duplicates](#)
- DST (daylight saving time), [Time Zone Handling](#), [Operations with Time Zone-Aware Timestamp Objects](#)
- dtype property, [The NumPy ndarray: A Multidimensional Array Object](#), [Creating ndarrays](#), [Data Types for ndarrays](#)-[Data Types for ndarrays](#), [ndarray Object Internals](#)
- duck typing of objects, [Duck typing](#)
- duplicated() for DataFrame rows, [Removing Duplicates](#)
- duplicates removed from data, [Removing Duplicates](#)

## E

- Effective Python (Slatkin), [Python Language Basics](#), [IPython](#), and [Jupyter Notebooks](#)
- elapsed time, [Time Series](#)
- else, if, elif, [if, elif, and else](#)
  - NumPy array vectorized version, [Expressing Conditional Logic as Array Operations](#)
- email lists for data-related Python, [Community and Conferences](#)
- encoding of files
  - encoding property, [Files and the Operating System](#)
  - open(), [Files and the Operating System](#)
    - converting between encodings, [Bytes and Unicode with Files](#)
- encoding of strings, [Bytes and Unicode](#)
- end-of-line (EOL) markers, [Files and the Operating System](#)
- enumerate(), [enumerate](#)
- equal sign (=)
  - set update methods, [Set](#)
  - test for equality, [Binary operators and comparisons](#)
  - variable assignment, [Variables and argument passing](#)
    - unpacking tuples, [Unpacking tuples](#)

- errors and exception handling, [Errors and Exception Handling-Exceptions in IPython](#)
  - AssertionError and debugger, [Interactive Debugger-Other ways to use the debugger](#)
  - broadcasting, [Broadcasting over Other Axes](#)
  - debugger in IPython, [Interactive Debugger](#)
  - integer indexing, [Integer indexing pitfalls](#)
  - IPython exceptions, [Exceptions in IPython](#)
  - periods and resampling, [Resampling with Periods](#)
  - raise\_for\_status() for HTTP errors, [Interacting with Web APIs](#)
  - SettingWithCopyWarning, [Pitfalls with chained indexing](#)
  - substring find() versus index(), [Python Built-In String Object Methods](#)
  - time zone-aware data with naive, [Operations Between Different Time Zones](#)
  - try/except blocks, [Errors and Exception Handling](#)
  - ValueError in NumPy casting, [Data Types for ndarrays](#)
- ewm() for exponentially weighted moving functions, [Exponentially Weighted Functions](#)
- example datasets
  - about, [Data Analysis Examples](#)
  - Bitly links with .gov or .mil, [Bitly Data from 1.USA.gov-Counting Time Zones with pandas](#)
  - Federal Election Commission (2012), [2012 Federal Election Commission Database-Donation Statistics by State](#)
  - MovieLens, [MovieLens 1M Dataset-Measuring Rating Disagreement](#)
  - US baby names, [US Baby Names 1880–2010-Boy names that became girl names \(and vice versa\)](#)
- ExcelFile class (pandas), [Reading Microsoft Excel Files](#)
- exceptions (see errors and exception handling)
- exclamation point (!) for shell commands, [Interacting with the Operating System](#)
- executing code from clipboard, [Executing Code from the Clipboard](#)
- execution time measured, [Timing Code: %time and %timeit-Timing Code: %time and %timeit](#)
- exit() to exit Python shell, [The Python Interpreter](#)
  - GNU/Linux, [GNU/Linux](#)

- macOS, [Miniconda on macOS](#)
- Windows, [Miniconda on Windows](#)
- exp0 (NumPy ufunc), [Universal Functions: Fast Element-Wise Array Functions](#)
- expanding() in moving window functions, [Moving Window Functions](#)
- experimental time series, [Time Series](#)
- exporting data (see writing data to a file)
- extension data types, [Extension Data Types](#), [String Functions in pandas](#)
  - astype(), [Extension Data Types](#), [Categorical Extension Type in pandas](#)
- extract() regex from string, [String Functions in pandas](#)

## F

- f-strings, [Strings](#)
- facet grids, [Facet Grids and Categorical Data](#)-[Facet Grids and Categorical Data](#)
- False, [Booleans](#)
- fancy indexing by NumPy ndarrays, [Fancy Indexing](#)-[Fancy Indexing](#)
  - take() and put(), [Fancy Indexing Equivalents: take and put](#)
- feature engineering in modeling, [Interfacing Between pandas and Model Code](#)
- Federal Election Commission dataset, [2012 Federal Election Commission Database](#)-[Donation Statistics by State](#)
  - bucketing donation amounts, [Bucketing](#) [Donation Amounts](#)-[Bucketing](#) [Donation Amounts](#)
  - donations by occupation and employer, [Donation Statistics by Occupation and Employer](#)-[Donation Statistics by Occupation and Employer](#)
  - donations by state, [Donation Statistics by State](#)
- figure() (matplotlib), [Figures and Subplots](#)
  - add\_subplot() (matplotlib), [Figures and Subplots](#)
    - AxesSubplot objects returned, [Figures and Subplots](#)
    - savefig(), [Saving Plots to File](#)
- files in Python, [Files and the Operating System](#)-[Bytes and Unicode with Files](#)

- binary data formats
  - about non-pickle formats, [Binary Data Formats](#)
  - HDF5 format, [Using HDF5 Format](#)-[Using HDF5 Format](#), [HDF5 and Other Array Storage Options](#)
  - memory-mapped files, [Memory-Mapped Files](#)
  - Microsoft Excel files, [Reading Microsoft Excel Files](#)
  - ndarrays saved, [File Input and Output with Arrays](#)
  - Parquet (Apache), [Binary Data Formats](#), [Using HDF5 Format](#)
  - pickle format, [Binary Data Formats](#)
  - pickle format caution, [Binary Data Formats](#)
  - plots saved to files, [Saving Plots to File](#)
- database interactions, [Interacting with Databases](#)-[Interacting with Databases](#)
- methods most commonly used, [Files and the Operating System](#)
- modes, [Files and the Operating System](#)
  - binary mode, [Bytes and Unicode with Files](#)
  - open() default read only, [Files and the Operating System](#)
  - open() write-only modes, [Files and the Operating System](#)
  - text mode default, [Bytes and Unicode with Files](#)
- open(), [Files and the Operating System](#)
  - close() when finished, [Files and the Operating System](#)
  - converting between encodings, [Bytes and Unicode with Files](#)
  - default read only mode, [Files and the Operating System](#)
  - read/write modes, [Files and the Operating System](#)
  - with statement for clean-up, [Files and the Operating System](#)
  - write-only modes, [Files and the Operating System](#)
  - writing delimited files, [Working with Other Delimited Formats](#)
- plots saved to files, [Saving Plots to File](#)
- reading text data, [Reading and Writing Data in Text Format](#)-[Reading and Writing Data in Text Format](#)
  - CSV files, [Reading and Writing Data in Text Format](#)-[Reading and Writing Data in Text Format](#)
  - CSV files of other formats, [Working with Other Delimited Formats](#)
  - JSON data, [JSON Data](#)
  - missing data, [DataFrame](#)
  - other delimited formats, [Working with Other Delimited Formats](#)

- reading in pieces, [Reading Text Files in Pieces](#)
- type inference, [Reading and Writing Data in Text Format](#)
- XML and HTML, [XML and HTML: Web Scraping](#)
- writing text data
  - CSV files, [Writing Data to Text Format](#)
  - JSON data, [JSON Data](#)
  - missing data, [Writing Data to Text Format](#)
  - other delimited format, [Working with Other Delimited Formats](#)
  - subset of columns, [Writing Data to Text Format](#)
- fillna() to fill in missing data, [Handling Missing Data](#), [Filling In Missing Data](#)-[Filling In Missing Data](#)
  - arguments, [Filling In Missing Data](#)
  - filling values with mean, [Example: Filling Missing Values with Group-Specific Values](#)-[Example: Filling Missing Values with Group-Specific Values](#)
  - resampling, [Upsampling and Interpolation](#)
- filtering out missing data, [Filtering Out Missing Data](#)-[Filtering Out Missing Data](#)
- find() in substring, [Python Built-In String Object Methods](#)
- fiscal years, [Period Frequency Conversion](#)
  - fiscal year end for quarterly periods, [Quarterly Period Frequencies](#)
- fixed frequency time series, [Time Series](#)
- fixed period time series, [Time Series](#)
- float scalar type, [Scalar Types](#)
  - scientific notation, [Numeric types](#)
  - type casting, [Type casting](#)
- Float32Dtype extension data type, [Extension Data Types](#)
- Float64Dtype extension data type, [Extension Data Types](#)
- floor (//), [Binary operators and comparisons](#), [Numeric types](#)
- Fluent Python (Ramalho), [Python Language Basics](#), [IPython](#), and [Jupyter Notebooks](#)
- flush() I/O buffer, [Files and the Operating System](#)
- for loops, [for loops](#)
  - NumPy array vectorization instead, [NumPy Basics: Arrays and Vectorized Computation](#), [Arithmetic with NumPy Arrays](#), [Array-Oriented Programming with Arrays](#), [Array-Oriented Programming with Arrays](#)

- performance tip, [Performance Tips](#)
- formatting strings, [Strings](#)
  - datetime type to string, [Dates and times](#)
  - documentation online, [Strings](#)
  - f-strings, [Strings](#)
- FORTRAN language
  - about legacy libraries, [Python as Glue](#)
  - NumPy arrays, [NumPy](#), [NumPy Basics: Arrays and Vectorized Computation](#), [Data Types for ndarrays](#)
  - performance tip, [Performance Tips](#)
- frequencies in periods, [Periods and Period Arithmetic](#)
  - quarterly period frequencies, [Quarterly Period Frequencies](#)
  - resampling and frequency conversion, [Resampling and Frequency Conversion](#)
    - downsampling, [Downsampling-Open-high-low-close \(OHLC\) resampling](#)
- frequencies in time series chart, [Generating Date Ranges](#)
- frequency table via crosstab(), [Bar Plots](#)
- fromfile() (NumPy), [Why Use Structured Arrays?](#)
- frompyfunc() (NumPy), [Writing New ufuncs in Python](#)
- from\_codes() for Categorical, [Categorical Extension Type in pandas](#)
- functions, [Namespaces, Scope, and Local Functions](#)
  - about, [Functions](#)
  - anonymous (lambda), [Anonymous \(Lambda\) Functions](#)
  - arguments, [Function and object method calls](#), [Functions](#)
    - dynamic references, strong types, [Dynamic references, strong types](#)
    - functions as, [Functions Are Objects](#)
    - keyword arguments, [Functions](#)
    - None as default value, [None](#)
    - positional arguments, [Functions](#)
  - calling, [Function and object method calls](#)
  - declaring with def, [Functions](#)
  - errors and exception handling, [Errors and Exception Handling-Exceptions in IPython](#)
  - generators, [Generators](#)
    - generator expressions, [Generator expressions](#)

- grouping via, [Grouping with Functions](#)
  - aggregating data, [Column-Wise and Multiple Function Application](#)
- methods of Python objects, [Function and object method calls, Attributes and methods](#)
- `_name_` attribute, [Column-Wise and Multiple Function Application](#)
- namespaces, [Namespaces, Scope, and Local Functions](#)
- profiling line by line, [Profiling a Function Line by Line-Profiling a Function Line by Line](#)
- Python into NumPy via frompyfunc(), [Writing New ufuncs in Python](#)
- Python objects, [Everything is an object](#), [Functions Are Objects](#)
- return keyword optional, [Functions](#)
- returning multiple values, [Returning Multiple Values](#)
- yield instead of return, [Generators](#)
- Fundamentals of Data Visualization (Wilke), [Other Python Visualization Tools](#)

## G

- generators, [Generators](#)
  - generator expressions, [Generator expressions](#)
  - itertools module generator collection, [itertools module](#)
  - reversed() as, [reversed](#)
- Géron, Aurélien, [Conclusion](#)
- get() for HTTP request, [Interacting with Web APIs](#)
- get() for string element, [String Functions in pandas](#)
- getattr(), [Attributes and methods](#)
- getdefaultencoding() (sys module), [Files and the Operating System](#)
- get\_dummies(), [Computing Indicator/Dummy Variables](#), [Interfacing Between pandas and Model Code](#)
- Gitee for datasets, [Data Analysis Examples](#)
- GitHub
  - alternate site Gitee, [Data Analysis Examples](#)
  - book materials
    - datasets, [Data for Examples](#)

- datasets for last chapter, [Data Analysis Examples](#)
- Jupyter code examples, [IPython and Jupyter](#)
- get() HTTP request, [Interacting with Web APIs](#)
- global interpreter lock (GIL), [Why Not Python?](#)
- global variables
  - caution against using, [Namespaces, Scope, and Local Functions](#)
  - function scope, [Namespaces, Scope, and Local Functions](#)
- GNU/Linux Miniconda installation, [GNU/Linux](#)
  - exit() or Ctrl-D to exit Python shell, [GNU/Linux, The Python Interpreter](#)
- greater than (>) operator, [Binary operators and comparisons](#)
  - set elements, [Set](#)
- group operations
  - about, [Data Aggregation and Group Operations](#)
  - cross-tabulations, [Cross-Tabulations: Crosstab](#)
  - examples
    - group-wise linear regression, [Example: Group-Wise Linear Regression](#)
    - missing data replacement, [Example: Filling Missing Values with Group-Specific Values-Example: Filling Missing Values with Group-Specific Values](#)
    - random sampling and permutations, [Example: Random Sampling and Permutation](#)
    - weighted average and correlation, [Example: Group Weighted Average and Correlation-Example: Group Weighted Average and Correlation](#)
  - group keys suppressed, [Suppressing the Group Keys](#)
  - how to think about, [How to Think About Group Operations-Grouping by Index Levels](#)
    - dictionaries and Series for grouping, [Grouping with Dictionaries and Series](#)
    - functions for grouping, [Grouping with Functions](#)
    - group aggregation example, [How to Think About Group Operations](#)
    - index levels for grouping, [Grouping by Index Levels](#)
    - iterating over groups, [Iterating over Groups](#)
    - missing values excluded, [How to Think About Group Operations](#)

- selecting columns, [Selecting a Column or Subset of Columns](#)
- pivot tables, [Pivot Tables and Cross-Tabulation](#)
  - about, [Pivot Tables and Cross-Tabulation](#)
  - cross-tabulations, [Cross-Tabulations: Crosstab](#)
  - hierarchical indexing, [Hierarchical Indexing](#)
- split-apply-combine, [How to Think About Group Operations](#), [Apply: General split-apply-combine](#)
- transform(), [Group Transforms and “Unwrapped” GroupBys](#)
- unwrapped group operations, [Group Transforms and “Unwrapped” GroupBys](#)
- groupby() (itertools), [itertools module](#), [How to Think About Group Operations](#)
  - aggregations chart, [Data Aggregation](#)
  - apply(), [Apply: General split-apply-combine](#)
  - cut() and qcut() with, [Quantile and Bucket Analysis-Quantile and Bucket Analysis](#)
  - date offsets with, [Shifting dates with offsets](#)
  - group keys suppressed, [Suppressing the Group Keys](#)
  - GroupBy object, [How to Think About Group Operations](#)
    - indexing with column name(s), [Selecting a Column or Subset of Columns](#)
  - grouped by key, [Group Transforms and “Unwrapped” GroupBys](#)
  - iterating over groups, [Iterating over Groups](#)
  - level specified, [Summary Statistics by Level](#)
  - nuisance columns excluded, [How to Think About Group Operations](#)
  - size(), [How to Think About Group Operations](#)
  - split-apply-combine, [How to Think About Group Operations](#), [Apply: General split-apply-combine](#)
  - transform(), [Group Transforms and “Unwrapped” GroupBys](#)
- Grus, Joel, [Conclusion](#)
- Guido, Sarah, [Conclusion](#)

## H

- h5py package for HDF5 files, [Using HDF5 Format](#), [Using HDF5 Format](#), [HDF5 and Other Array Storage Options](#)

- Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow (Géron), [Conclusion](#)
- hasattr(), [Attributes and methods](#)
- hash maps, [Dictionary](#)
  - (see also dictionaries)
- hash mark (#) for comment, [Comments](#)
- hashability
  - dictionaries, [Valid dictionary key types](#)
  - hash() for, [Valid dictionary key types](#)
  - set elements, [Set](#)
- HDF5 binary file format, [Using HDF5 Format](#)-[Using HDF5 Format](#), [HDF5 and Other Array Storage Options](#)
- HDFStore(), [Using HDF5 Format](#)
  - fixed versus table storage schemas, [Using HDF5 Format](#)
- header row in text data files, [Reading and Writing Data in Text Format](#), [Working with Other Delimited Formats](#)
- hierarchical indexing, [Hierarchical Indexing](#)-[Indexing with a DataFrame's columns](#)
  - about, [Hierarchical Indexing](#)
  - about DataFrames, [DataFrame](#)
  - indexing with columns, [Indexing with a DataFrame's columns](#)
  - MultiIndex, [Hierarchical Indexing](#)
    - created by itself then reused, [Hierarchical Indexing](#)
  - names for levels, [Hierarchical Indexing](#)
  - number of levels attribute, [Hierarchical Indexing](#)
  - partial indexing made possible, [Hierarchical Indexing](#)
  - reading text data from a file, [Reading and Writing Data in Text Format](#)
- reordering and sorting levels, [Reordering and Sorting Levels](#)
- reshaping data, [Hierarchical Indexing](#), [Reshaping with Hierarchical Indexing](#)-[Reshaping with Hierarchical Indexing](#)
- summary statistics by level, [Summary Statistics by Level](#)
- hist() (matplotlib), [Figures and Subplots](#)
- histograms and density plots, [Histograms and Density Plots](#)-[Histograms and Density Plots](#)
- histplot() (seaborn), [Histograms and Density Plots](#)
- Hour0, [Frequencies and Date Offsets](#)

- hstack() (NumPy), [Concatenating and Splitting Arrays](#)
- HTML file format, [XML and HTML: Web Scraping](#)
  - reading, [XML and HTML: Web Scraping](#)
- Hugunin, Jim, [NumPy Basics: Arrays and Vectorized Computation](#)
- Hunter, John D., [matplotlib, Plotting and Visualization](#)

## I

- IDEs (integrated development environments), [Integrated Development Environments and Text Editors](#)
  - available IDEs, [Integrated Development Environments and Text Editors](#)
- if, elif, else, [if, elif, and else](#)
  - NumPy array vectorized version, [Expressing Conditional Logic as Array Operations](#)
- iloc operator
  - DataFrame indexing, [Selection on DataFrame with loc and iloc](#)
  - Series indexing, [Indexing, Selection, and Filtering](#)
  - square brackets ([ ]), [Indexing, Selection, and Filtering](#), [Indexing, Selection, and Filtering](#)
- immutable and mutable objects, [Mutable and immutable objects](#)
  - datetime types immutable, [Dates and times](#)
  - Index objects immutable, [Index Objects](#)
  - lists mutable, [List](#)
  - set elements generally immutable, [Set](#)
  - strings immutable, [Strings](#)
  - tuples themselves immutable, [Tuple](#)
- implicit casting and conversions, [Dynamic references, strong types](#)
- importing data (see reading data from a file)
- importing modules, [Imports](#)
  - import csv, [Working with Other Delimited Formats](#)
  - import datetime, [Date and Time Data Types and Tools](#)
  - import matplotlib.pyplot as plt, [A Brief matplotlib API Primer](#)
  - import numpy as np, [Import Conventions](#), [The NumPy ndarray: A Multidimensional Array Object](#), [Getting Started with pandas](#), [Data Aggregation and Group Operations](#)
  - import os, [Using HDF5 Format](#)

- import pandas as pd, [Import Conventions](#), [Getting Started with pandas](#), [Data Aggregation and Group Operations](#)
  - import Series, DataFrame, [Getting Started with pandas](#)
- import patsy, [Creating Model Descriptions with Patsy](#)
- import pytz, [Time Zone Handling](#)
- import requests, [Interacting with Web APIs](#)
- import seaborn as sns, [Import Conventions](#), [Bar Plots](#)
- import statsmodels.api as sm, [Import Conventions](#), [Example: Group-Wise Linear Regression](#), [Estimating Linear Models](#)
- import statsmodels.formula.api as smf, [Estimating Linear Models](#)
- indentation in Python, [Indentation, not braces](#)
- index levels for grouping, [Grouping by Index Levels](#)
- Index objects (pandas), [Index Objects](#)
  - map() to transform data, [Renaming Axis Indexes](#)
  - set methods available, [Index Objects](#)
- index() of substring, [Python Built-In String Object Methods](#)
- inner join of merged data, [Database-Style DataFrame Joins](#)
- installation and setup of Python
  - about, [Installation and Setup](#)
  - about Miniconda, [Installation and Setup](#), [GNU/Linux](#)
  - GNU/Linux, [GNU/Linux](#)
  - macOS, [Miniconda on macOS](#)
  - necessary packages, [Installing Necessary Packages](#)
  - Windows, [Miniconda on Windows](#)
- int scalar type, [Scalar Types](#)
  - integer division, [Numeric types](#)
  - NumPy signed and unsigned, [Data Types for ndarrays](#)
  - range(), [range](#)
  - type casting, [Type casting](#)
- Int16Dtype extension data type, [Extension Data Types](#)
- Int32Dtype extension data type, [Extension Data Types](#)
- Int64Dtype extension data type, [Extension Data Types](#)
- Int8Dtype extension data type, [Extension Data Types](#)
- integrated development environments (IDEs), [Integrated Development Environments and Text Editors](#)
  - available IDEs, [Integrated Development Environments and Text Editors](#)

- interactive debugger in IPython, [Interactive Debugger-Other ways to use the debugger](#)
- interpreted Python language
  - about the interpreter, [The Python Interpreter](#)
  - global interpreter lock, [Why Not Python?](#)
  - invoking via “python”, [The Python Interpreter](#)
    - GNU/Linux, [GNU/Linux](#)
    - IPython, [The Python Interpreter](#)
    - macOS, [Miniconda on macOS](#)
    - Windows, [Miniconda on Windows](#)
  - IPython project, [IPython and Jupyter](#)
    - (see also IPython)
    - prompt, [The Python Interpreter](#)
- intersection(), [Set](#)
  - DataFrame method, [Index Objects](#)
- intersection\_update(), [Set](#)
- intervals
  - open versus closed, [Discretization and Binning](#)
    - half-open, [Downsampling](#)
  - time intervals, [Time Series](#)
- Introduction to Machine Learning with Python (Müller and Guido), [Conclusion](#)
- introspection in IPython, [Introspection](#)
- IPython
  - about, [IPython and Jupyter](#)
  - advanced features, [Advanced IPython Features](#)
  - basics, [Running the IPython Shell](#)
    - tab completion, [Tab Completion, Attributes and methods](#)
  - command history, [Using the Command History-Input and Output Variables](#)
    - input and output variables, [Input and Output Variables](#)
    - searching and reusing, [Searching and Reusing the Command History](#)
  - configuration, [Profiles and Configuration](#)
  - DataFrame access, [DataFrame](#)
  - development environment, [Integrated Development Environments and Text Editors](#)

- (see also software development tools)
- documentation link, [Tab Completion](#)
- exceptions, [Exceptions in IPython](#)
- executing code from clipboard, [Executing Code from the Clipboard](#)
- introspection, [Introspection](#)
- invoking, [Running the IPython Shell](#)
- keyboard shortcuts, [Terminal Keyboard Shortcuts](#)
- magic commands, [About Magic Commands](#)-[About Magic Commands](#)
  - %alias, [Shell Commands and Aliases](#)
  - %bookmark, [Directory Bookmark System](#)
  - Ctrl-C to interrupt running code, [Interrupting running code](#)
  - %debug, [Interactive Debugger](#)-[Other ways to use the debugger](#)
  - executing code from clipboard, [Executing Code from the Clipboard](#)
  - %lprun, [Profiling a Function Line by Line](#)-[Profiling a Function Line by Line](#)
  - operating system commands, [Interacting with the Operating System](#)
  - %prun, [Basic Profiling: %prun and %run -p](#)-[Basic Profiling: %prun and %run -p](#)
  - %run, [The Python Interpreter](#), [The %run Command](#)
- operating system commands, [Interacting with the Operating System](#)
  - directory bookmark system, [Directory Bookmark System](#)
  - shell commands and aliases, [Shell Commands and Aliases](#)
- profiles, [Profiles and Configuration](#)
- prompt, [The Python Interpreter](#)
- software development tools
  - about, [Software Development Tools](#)
  - debugger, [Interactive Debugger](#)-[Other ways to use the debugger](#)
  - measuring execution time, [Timing Code: %time and %timeit](#)-[Timing Code: %time and %timeit](#)
  - profiling code, [Basic Profiling: %prun and %run -p](#)-[Basic Profiling: %prun and %run -p](#)
  - profiling function line by line, [Profiling a Function Line by Line](#)-[Profiling a Function Line by Line](#)

- tips for productive development, [Tips for Productive Code Development Using IPython](#)
- irregular time series, [Time Series](#)
- is operator, [Binary operators and comparisons](#)
  - test for None, [Binary operators and comparisons](#)
- isdisjoint(), [Set](#)
- isinstance(), [Dynamic references, strong types](#)
- isna() to detect NaN, [Series](#), [Handling Missing Data](#), [Handling Missing Data](#)
- ISO 8601 date format parsed, [Converting Between String and Datetime](#)
- issubset(), [Set](#)
- issuperset(), [Set](#)
- is\_unique() property of indexes, [Axis Indexes with Duplicate Labels](#)
- iterators
  - dictionary keys and values, [Dictionary](#)
  - duck typing to verify, [Duck typing](#)
  - for loops for unpacking, [for loops](#)
  - generators, [Generators](#)
  - groupby() object, [Iterating over Groups](#)
  - list(), [List](#)
  - range(), [range](#)
  - read\_csv(), [Reading and Writing Data in Text Format](#)
    - TextFileReader object, [Reading Text Files in Pieces](#)
  - tuple(), [Tuple](#)
- itertools module generator collection, [itertools module](#)
  - chain(), [itertools module](#)
  - combinations(), [itertools module](#)
  - documentation online, [itertools module](#)
  - groupby(), [itertools module](#)
    - (see also groupby())
  - permutations(), [itertools module](#)
  - product(), [itertools module](#)

## J

- join() for DataFrames, [Merging on Index](#)
  - example of use, [Computing Indicator/Dummy Variables](#)

- join() for string concatenation, [Python Built-In String Object Methods](#)
- joins of merged data, [Database-Style DataFrame Joins-Database-Style DataFrame Joins](#)
  - inner join, [Database-Style DataFrame Joins](#)
  - join() for DataFrames, [Merging on Index](#)
    - example of use, [Computing Indicator/Dummy Variables](#)
  - left and right joins, [Database-Style DataFrame Joins](#)
  - many-to-many, [Database-Style DataFrame Joins](#)
  - many-to-one, [Database-Style DataFrame Joins](#)
  - merge key(s) in index, [Merging on Index-Merging on Index](#)
  - outer join, [Database-Style DataFrame Joins](#)
  - overlapping column names, [Database-Style DataFrame Joins](#)
- Jones, Brian K., [Python Language Basics, IPython, and Jupyter Notebooks](#)
- JSON data
  - about, [JSON Data](#)
    - valid Python code almost, [JSON Data](#)
  - get() for HTTP, [Interacting with Web APIs](#)
  - json library functions, [JSON Data](#)
  - Python objects to and from JSON, [JSON Data](#)
  - reading into Series or DataFrames, [JSON Data](#)
  - writing from Series or DataFrames, [JSON Data](#)
- Julia programming language, [Solving the “Two-Language” Problem](#)
- Jupyter notebooks
  - about, [IPython and Jupyter](#)
  - about GitHub notebooks, [IPython and Jupyter](#)
  - about IPython project, [IPython and Jupyter](#)
  - basics, [Running the Jupyter Notebook](#)
  - configuration, [Profiles and Configuration](#)
  - development environment, [Integrated Development Environments and Text Editors](#)
  - executing code from clipboard, [Executing Code from the Clipboard](#)
  - importing a script into a code cell, [The %run Command](#)
  - pandas DataFrame object display, [DataFrame](#)
  - plotting and visualization, [Plotting and Visualization](#)
    - alternatives to Jupyter notebooks, [Plotting and Visualization](#)
    - plotting commands into single cell, [Figures and Subplots](#)

- just-in-time (JIT) compiler technology, [Solving the “Two-Language” Problem](#)

## K

- Kaggle competition dataset, [Introduction to scikit-learn](#)
- kernel density estimate (KDE) plots, [Histograms and Density Plots](#)
- keyboard shortcuts for IPython, [Terminal Keyboard Shortcuts](#)
- KeyboardInterrupt, [Interrupting running code](#)
- keyword arguments, [Functions](#)
- Klein, Adam, [pandas](#)
- Komodo IDE, [Integrated Development Environments and Text Editors](#)

## L

- lambda functions, [Anonymous \(Lambda\) Functions](#)
  - named “<lambda>”, [Column-Wise and Multiple Function Application](#)
- left joins of merged data, [Database-Style DataFrame Joins](#)
- legend() (matplotlib), [Colors, Markers, and Line Styles](#), [Adding legends](#)
- less than (<) operator, [Binary operators and comparisons](#)
  - set elements, [Set](#)
- lexsort() (NumPy), [Indirect Sorts: argsort and lexsort](#)
- libraries
  - essential Python libraries
    - matplotlib, [matplotlib](#)
    - NumPy, [NumPy](#)
    - pandas, [pandas](#)
    - scikit-learn, [scikit-learn](#)
    - SciPy, [SciPy](#)
    - statsmodels, [statsmodels](#)
  - import conventions, [Import Conventions](#)
  - legacy libraries, [Python as Glue](#)
  - Numba for JIT compiler technology, [Solving the “Two-Language” Problem](#)
- line plots
  - matplotlib, [A Brief matplotlib API Primer](#)

- (see also matplotlib)
- pandas, [Line Plots-Line Plots](#)
- linear algebra with NumPy arrays, [Linear Algebra](#)
- linear models
  - group-wise linear regression, [Example: Group-Wise Linear Regression](#)
  - intercept, [Creating Model Descriptions with Patsy](#), [Categorical Data and Patsy](#), [Estimating Linear Models](#)
  - ordinary least squares linear regression, [Estimating Linear Models](#)
  - Patsy, [Creating Model Descriptions with Patsy-Categorical Data and Patsy](#)
    - about Patsy, [Creating Model Descriptions with Patsy](#)
    - DesignMatrix instances, [Creating Model Descriptions with Patsy](#)
    - model metadata in design\_info, [Creating Model Descriptions with Patsy](#)
    - objects into NumPy, [Creating Model Descriptions with Patsy](#)
    - Patsy's formulas, [Creating Model Descriptions with Patsy](#), [Data Transformations in Patsy Formulas](#)
  - statsmodels estimations, [Estimating Linear Models-Estimating Linear Models](#)
- Linux Miniconda installation, [GNU/Linux](#)
  - exit() or Ctrl-D to exit Python shell, [GNU/Linux](#), [The Python Interpreter](#)
- list(), [List](#)
- lists, [List-Slicing](#)
  - adding or removing elements, [Adding and removing elements](#)
    - append() versus insert(), [Adding and removing elements](#)
  - concatenating and combining, [Concatenating and combining lists](#)
  - DataFrame columns, [DataFrame](#)
  - dictionary keys from, [Valid dictionary key types](#)
  - file open(), [Files and the Operating System](#)
  - list comprehensions, [List, Set, and Dictionary Comprehensions](#)
    - nested, [Nested list comprehensions](#)
  - mutable, [List](#)
  - performance of ndarray versus, [NumPy Basics: Arrays and Vectorized Computation](#)
  - range(), [range](#)

- slicing, [Slicing](#)
- sort() in place, [Sorting](#)
- sorted() to new list, [sorted](#)
- strings as, [Strings](#)
- LLVM Project, [Writing Fast NumPy Functions with Numba](#)
- load() ndarray (NumPy), [File Input and Output with Arrays](#)
- loc operator
  - DataFrame indexing, [Selection on DataFrame with loc and iloc](#)
    - modeling, [Interfacing Between pandas and Model Code](#)
  - Series indexing, [Indexing, Selection, and Filtering](#)
  - square brackets ([ ]), [Indexing, Selection, and Filtering](#), [Indexing, Selection, and Filtering](#)
- local namespace, [Namespaces, Scope, and Local Functions](#)
- locale-specific datetime formatting, [Converting Between String and Datetime](#)
- loops
  - for loops, [for loops](#)
  - NumPy array vectorization instead, [NumPy Basics: Arrays and Vectorized Computation](#), [Arithmetic with NumPy Arrays](#), [Array-Oriented Programming with Arrays](#), [Array-Oriented Programming with Arrays](#)
  - performance tip, [Performance Tips](#)
  - while loops, [while loops](#)
- %lprun command, [Profiling a Function Line by Line](#)-[Profiling a Function Line by Line](#)
- lxml, [XML and HTML: Web Scraping](#)
  - objectify parsing XML, [Parsing XML with lxml.objectify](#)

## M

- machine learning toolkit (see scikit-learn)
- macOS Miniconda installation, [Miniconda on macOS](#)
  - exit() or Ctrl-D to exit Python shell, [Miniconda on macOS](#), [The Python Interpreter](#)
- many-to-many joins of merged data, [Database-Style DataFrame Joins](#)
- many-to-one joins of merged data, [Database-Style DataFrame Joins](#)

- map() to transform data, [Transforming Data Using a Function or Mapping](#)
  - axis indexes, [Renaming Axis Indexes](#)
- math operators in Python, [Binary operators and comparisons](#)
- matplotlib
  - about, [matplotlib, Plotting and Visualization, Other Python Visualization Tools](#)
  - API primer, [A Brief matplotlib API Primer-matplotlib Configuration](#)
    - about matplotlib, [A Brief matplotlib API Primer](#)
    - annotations and drawing on subplot, [Annotations and Drawing on a Subplot-Annotations and Drawing on a Subplot](#)
    - colors, markers, line styles, [Colors, Markers, and Line Styles-Colors, Markers, and Line Styles](#)
    - figures and subplots, [Figures and Subplots-Adjusting the spacing around subplots](#)
    - saving plots to file, [Saving Plots to File](#)
    - ticks, labels, legends, [Ticks, Labels, and Legends-Adding legends](#)
  - configuration, [matplotlib Configuration](#)
  - documentation online, [Figures and Subplots](#)
  - invoking, [A Brief matplotlib API Primer](#)
  - patch objects, [Annotations and Drawing on a Subplot](#)
  - plots saved to files, [Saving Plots to File](#)
  - two-dimensional NumPy array, [Array-Oriented Programming with Arrays](#)
- matrix multiplication via dot() (NumPy), [Linear Algebra](#)
- maximum() (NumPy ufunc), [Universal Functions: Fast Element-Wise Array Functions](#)
- mean(), [How to Think About Group Operations](#)
  - grouping by key, [Group Transforms and “Unwrapped” GroupBys](#)
  - missing data replaced with mean, [Example: Filling Missing Values with Group-Specific Values-Example: Filling Missing Values with Group-Specific Values](#)
- pivot table default aggregation, [Pivot Tables and Cross-Tabulation](#)
- median value to fill in missing data, [Introduction to scikit-learn](#)
- melt() multiple columns into one, [Pivoting “Wide” to “Long” Format-Pivoting “Wide” to “Long” Format](#)
- memmap() (NumPy), [Memory-Mapped Files](#)

- memory usage
  - contiguous memory importance, [The Importance of Contiguous Memory](#)
  - generators, [Generators](#)
  - NumPy ndarrays, [NumPy Basics: Arrays and Vectorized Computation](#)
    - row versus column major order, [C Versus FORTRAN Order](#)
    - striding information, [ndarray Object Internals](#)
  - memory-mapped files (NumPy), [Memory-Mapped Files](#)
  - merge() datasets, [Database-Style DataFrame Joins-Database-Style DataFrame Joins](#)
    - merge key(s) in index, [Merging on Index-Merging on Index](#)
  - mergesort parameter for stable sorts, [Alternative Sort Algorithms](#)
  - metadata
    - dtype as, [Creating ndarrays](#), [Data Types for ndarrays](#)
    - model metadata in design\_info, [Creating Model Descriptions with Patsy](#)
    - pandas preserving, [pandas](#)
  - Microsoft Excel files read, [Reading Microsoft Excel Files](#)
  - Miniconda package manager
    - about, [Installation and Setup](#), [GNU/Linux](#)
    - conda to invoke commands (see conda)
    - conda-forge, [Installation and Setup](#)
    - GNU/Linux installation, [GNU/Linux](#)
    - necessary packages installed, [Installing Necessary Packages](#)
    - Windows installation, [Miniconda on Windows](#)
  - minus (-) operator, [Binary operators and comparisons](#)
    - sets, [Set](#)
    - timedelta type, [Dates and times](#)
  - Mirjalili, Vahid, [Conclusion](#)
  - missing data, [Handling Missing Data](#)
    - combine\_first(), [Combining Data with Overlap](#)
    - datetime parsing, [Converting Between String and Datetime](#)
    - filling in, [Filling In Missing Data-Filling In Missing Data](#)
      - pivot tables, [Pivot Tables and Cross-Tabulation](#)
      - with mean value, [Example: Filling Missing Values with Group-Specific Values-Example: Filling Missing Values with Group-](#)

## Specific Values

- with median value, [Introduction to scikit-learn](#)
- filtering out, [Filtering Out Missing Data-Filtering Out Missing Data](#)
- groupby() group key, [How to Think About Group Operations](#)
- introduced during shifting, [Shifting \(Leading and Lagging\) Data](#)
- NA and NULL sentinels (pandas), [Reading and Writing Data in Text Format](#)
- NA for not available, [Reading and Writing Data in Text Format](#), [Handling Missing Data](#)
- NaN (pandas), [Series](#), [Handling Missing Data](#)
  - fillna() to fill in missing data, [Handling Missing Data](#), [Filling In Missing Data-Filling In Missing Data](#)
  - isna() and notna() to detect, [Series](#), [Handling Missing Data](#), [Handling Missing Data](#)
- None, [Scalar Types](#), [None](#), [Handling Missing Data](#)
  - dictionary key not present, [Default values](#)
  - function without return, [Functions](#)
  - is operator testing for, [Binary operators and comparisons](#)
- reading text data from a file, [Reading and Writing Data in Text Format](#)
- scikit-learn not allowing, [Introduction to scikit-learn](#)
- sentinel (placeholder) values, [Reading and Writing Data in Text Format](#), [Writing Data to Text Format](#), [Handling Missing Data](#)
- statsmodels not allowing, [Introduction to scikit-learn](#)
- string data, [String Functions in pandas](#)-[String Functions in pandas](#)
- writing text data to a file, [Writing Data to Text Format](#)
- modeling
  - about, [Introduction to Modeling Libraries in Python](#)
  - data
    - feature engineering, [Interfacing Between pandas and Model Code](#)
    - nonnumeric column, [Interfacing Between pandas and Model Code](#)
    - NumPy arrays for transfer, [Interfacing Between pandas and Model Code](#)
    - pandas for loading and cleaning, [Interfacing Between pandas and Model Code](#)

- intercept, [Creating Model Descriptions with Patsy](#), [Categorical Data and Patsy](#), [Estimating Linear Models](#)
- Patsy for model descriptions, [Creating Model Descriptions with Patsy](#)-[Categorical Data and Patsy](#)
  - about Patsy, [statsmodels](#), [Creating Model Descriptions with Patsy](#)
  - categorical data, [Categorical Data and Patsy](#)-[Categorical Data and Patsy](#)
  - data transformations, [Data Transformations in Patsy Formulas](#)
  - DesignMatrix instances, [Creating Model Descriptions with Patsy](#)
  - model metadata in design\_info, [Creating Model Descriptions with Patsy](#)
  - objects into NumPy, [Creating Model Descriptions with Patsy](#)
  - Patsy's formulas, [Creating Model Descriptions with Patsy](#), [Data Transformations in Patsy Formulas](#)
  - stateful transformations, [Data Transformations in Patsy Formulas](#)
- modf() (NumPy ufunc), [Universal Functions: Fast Element-Wise Array Functions](#)
- modules
  - about, [Imports](#)
  - csv module, [Working with Other Delimited Formats](#)
  - datetime module, [Dates and times](#)-[Dates and times](#)
  - importing, [Imports](#)
    - import matplotlib.pyplot as plt, [A Brief matplotlib API Primer](#)
    - import numpy as np, [Import Conventions](#), [The NumPy ndarray: A Multidimensional Array Object](#), [Getting Started with pandas](#), [Data Aggregation and Group Operations](#)
    - import pandas as pd, [Import Conventions](#), [Getting Started with pandas](#), [Data Aggregation and Group Operations](#)
    - import seaborn as sns, [Import Conventions](#)
    - import Series, DataFrame, [Getting Started with pandas](#)
    - import statsmodels as sm, [Import Conventions](#)
  - itertools module, [itertools module](#)
  - os module, [Using HDF5 Format](#)
  - pickle module, [Binary Data Formats](#)
    - caution about, [Binary Data Formats](#)

- random modules, [Pseudorandom Number Generation](#)
- re for regular expressions, [Regular Expressions](#)
- requests module, [Interacting with Web APIs](#)
- Monte Carlo simulation example, [Example: Random Sampling and Permutation](#)
- MovieLens example dataset, [MovieLens 1M Dataset-Measuring Rating Disagreement](#)
  - measuring rating disagreement, [Measuring Rating Disagreement](#)-[Measuring Rating Disagreement](#)
- moving window functions, [Moving Window Functions](#)-User-Defined [Moving Window Functions](#)
  - binary, [Binary Moving Window Functions](#)
  - decay factor, [Exponentially Weighted Functions](#)
  - expanding window mean, [Moving Window Functions](#)
  - exponentially weighted functions, [Exponentially Weighted Functions](#)
  - rolling operator, [Moving Window Functions](#)
  - span, [Exponentially Weighted Functions](#)
  - user-defined, [User-Defined Moving Window Functions](#)
- Müller, Andreas, [Conclusion](#)
- MultiIndex, [Hierarchical Indexing](#)
  - created by itself then reused, [Hierarchical Indexing](#)
- multithreading and Python, [Why Not Python?](#)
- mutable and immutable objects, [Mutable and immutable objects](#)
  - datetime types immutable, [Dates and times](#)
  - Index objects immutable, [Index Objects](#)
  - lists mutable, [List](#)
  - set elements generally immutable, [Set](#)
  - strings immutable, [Strings](#)
  - tuples themselves immutable, [Tuple](#)

## N

- NA for data not available, [Reading and Writing Data in Text Format](#), [Handling Missing Data](#)
  - (see also null values)
- namespaces

- functions, [Namespaces, Scope, and Local Functions](#)
- importing modules (see importing modules)
- importing Series, DataFrame into local, [Getting Started with pandas](#)
- IPython namespace search, [Introspection](#)
- scripts run in empty namespace, [The %run Command](#)
- NaN (Not a Number; pandas), [Series](#), [Handling Missing Data](#)
  - dropna() filter for missing data, [Handling Missing Data-Filtering Out Missing Data](#)
  - fillna() to fill in missing data, [Handling Missing Data](#), [Filling In Missing Data-Filling In Missing Data](#)
  - isna() and notna() to detect, [Series](#), [Handling Missing Data](#), [Handling Missing Data](#)
  - missing data in file read, [Reading and Writing Data in Text Format](#)
- NaT (Not a Time), [Converting Between String and Datetime](#)
- ndarrays (NumPy)
  - @ infix operator, [Transposing Arrays and Swapping Axes](#)
  - about, [The NumPy ndarray: A Multidimensional Array Object](#)
  - advanced concepts
    - broadcasting, [Arithmetic with NumPy Arrays](#), [Operations between DataFrame and Series](#), [Broadcasting-Setting Array Values by Broadcasting](#)
    - C order versus FORTRAN order, [C Versus FORTRAN Order](#)
    - C versus FORTRAN order, [Reshaping Arrays](#)
    - concatenating arrays, [Concatenating Along an Axis](#)-[Concatenating Along an Axis](#), [Concatenating and Splitting Arrays](#)-[Stacking helpers: r\\_ and c\\_](#)
    - data type hierarchy, [NumPy Data Type Hierarchy](#)
    - fancy indexing equivalents, [Fancy Indexing Equivalents: take and put](#)
    - object internals, [ndarray Object Internals](#)
    - r\_ and c\_ objects, [Stacking helpers: r\\_ and c\\_](#)
    - repeating elements, [Repeating Elements: tile and repeat](#)
    - reshaping arrays, [Advanced Array Manipulation-Reshaping Arrays](#)
    - row major versus column major order, [C Versus FORTRAN Order](#)
    - row versus column major order, [Reshaping Arrays](#)

- sorting, [Sorting](#), [More About Sorting](#)-[numpy.searchsorted](#): [Finding Elements in a Sorted Array](#)
- splitting arrays, [Concatenating and Splitting Arrays](#)
- striding information, [ndarray Object Internals](#)
- structured arrays, [Structured and Record Arrays](#)-[Why Use Structured Arrays?](#)
- tiling arrays, [Repeating Elements: tile and repeat](#)
- ufunc methods, [Universal Functions: Fast Element-Wise Array Functions](#), [ufunc Instance Methods](#)-[ufunc Instance Methods](#)
- ufuncs compiled via Numba, [Creating Custom numpy.ufunc Objects with Numba](#)
- ufuncs faster with Numba, [Writing Fast NumPy Functions with Numba](#)
- ufuncs written in Python, [Writing New ufuncs in Python](#)
- arithmetic with, [Arithmetic with NumPy Arrays](#)
- array-oriented programming, [Array-Oriented Programming with Arrays](#)-[Unique and Other Set Logic](#)
  - conditional logic as array operations, [Expressing Conditional Logic as Array Operations](#)
  - random walks, [Example: Random Walks](#)-[Simulating Many Random Walks at Once](#)
  - random walks, many at once, [Simulating Many Random Walks at Once](#)
  - unique and other set logic, [Unique and Other Set Logic](#)
  - vectorization, [Array-Oriented Programming with Arrays](#), [Array-Oriented Programming with Arrays](#)
- basic indexing and slicing, [Basic Indexing and Slicing](#)-[Indexing with slices](#)
- Boolean array methods, [Methods for Boolean Arrays](#)
- Boolean indexing, [Boolean Indexing](#)-[Boolean Indexing](#)
- broadcasting, [Broadcasting](#)-[Setting Array Values by Broadcasting](#)
  - about, [Arithmetic with NumPy Arrays](#), [Operations between DataFrame and Series](#)
  - over other axes, [Broadcasting over Other Axes](#)
  - performance tip, [Performance Tips](#)
  - setting array values via, [Setting Array Values by Broadcasting](#)

- concatenating, [Concatenating Along an Axis](#)-[Concatenating Along an Axis](#), [Concatenating and Splitting Arrays](#)-[Stacking helpers: r\\_ and c\\_](#)
  - r\_ and c\_ objects, [Stacking helpers: r\\_ and c\\_](#)
- creating, [Creating ndarrays](#)-[Creating ndarrays](#)
- data types, [Creating ndarrays](#), [Data Types for ndarrays](#)-[Data Types for ndarrays](#), [ndarray Object Internals](#)
  - hierarchy of, NumPy Data Type Hierarchy
  - type casting, [Data Types for ndarrays](#)
  - ValueError, [Data Types for ndarrays](#)
- dtype property, [The NumPy ndarray: A Multidimensional Array Object](#), [Creating ndarrays](#), [Data Types for ndarrays](#)-[Data Types for ndarrays](#)
- fancy indexing, [Fancy Indexing](#)-[Fancy Indexing](#)
  - take() and put(), [Fancy Indexing Equivalents: take and put](#)
- linear algebra, [Linear Algebra](#)
- model data transfer via, [Interfacing Between pandas and Model Code](#)
- Patsy DesignMatrix instances, [Creating Model Descriptions with Patsy](#)
- performance of Python list versus, [NumPy Basics: Arrays and Vectorized Computation](#)
- shape property, [The NumPy ndarray: A Multidimensional Array Object](#)
- sorting, [Sorting](#), [More About Sorting](#)-[numpy.searchsorted: Finding Elements in a Sorted Array](#)
  - alternative sort algorithms, [Alternative Sort Algorithms](#)
  - descending order problem, [More About Sorting](#)
  - indirect sorts, [Indirect Sorts: argsort and lexsort](#)
  - partially sorting, [Partially Sorting Arrays](#)
  - searching sorted arrays, [numpy.searchsorted: Finding Elements in a Sorted Array](#)
- statistical methods, [Mathematical and Statistical Methods](#)
- structured arrays
  - about, [Structured and Record Arrays](#)
  - memory maps working with, [Memory-Mapped Files](#)

- nested data types, [Nested Data Types and Multidimensional Fields](#)
  - why use, [Why Use Structured Arrays?](#)
- swapping axes, [Transposing Arrays and Swapping Axes](#)
- transposing arrays, [Transposing Arrays and Swapping Axes](#)
- ufuncs, [Universal Functions: Fast Element-Wise Array Functions](#)
  - compiled via Numba, [Creating Custom numpy.ufunc Objects with Numba](#)
  - faster with Numba, [Writing Fast NumPy Functions with Numba](#)
  - methods, [Universal Functions: Fast Element-Wise Array Functions, ufunc Instance Methods-ufunc Instance Methods](#)
  - pandas objects and, [Function Application and Mapping](#)
  - Python-written ufuncs, [Writing New ufuncs in Python](#)
- nested list comprehensions, [Nested list comprehensions](#)
- newline character (\n) counted, [Strings](#)
- None, [None, Handling Missing Data](#)
  - about, [Scalar Types](#)
  - dictionary key not present, [Default values](#)
  - function without return, [Functions](#)
  - is operator testing for, [Binary operators and comparisons](#)
- nonlocal variables, [Namespaces, Scope, and Local Functions](#)
- not equal to (!=), [Binary operators and comparisons](#)
- notna() to detect NaN, [Series, Handling Missing Data](#)
  - filtering out missing data, [Filtering Out Missing Data](#)
- np (see NumPy)
- null values
  - combining data with overlap, [Combining Data with Overlap](#)
  - missing data, [Reading and Writing Data in Text Format, Handling Missing Data](#)
    - (see also missing data)
- NaN (pandas), [Series, Handling Missing Data](#)
  - dropna() filter for missing data, [Handling Missing Data-Filtering Out Missing Data](#)
  - fillna() to fill in, [Handling Missing Data, Filling In Missing Data-Filling In Missing Data](#)
  - isna() and notna() to detect, [Series, Handling Missing Data, Handling Missing Data](#)

- missing data in file read, [Reading and Writing Data in Text Format](#)
- NaT (Not a Time; pandas), [Converting Between String and Datetime](#)
- None, [Scalar Types](#), [None](#), [Handling Missing Data](#)
  - dictionary key not present, [Default values](#)
  - function without return, [Functions](#)
  - is operator testing for, [Binary operators and comparisons](#)
- nullable data types, [Extension Data Types](#), [Database-Style DataFrame Joins](#)
- pivot table fill values, [Pivot Tables and Cross-Tabulation](#)
- string data preparation, [String Functions in pandas](#)
- Numba library
  - about, [Writing Fast NumPy Functions with Numba](#)
  - custom compiled NumPy ufuncs, [Creating Custom numpy.ufunc Objects with Numba](#)
  - just-in-time (JIT) compiler technology, [Solving the “Two-Language” Problem](#)
- numeric types, [Numeric types](#)
  - NaN as floating-point value, [Handling Missing Data](#)
  - nullable data types, [Extension Data Types](#), [Database-Style DataFrame Joins](#)
  - NumPy ndarrays
    - data type hierarchy, [NumPy Data Type Hierarchy](#)
    - type casting, [Data Types for ndarrays](#)
- NumPy
  - about, [NumPy](#), [NumPy Basics: Arrays and Vectorized Computation](#)-  
[NumPy Basics: Arrays and Vectorized Computation](#)
    - shortcomings, [Extension Data Types](#)
  - array-oriented programming, [Array-Oriented Programming with Arrays](#)-[Unique and Other Set Logic](#)
    - conditional logic as array operations, [Expressing Conditional Logic as Array Operations](#)
    - random walks, [Example: Random Walks-Simulating Many Random Walks at Once](#)
    - random walks, many at once, [Simulating Many Random Walks at Once](#)
    - unique and other set logic, [Unique and Other Set Logic](#)

- vectorization, [Array-Oriented Programming with Arrays](#), [Array-Oriented Programming with Arrays](#)
- data types, [Data Types for ndarrays](#)-[Data Types for ndarrays](#)
  - hierarchy of, [NumPy Data Type Hierarchy](#)
  - string\_type caution, [Data Types for ndarrays](#)
  - trailing underscores in names, [NumPy Data Type Hierarchy](#)
  - ValueError, [Data Types for ndarrays](#)
- DataFrames to\_numpy(), [DataFrame](#)
- email list, [Community and Conferences](#)
- import numpy as np, [Import Conventions](#), [The NumPy ndarray: A Multidimensional Array Object](#), [Getting Started with pandas](#), [Data Aggregation and Group Operations](#)
- ndarrays, [The NumPy ndarray: A Multidimensional Array Object](#)
  - (see also ndarrays)
- Patsy objects directly into, [Creating Model Descriptions with Patsy](#)
- permutation of data, [Permutation and Random Sampling](#)
- pseudorandom number generation, [Pseudorandom Number Generation](#)
  - methods available, [Pseudorandom Number Generation](#)
- Python functions via frompyfunc(), [Writing New ufuncs in Python](#)

## O

- object introspection in IPython, [Introspection](#)
- object model of Python, [Everything is an object](#)
  - (see also Python objects)
- OHLC (open-high-low-close) resampling, [Open-high-low-close \(OHLC\) resampling](#)
- Oliphant, Travis, [NumPy Basics: Arrays and Vectorized Computation](#)
- Olson database of time zone information, [Time Zone Handling](#)
- online resources (see resources online)
- open() a file, [Files and the Operating System](#)
  - close() when finished, [Files and the Operating System](#)
  - converting between encodings, [Bytes and Unicode with Files](#)
  - default read only mode, [Files and the Operating System](#)
  - read/write modes, [Files and the Operating System](#)
  - with statement for clean-up, [Files and the Operating System](#)

- write-only modes, [Files and the Operating System](#)
- writing delimited files, [Working with Other Delimited Formats](#)
- open-high-low-close (OHLC) resampling, [Open-high-low-close \(OHLC\) resampling](#)
- operating system via IPython, [Interacting with the Operating System](#)
  - directory bookmark system, [Directory Bookmark System](#)
  - os module, [Using HDF5 Format](#)
  - shell commands and aliases, [Shell Commands and Aliases](#)
- ordinary least squares linear regression, [Estimating Linear Models](#)
- os module to remove HDF5 file, [Using HDF5 Format](#)
- outer join of merged data, [Database-Style DataFrame Joins](#)
- outer() (NumPy ufunc), [ufunc Instance Methods](#)

## P

- package manager Miniconda, [Installation and Setup](#)
  - conda-forge, [Installation and Setup](#)
- pairplot() (seaborn), [Scatter or Point Plots](#)
- pandas
  - about, [pandas](#), [NumPy Basics: Arrays and Vectorized Computation](#), [Getting Started with pandas](#)
    - book coverage, [Getting Started with pandas](#)
    - file input and output, [File Input and Output with Arrays](#)
    - non-numeric data handling, [Data Types for ndarrays](#)
    - time series, [pandas](#)
  - DataFrames, [DataFrame-DataFrame](#)
    - (see also DataFrames)
    - about, [pandas](#), [pandas](#)
    - arithmetic, [Arithmetic and Data Alignment](#)
    - arithmetic with fill values, [Arithmetic methods with fill values](#)
    - arithmetic with Series, [Operations between DataFrame and Series](#)
    - columns retrieved as Series, [DataFrame](#)
    - constructing, [DataFrame](#)
    - hierarchical indexing, [DataFrame](#)
    - importing into local namespace, [Getting Started with pandas](#)
    - Index objects, [Index Objects](#)

- indexes for row and column, [DataFrame](#)
- indexing options chart, [Selection on DataFrame with loc and iloc](#)
- integer indexing pitfalls, [Integer indexing pitfalls](#)
- Jupyter notebook display of, [DataFrame](#)
- objects that have different indexes, [Arithmetic and Data Alignment](#)
- possible data inputs chart, [DataFrame](#)
- reindexing, [Reindexing](#)
- to\_numpy(), [DataFrame](#)
- documentation online, [Time Series, HDF5 and Other Array Storage Options](#)
- import pandas as pd, [Import Conventions, Getting Started with pandas, Data Aggregation and Group Operations](#)
  - import Series, DataFrame, [Getting Started with pandas](#)
- Index objects, [Index Objects](#)
  - set methods available, [Index Objects](#)
- missing data representations, [Handling Missing Data](#)
  - (see also missing data)
- modeling with, [Introduction to Modeling Libraries in Python](#)
  - (see also modeling)
- NaN for missing or NA values, [Series, Handling Missing Data](#)
  - dropna() filter for missing data, [Handling Missing Data-Filtering Out Missing Data](#)
  - fillna() to fill in missing data, [Handling Missing Data, Filling In Missing Data-Filling In Missing Data](#)
  - isna() and notna() to detect, [Series, Handling Missing Data, Handling Missing Data](#)
  - missing data in file read, [Reading and Writing Data in Text Format](#)
- Series, [Series-Series](#)
  - (see also Series)
  - about, [pandas](#)
  - arithmetic, [Arithmetic and Data Alignment](#)
  - arithmetic methods chart, [Arithmetic methods with fill values](#)
  - arithmetic with DataFrames, [Operations between DataFrame and Series](#)
  - arithmetic with fill values, [Arithmetic methods with fill values](#)

- array attribute, [Series](#)
- DataFrame column retrieved as, [DataFrame](#)
- importing into local namespace, [Getting Started with pandas](#)
- index, [Series](#)
- index attribute, [Series](#)
- Index objects, [Index Objects](#)
- indexing, [Indexing, Selection, and Filtering](#)
- integer indexing pitfalls, [Integer indexing pitfalls](#)
- name attribute, [Series](#)
- NumPy ufuncs and, [Function Application and Mapping](#)
- objects that have different indexes, [Arithmetic and Data Alignment](#)
- PandasArray, [Series](#)
- reindexing, [Reindexing](#)
- statistical methods
  - correlation and covariance, [Correlation and Covariance-Correlation and Covariance](#)
  - summary statistics, [Summarizing and Computing Descriptive Statistics-Summarizing and Computing Descriptive Statistics](#)
  - summary statistics by level, [Summary Statistics by Level](#)
- unique and other set logic, [Unique Values, Value Counts, and Membership-Unique Values, Value Counts, and Membership](#)
- PandasArray, [Series](#)
- parentheses ()
  - calling functions and object methods, [Function and object method calls](#)
  - intervals open (exclusive), [Discretization and Binning](#)
  - method called on results, [Detecting and Filtering Outliers](#)
  - tuples, [Tuple-Tuple methods](#)
    - tuples of exception types, [Errors and Exception Handling](#)
- Parquet (Apache)
  - read\_parquet(), [Binary Data Formats](#)
  - remote servers for processing data, [Using HDF5 Format](#)
- parsing a text file, [Data Loading, Storage, and File Formats-Reading and Writing Data in Text Format](#)
  - HTML, [XML and HTML: Web Scraping](#)
  - JSON, [JSON Data](#)

- XML with lxml.objectify, [Parsing XML with lxml.objectify](#)
- parsing date format , [Converting Between String and Datetime](#)
- pass statement, [pass](#)
- passenger survival scikit-learn example, [Introduction to scikit-learn-Introduction to scikit-learn](#)
- patch objects in matplotlib, [Annotations and Drawing on a Subplot](#)
- PATH and invoking Miniconda, [Miniconda on Windows](#)
- Patsy, [Creating Model Descriptions with Patsy-Categorical Data and Patsy](#)
  - about, [statsmodels](#), [Creating Model Descriptions with Patsy](#)
    - Intercept, [Creating Model Descriptions with Patsy](#), [Categorical Data and Patsy](#), [Estimating Linear Models](#)
  - DesignMatrix instances, [Creating Model Descriptions with Patsy](#)
  - model metadata in design\_info, [Creating Model Descriptions with Patsy](#)
  - objects into NumPy, [Creating Model Descriptions with Patsy](#)
  - Patsy's formulas, [Creating Model Descriptions with Patsy](#), [Data Transformations in Patsy Formulas](#)
    - categorical data, [Categorical Data and Patsy-Categorical Data and Patsy](#)
    - data transformations, [Data Transformations in Patsy Formulas](#)
    - stateful transformations, [Data Transformations in Patsy Formulas](#)
- pd (see pandas)
- pdb (see debugger in IPython)
- percent (%)
  - datetime formatting, [Converting Between String and Datetime](#)
  - IPython magic commands, [About Magic Commands](#)-[About Magic Commands](#)
    - %alias, [Shell Commands and Aliases](#)
    - %bookmark, [Directory Bookmark System](#)
    - Ctrl-C to interrupt running code, [Interrupting running code](#)
    - %debug, [Interactive Debugger](#)-[Other ways to use the debugger](#)
    - executing code from clipboard, [Executing Code from the Clipboard](#)
    - %lprun, [Profiling a Function Line by Line](#)-[Profiling a Function Line by Line](#)

- operating system commands, [Interacting with the Operating System](#)
- %prun, [Basic Profiling: %prun and %run -p-Basic Profiling: %prun and %run -p](#)
- %run, [The Python Interpreter, The %run Command](#)
- %run -p, [Basic Profiling: %prun and %run -p-Basic Profiling: %prun and %run -p](#)
- %time and %timeit, [Timing Code: %time and %timeit-Timing Code: %time and %timeit](#)
- Pérez, Fernando, [IPython and Jupyter](#)
- performance
  - aggregation functions, [Data Aggregation, Group Transforms and “Unwrapped” GroupBys](#)
  - categorical data computations, [Better performance with categoricals](#)
  - NumPy ndarray versus Python list, [NumPy Basics: Arrays and Vectorized Computation](#)
  - Python ufuncs via NumPy, [Writing New ufuncs in Python](#)
    - faster with Numba, [Writing Fast NumPy Functions with Numba](#)
  - sort\_index() data selection, [Reordering and Sorting Levels](#)
  - tips for, [Performance Tips](#)
    - contiguous memory importance, [The Importance of Contiguous Memory](#)
  - value\_counts() on categoricals, [Better performance with categoricals](#)
- Period object, [Periods and Period Arithmetic](#)
  - converted to another frequency, [Period Frequency Conversion](#)
  - converting timestamps to and from, [Converting Timestamps to Periods \(and Back\)](#)
  - quarterly period frequencies, [Quarterly Period Frequencies](#)
- PeriodIndex object, [Periods and Period Arithmetic](#)
  - converted to another frequency, [Period Frequency Conversion, Period Frequency Conversion](#)
  - creating from arrays, [Creating a PeriodIndex from Arrays](#)
- PeriodIndex(), [Pivoting “Long” to “Wide” Format](#)
- periods
  - about, [Periods and Period Arithmetic](#)

- period frequency conversion, [Period Frequency Conversion](#)
- resampling with, [Resampling with Periods](#)
- period\_range(), [Periods and Period Arithmetic](#)
- Perktold, Josef, [statsmodels](#)
- permutation of data, [Permutation and Random Sampling](#)
  - example, [Example: Random Sampling and Permutation](#)
  - itertools function, [itertools module](#)
  - NumPy random generator method, [Pseudorandom Number Generation](#)
- permutations() (itertools), [itertools module](#)
- pickle module, [Binary Data Formats](#)
  - read\_pickle(), [Correlation and Covariance, Binary Data Formats](#)
  - to\_pickle(), [Binary Data Formats](#)
    - caution about long-term storage, [Binary Data Formats](#)
- pip
  - install, [Installing Necessary Packages](#)
    - conda install recommended, [Installing Necessary Packages](#)
  - upgrade flag, [Installing Necessary Packages](#)
- pipe () for OR, [Binary operators and comparisons](#)
  - NumPy ndarrays, [Boolean Indexing](#)
- pivot tables, [Pivot Tables and Cross-Tabulation-Pivot Tables and Cross-Tabulation](#)
  - about, [Pivot Tables and Cross-Tabulation](#)
  - cross-tabulations, [Cross-Tabulations: Crosstab](#)
  - default aggregation mean(), [Pivot Tables and Cross-Tabulation](#)
    - aggfunc keyword for other, [Pivot Tables and Cross-Tabulation](#)
  - fill value for NA entries, [Pivot Tables and Cross-Tabulation](#)
  - hierarchical indexing, [Hierarchical Indexing](#)
  - margins, [Pivot Tables and Cross-Tabulation](#)
  - pivot tables, [Pivot Tables and Cross-Tabulation](#)
- pivot(), [Pivoting “Long” to “Wide” Format](#)
- pivot\_table(), [Pivot Tables and Cross-Tabulation](#)
  - options chart, [Pivot Tables and Cross-Tabulation](#)
- plot() (matplotlib), [Figures and Subplots](#)
  - colors and line styles, [Colors, Markers, and Line Styles-Colors, Markers, and Line Styles](#)
- plot() (pandas objects)

- bar plots, [Bar Plots-Bar Plots](#)
  - value\_counts() for, [Bar Plots](#)
- density plots, [Histograms and Density Plots-Histograms and Density Plots](#)
- histograms, [Histograms and Density Plots-Histograms and Density Plots](#)
- line plots, [Line Plots-Line Plots](#)
- Plotly for visualization, [Other Python Visualization Tools](#)
- plotting
  - about, [Plotting and Visualization](#)
  - book on data visualization, [Other Python Visualization Tools](#)
  - matplotlib
    - about, [matplotlib](#), [Plotting and Visualization](#), [Other Python Visualization Tools](#)
    - API primer, [A Brief matplotlib API Primer](#)
    - configuration, [matplotlib Configuration](#)
    - documentation online, [Figures and Subplots](#)
    - patch objects, [Annotations and Drawing on a Subplot](#)
    - plots saved to files, [Saving Plots to File](#)
    - two-dimensional NumPy array, [Array-Oriented Programming with Arrays](#)
  - other Python tools, [Other Python Visualization Tools](#)
  - seaborn and pandas, [Plotting with pandas and seaborn-Facet Grids and Categorical Data](#)
    - about seaborn, [Plotting and Visualization](#), [Plotting with pandas and seaborn](#), [Bar Plots](#)
    - bar plots via pandas, [Bar Plots-Bar Plots](#)
    - bar plots with seaborn, [Bar Plots](#)
    - box plots, [Facet Grids and Categorical Data](#)
    - density plots, [Histograms and Density Plots-Histograms and Density Plots](#)
    - documentation online, [Facet Grids and Categorical Data](#)
    - facet grids, [Facet Grids and Categorical Data-Facet Grids and Categorical Data](#)
    - histograms, [Histograms and Density Plots-Histograms and Density Plots](#)
    - import seaborn as sns, [Import Conventions](#), [Bar Plots](#)

- line plots, [Line Plots-Line Plots](#)
- scatter or point plots, [Scatter or Point Plots-Scatter or Point Plots](#)
- plus (+) operator, [Binary operators and comparisons](#)
  - adding strings, [Strings](#)
  - lists, [Concatenating and combining lists](#)
  - Patsy's formulas, [Creating Model Descriptions with Patsy](#)
    - I() wrapper for addition, [Data Transformations in Patsy Formulas](#)
  - timedelta type, [Dates and times](#)
  - tuple concatenation, [Tuple](#)
- point or scatter plots, [Scatter or Point Plots-Scatter or Point Plots](#)
- Polygon() (matplotlib), [Annotations and Drawing on a Subplot](#)
- pop() column from DataFrame, [Pivoting “Long” to “Wide” Format](#)
- pound sign (#) for comment, [Comments](#)
- preparation of data (see data preparation)
- product() (itertools), [itertools module](#)
- profiles in IPython, [Profiles and Configuration](#)
- profiling code, [Basic Profiling: %prun and %run -p-Basic Profiling: %prun and %run -p](#)
  - function line by line, [Profiling a Function Line by Line-Profiling a Function Line by Line](#)
- prompt for IPython, [The Python Interpreter](#)
- prompt for Python interpreter, [The Python Interpreter](#)
- %prun, [Basic Profiling: %prun and %run -p-Basic Profiling: %prun and %run -p](#)
- pseudorandom number generation, [Pseudorandom Number Generation](#)
  - methods available, [Pseudorandom Number Generation](#)
- put0 (NumPy), [Fancy Indexing Equivalents: take and put](#)
- pyarrow package for read\_parquet(), [Binary Data Formats](#)
- PyCharm IDE, [Integrated Development Environments and Text Editors](#)
- PyDev IDE, [Integrated Development Environments and Text Editors](#)
- PyTables package for HDF5 files, [Using HDF5 Format, Using HDF5 Format, HDF5 and Other Array Storage Options](#)
- Python
  - about data analysis, [What Is This Book About?](#)
    - Python drawbacks, [Why Not Python?](#)

- Python for, [Why Python for Data Analysis?](#)
- about Python
  - both prototyping and production, [Solving the “Two-Language” Problem](#)
  - legacy libraries and, [Python as Glue](#)
- about version used by book, [Installation and Setup](#)
- community, [Community and Conferences](#)
- conferences, [Community and Conferences](#)
- data structures and sequences
  - dictionaries, [Dictionary-Valid dictionary key types](#)
  - lists, [List-Slicing](#)
  - sequence functions built in, [Built-In Sequence Functions](#)
  - sets, [Set-Set](#)
  - tuples, [Tuple-Tuple methods](#)
- errors and exception handling, [Errors and Exception Handling-Exceptions in IPython](#)
  - (see also errors and exception handling)
- everything is an object, [Everything is an object](#)
- exit() to exit, [The Python Interpreter](#)
  - GNU/Linux, [GNU/Linux](#)
  - macOS, [Miniconda on macOS](#)
  - Windows, [Miniconda on Windows](#)
- files, [Files and the Operating System-Bytes and Unicode with Files](#)
  - (see also files in Python)
- installation and setup
  - about, [Installation and Setup](#)
  - about Miniconda, [Installation and Setup](#), [GNU/Linux](#)
  - GNU/Linux, [GNU/Linux](#)
  - macOS, [Miniconda on macOS](#)
  - necessary packages, [Installing Necessary Packages](#)
  - Windows, [Miniconda on Windows](#)
- interpreted language
  - about the interpreter, [The Python Interpreter](#)
  - global interpreter lock, [Why Not Python?](#)
  - IPython project, [IPython and Jupyter](#)
    - (see also IPython)
  - speed trade-off, [Why Not Python?](#)

- invoking, [The Python Interpreter](#)
  - GNU/Linux, [GNU/Linux](#)
  - IPython, [Running the IPython Shell](#)
  - macOS, [Miniconda on macOS](#)
  - Windows, [Miniconda on Windows](#)
- invoking matplotlib, [A Brief matplotlib API Primer](#)
  - (see also matplotlib)
- JSON objects to and from, [JSON Data](#)
- just-in-time (JIT) compiler technology, [Solving the “Two-Language” Problem](#)
- libraries that are key
  - matplotlib, [matplotlib](#)
  - NumPy, [NumPy](#)
  - pandas, [pandas](#)
  - scikit-learn, [scikit-learn](#)
  - SciPy, [SciPy](#)
  - statsmodels, [statsmodels](#)
- module import conventions, [Import Conventions](#)
  - (see also modules)
- objects, [Everything is an object](#)
  - duck typing, [Duck typing](#)
  - dynamic references, strong types, [Dynamic references, strong types](#)
  - is operator, [Binary operators and comparisons](#)
  - methods, [Function and object method calls](#), [Attributes and methods](#)
  - module imports, [Imports](#)
  - mutable and immutable, [Mutable and immutable objects](#)
  - None tested for, [Binary operators and comparisons](#)
  - object introspection in IPython, [Introspection](#)
  - variables, [Variables and argument passing](#)
- tutorial
  - about additional resources, [Python Language Basics, IPython, and Jupyter Notebooks](#)
  - about experimentation, [Python Language Basics, IPython, and Jupyter Notebooks](#)
  - binary operators, [Binary operators and comparisons](#)

- control flow, [Control Flow-range](#)
- exiting Python shell, [The Python Interpreter](#)
- importing modules, [Imports](#)
- invoking Python, [The Python Interpreter](#)
- IPython basics, [The Python Interpreter](#)
- IPython introspection, [Introspection](#)
- IPython tab completion, [Tab Completion](#), [Attributes and methods](#)
- Jupyter notebook basics, [Running the Jupyter Notebook](#)
- scalars, [Scalar Types-Dates and times](#)
- semantics of Python, [Python Language Basics-Mutable and immutable objects](#)
- ufuncs written in, [Writing New ufuncs in Python](#)
  - custom compiled NumPy ufuncs, [Creating Custom numpy.ufunc Objects with Numba](#)
  - faster with Numba, [Writing Fast NumPy Functions with Numba](#)
- Python Cookbook (Beazley and Jones), [Python Language Basics](#), [IPython](#), and [Jupyter Notebooks](#)
- Python Data Science Handbook (VanderPlas), [Conclusion](#)
- Python Machine Learning (Raschka and Mirjalili), [Conclusion](#)
- Python objects, [Everything is an object](#)
  - attributes, [Attributes and methods](#)
  - converting to strings, [Strings](#)
  - duck typing, [Duck typing](#)
  - functions, [Functions Are Objects](#)
  - is operator, [Binary operators and comparisons](#)
    - test for None, [Binary operators and comparisons](#)
  - key-value pairs of dictionaries, [Dictionary](#)
  - methods, [Function and object method calls](#), [Attributes and methods](#)
  - mutable and immutable, [Mutable and immutable objects](#)
    - datetime types immutable, [Dates and times](#)
    - Index objects immutable, [Index Objects](#)
    - lists mutable, [List](#)
    - set elements generally immutable, [Set](#)
    - strings immutable, [Strings](#)
    - tuples themselves immutable, [Tuple](#)
  - object introspection in IPython, [Introspection](#)

- scalars, [Scalar Types-Dates and times](#)
- to\_numpy() with heterogeneous data, [Interfacing Between pandas and Model Code](#)
- variables, [Variables and argument passing](#)
  - dynamic references, strong types, [Dynamic references, strong types](#)
  - module imports, [Imports](#)
- Python Tools for Visual Studio (Windows), [Integrated Development Environments and Text Editors](#)
- pytz library for time zones, [Time Zone Handling](#)

## Q

- qcut() for data binning per quantiles, [Discretization and Binning](#)
  - groupby() with, [Quantile and Bucket Analysis-Quantile and Bucket Analysis](#)
- quarterly period frequencies, [Quarterly Period Frequencies](#)
- question mark (?)
- namespace search in IPython, [Introspection](#)
- object introspection in IPython, [Introspection](#)
- quote marks
  - multiline strings, [Strings](#)
  - string literals declared, [Strings](#)

## R

- raise\_for\_status() for HTTP errors, [Interacting with Web APIs](#)
- Ramalho, Luciano, [Python Language Basics, IPython, and Jupyter Notebooks](#)
- random modules (NumPy and Python), [Pseudorandom Number Generation](#)
  - book use of np.random, [Advanced NumPy](#)
  - NumPy permutation(), [Permutation and Random Sampling](#)
- random sampling, [Permutation and Random Sampling](#)
  - example, [Example: Random Sampling and Permutation](#)
- random walks via NumPy arrays, [Example: Random Walks-Simulating Many Random Walks at Once](#)

- many at once, [Simulating Many Random Walks at Once](#)
- range(), [range](#)
- Raschka, Sebastian, [Conclusion](#)
- ravel() (NumPy), [C Versus FORTRAN Order](#)
- rc() for matplotlib configuration, [matplotlib Configuration](#)
- re module for regular expressions, [Regular Expressions](#)
- read() a file, [Files and the Operating System](#)
- readable() file, [Files and the Operating System](#)
- reading data from a file
  - about, [Data Loading, Storage, and File Formats](#)
  - binary data
    - about non-pickle formats, [Binary Data Formats](#)
    - HDF5 format, [Using HDF5 Format](#)-[Using HDF5 Format](#), [HDF5 and Other Array Storage Options](#)
    - memory-mapped files, [Memory-Mapped Files](#)
    - Microsoft Excel files, [Reading Microsoft Excel Files](#)
    - pickle format, [Binary Data Formats](#)
    - pickle format caution, [Binary Data Formats](#)
  - CSV Dialect, [Working with Other Delimited Formats](#)
  - database interactions, [Interacting with Databases](#)-[Interacting with Databases](#)
  - fromfile() into ndarray, [Why Use Structured Arrays?](#)
  - text data, [Reading and Writing Data in Text Format](#)-[Reading and Writing Data in Text Format](#)
    - CSV files, [Reading and Writing Data in Text Format](#)-[Reading and Writing Data in Text Format](#)
    - CSV files of other formats, [Working with Other Delimited Formats](#)
    - delimiters separating fields, [Reading and Writing Data in Text Format](#)
    - header row, [Reading and Writing Data in Text Format](#), [Working with Other Delimited Formats](#)
    - JSON, [JSON Data](#)
    - missing data, [DataFrame](#), [Reading and Writing Data in Text Format](#)
    - other delimited formats, [Working with Other Delimited Formats](#)
    - parsing, [Data Loading, Storage, and File Formats](#)

- reading in pieces, [Reading Text Files in Pieces](#)
- type inference, [Reading and Writing Data in Text Format](#)
- XML and HTML, [XML and HTML: Web Scraping](#)
- readlines() of a file, [Files and the Operating System](#)
- read\_\* data loading functions, [Reading and Writing Data in Text Format](#)
- read\_csv(), [Reading and Writing Data in Text Format-Reading and Writing Data in Text Format](#)
  - about Python files, [itertools module](#)
  - arguments commonly used, [Reading and Writing Data in Text Format](#)
  - other delimited formats, [Working with Other Delimited Formats](#)
  - reading in pieces, [Reading Text Files in Pieces](#)
- read\_excel(), [Reading Microsoft Excel Files](#)
- read\_hdf(), [Using HDF5 Format](#)
- read\_html(), [XML and HTML: Web Scraping](#)
- read\_json(), [JSON Data](#)
- read\_parquet(), [Binary Data Formats](#)
- read\_pickle(), [Correlation and Covariance, Binary Data Formats](#)
- read\_sql(), [Interacting with Databases](#)
- read\_xml(), [Parsing XML with lxml.objectify](#)
- Rectangle() (matplotlib), [Annotations and Drawing on a Subplot](#)
- reduce() (NumPy ufunc), [ufunc Instance Methods](#)
- reduceat() (NumPy ufunc), [ufunc Instance Methods](#)
- reduction methods for pandas objects, [Summarizing and Computing Descriptive Statistics-Summarizing and Computing Descriptive Statistics](#)
- regplot() (seaborn), [Scatter or Point Plots](#)
- regular expressions (regex), [Regular Expressions-Regular Expressions](#)
  - data preparation, [String Functions in pandas-String Functions in pandas](#)
  - text file whitespace delimiter, [Reading and Writing Data in Text Format](#)
- reindex() (pandas), [Reindexing](#)
  - arguments, [Reindexing](#)
  - loc operator for, [Reindexing](#)
  - resampling, [Upsampling and Interpolation](#)

- remove() for sets, [Set](#)
- rename() to transform data, [Renaming Axis Indexes](#)
- repeat() (NumPy), [Repeating Elements: tile and repeat](#)
- replace() to transform data, [Replacing Values](#)
  - string data, [Python Built-In String Object Methods](#)
- requests package for web API support, [Interacting with Web APIs](#)
- resample(), [Resampling and Frequency Conversion](#)
  - arguments chart, [Resampling and Frequency Conversion](#)
  - resampling with periods, [Resampling with Periods](#)
- resampling and frequency conversion, [Resampling and Frequency Conversion](#)
  - downsampling, [Downsampling-Open-high-low-close \(OHLC\) resampling](#)
  - grouped time resampling, [Grouped Time Resampling](#)
  - open-high-low-close resampling, [Open-high-low-close \(OHLC\) resampling](#)
  - upsampling, [Upsampling and Interpolation](#), [Upsampling and Interpolation](#)
    - open-high-low-close resampling, [Upsampling and Interpolation](#)
- reshape() (NumPy), [Fancy Indexing](#), [Advanced Array Manipulation-Reshaping Arrays](#)
  - row major versus column major order, [C Versus FORTRAN Order](#)
- resources
  - book on data visualization, [Other Python Visualization Tools](#)
  - books on modeling and data science, [Conclusion](#)
- resources online
  - book on data visualization, [Other Python Visualization Tools](#)
  - IPython documentation, [Tab Completion](#)
  - matplotlib documentation, [Figures and Subplots](#)
  - pandas documentation, [Time Series](#), [HDF5 and Other Array Storage Options](#)
  - Python documentation
    - formatting strings, [Strings](#)
    - itertools functions, [itertools module](#)
  - Python tutorial, [Python Language Basics](#), [IPython](#), and [Jupyter Notebooks](#)
  - seaborn documentation, [Facet Grids and Categorical Data](#)

- visualization tools for Python, [Other Python Visualization Tools](#)
- reversed() sequence, [reversed](#)
  - generator, [reversed](#)
- right joins of merged data, [Database-Style DataFrame Joins](#)
- rolling() in moving window functions, [Moving Window Functions](#)
- %run command, [The Python Interpreter](#), [The %run Command](#)
  - %run -p, [Basic Profiling: %prun and %run -p-Basic Profiling:](#)  
[%prun and %run -p](#)

## S

- sample() for random sampling, [Permutation and Random Sampling](#)
- save() ndarray (NumPy), [File Input and Output with Arrays](#)
- savefig() (matplotlib), [Saving Plots to File](#)
- scalars, [Scalar Types-Dates and times](#)
- scatter or point plots, [Scatter or Point Plots-Scatter or Point Plots](#)
- scatter() (matplotlib), [Figures and Subplots](#)
- scikit-learn, [Introduction to scikit-learn-Introduction to scikit-learn](#)
  - about, [scikit-learn](#), [Introduction to scikit-learn](#)
  - conda install scikit-learn, [Introduction to scikit-learn](#)
  - cross-validation, [Introduction to scikit-learn](#)
  - email list, [Community and Conferences](#)
  - missing data not allowed, [Introduction to scikit-learn](#)
    - filling in missing data, [Introduction to scikit-learn](#)
- SciPy
  - about, [SciPy](#)
  - conda install scipy, [Histograms and Density Plots](#)
  - density plots requiring, [Histograms and Density Plots](#)
  - email list, [Community and Conferences](#)
- scripting languages, [Why Python for Data Analysis?](#)
- scripts via IPython %run command, [The Python Interpreter](#), [The %run Command](#)
  - Ctrl-C to interrupt running code, [Interrupting running code](#)
  - execution time measured, [Timing Code: %time and %timeit](#)
- Seabold, Skipper, [statsmodels](#)
- seaborn and pandas, [Plotting with pandas and seaborn-Facet Grids and Categorical Data](#)

- about seaborn, [Plotting and Visualization](#), [Plotting with pandas and seaborn](#), [Bar Plots](#)
- bar plots via pandas, [Bar Plots-Bar Plots](#)
- bar plots with seaborn, [Bar Plots](#)
- box plots, [Facet Grids and Categorical Data](#)
- density plots, [Histograms and Density Plots-Histograms and Density Plots](#)
- documentation online, [Facet Grids and Categorical Data](#)
- facet grids, [Facet Grids and Categorical Data-Facet Grids and Categorical Data](#)
- histograms, [Histograms and Density Plots](#)
- import seaborn as sns, [Import Conventions](#), [Bar Plots](#)
- line plots via pandas, [Line Plots-Line Plots](#)
- scatter or point plots, [Scatter or Point Plots-Scatter or Point Plots](#)
- `searchsorted()` (NumPy), [numpy.searchsorted: Finding Elements in a Sorted Array](#)
- `seek()` in a file, [Files and the Operating System](#)
- `seekable()` file, [Files and the Operating System](#)
- semicolon (;) for multiple statements, [Indentation, not braces](#)
- sentinel (placeholder) values, [Writing Data to Text Format](#), [Handling Missing Data](#)
  - NA and NULL sentinels (pandas), [Reading and Writing Data in Text Format](#)
- sequences
  - built-in sequence functions, [Built-In Sequence Functions](#)
  - Categorical from, [Categorical Extension Type in pandas](#)
  - dictionaries from, [Creating dictionaries from sequences](#)
  - lists, [List-Slicing](#)
  - `range()`, [range](#)
  - strings as, [Strings](#)
  - tuples, [Tuple-Tuple methods](#)
    - unpacking tuples, [Unpacking tuples](#)
- serializing data, [Binary Data Formats](#)
- Series (pandas), [Series-Series](#)
  - about, [pandas](#)
  - arithmetic, [Arithmetic and Data Alignment](#)
    - with fill values, [Arithmetic methods with fill values](#)

- arithmetic methods chart, [Arithmetic methods with fill values](#)
- arithmetic with DataFrames, [Operations between DataFrame and Series](#)
- array attribute, [Series](#)
  - PandasArray, [Series](#)
- axis indexes with duplicate labels, [Axis Indexes with Duplicate Labels](#)
- concatenating along an axis, [Concatenating Along an Axis](#)-  
[Concatenating Along an Axis](#)
- DataFrame columns, [DataFrame](#)
- dictionary from and to Series, [Series](#)
- dimension tables, [Background and Motivation](#)
- dropping entries from an axis, [Dropping Entries from an Axis](#)
- extension data types, [Extension Data Types](#), [String Functions in pandas](#)
- get\_dummies(), [Computing Indicator/Dummy Variables](#)
- grouping via, [Grouping with Dictionaries and Series](#)
- HDF5 binary data format, [Using HDF5 Format](#)
- importing into local namespace, [Getting Started with pandas](#)
- index, [Series](#)
  - indexing, [Indexing, Selection, and Filtering](#)
  - loc to select index values, [Indexing, Selection, and Filtering](#)
  - reindexing, [Reindexing](#)
- index attribute, [Series](#)
- Index objects, [Index Objects](#)
- integer indexing pitfalls, [Integer indexing pitfalls](#)
- JSON data to and from, [JSON Data](#)
- map() to transform data, [Transforming Data Using a Function or Mapping](#)
- missing data
  - dropna() to filter out, [Filtering Out Missing Data](#)
  - fillna() to fill in, [Filling In Missing Data](#)
- MultiIndex, [Hierarchical Indexing](#)
- name attribute, [Series](#)
- NumPy ufuncs and, [Function Application and Mapping](#)
- objects that have different indexes, [Arithmetic and Data Alignment](#)
- ranking, [Sorting and Ranking](#)

- reading data from a file (see reading data from a file)
- replace() to transform data, [Replacing Values](#)
- sorting, [Sorting and Ranking](#)
- statistical methods
  - correlation and covariance, [Correlation and Covariance-Correlation and Covariance](#)
  - summary statistics, [Summarizing and Computing Descriptive Statistics-Summarizing and Computing Descriptive Statistics](#)
  - summary statistics by level, [Summary Statistics by Level](#)
- string data preparation, [String Functions in pandas-String Functions in pandas](#)
- string methods chart, [String Functions in pandas](#)
- time series, [Time Series Basics](#)
  - (see also time series)
- unique and other set logic, [Unique Values, Value Counts, and Membership-Unique Values, Value Counts, and Membership](#)
- writing data (see writing data to a file)
- set(), [Set](#)
- setattr(), [Attributes and methods](#)
- sets
  - intersection of two, [Set](#)
  - list-like elements to tuples for storage, [Set](#)
  - pandas DataFrame methods, [Index Objects](#)
  - set comprehensions, [List, Set, and Dictionary Comprehensions](#)
  - union of two, [Set](#)
- set\_index() to DataFrame column, [Indexing with a DataFrame's columns](#)
- set\_title() (matplotlib), [Setting the title, axis labels, ticks, and tick labels](#)
- set\_trace(), [Other ways to use the debugger](#)
- set\_xlabel() (matplotlib), [Setting the title, axis labels, ticks, and tick labels](#)
- set\_xlim() (matplotlib), [Annotations and Drawing on a Subplot](#)
- set\_xticklabels() (matplotlib), [Setting the title, axis labels, ticks, and tick labels](#)
- set\_xticks() (matplotlib), [Setting the title, axis labels, ticks, and tick labels](#)
- set\_ylim() (matplotlib), [Annotations and Drawing on a Subplot](#)

- She, Chang, [pandas](#)
- shell commands and aliases, [Shell Commands and Aliases](#)
- shifting data through time, [Shifting \(Leading and Lagging\) Data-Shifted dates with offsets](#)
- side effects, [Mutable and immutable objects](#)
- sign() to test positive or negative, [Detecting and Filtering Outliers](#)
- single quote ()
  - multiline strings, [Strings, Strings](#)
  - string literals declared, [Strings](#)
- size() of groups, [How to Think About Group Operations](#)
- slash (/) division operator, [Binary operators and comparisons](#)
  - floor (//), [Binary operators and comparisons](#), [Numeric types](#)
- Slatkin, Brett, [Python Language Basics, IPython, and Jupyter Notebooks](#)
- slicing strings, [String Functions in pandas](#)
- sm (see statsmodels)
- Smith, Nathaniel, [statsmodels](#)
- sns (see seaborn)
- software development tools
  - about, [Software Development Tools](#)
  - debugger, [Interactive Debugger-Other ways to use the debugger](#)
    - chart of commands, [Interactive Debugger](#)
  - IDEs, [Integrated Development Environments and Text Editors](#)
    - available IDEs, [Integrated Development Environments and Text Editors](#)
- measuring execution time, [Timing Code: %time and %timeit-Timing Code: %time and %timeit](#)
- profiling code, [Basic Profiling: %prun and %run -p-Basic Profiling: %prun and %run -p](#)
  - function line by line, [Profiling a Function Line by Line-Profiling a Function Line by Line](#)
- tips for productive development, [Tips for Productive Code Development Using IPython](#)
- sort() in place, [Sorting](#)
  - NumPy arrays, [Sorting, More About Sorting](#)
    - descending order problem, [More About Sorting](#)
- sorted() to new list, [sorted](#)

- NumPy arrays, [Sorting](#)
- sorting ndarrays, [Sorting](#), [More About Sorting](#)-[numpy.searchsorted](#): [Finding Elements in a Sorted Array](#)
  - alternative sort algorithms, [Alternative Sort Algorithms](#)
  - descending order problem, [More About Sorting](#)
  - indirect sorts, [Indirect Sorts: argsort and lexsort](#)
  - partially sorting, [Partially Sorting Arrays](#)
  - searching sorted arrays, [numpy.searchsorted: Finding Elements in a Sorted Array](#)
- sort\_index() by all or subset of levels, [Reordering and Sorting Levels](#)
  - data selection performance, [Reordering and Sorting Levels](#)
- split() array (NumPy), [Concatenating and Splitting Arrays](#)
- split() string, [Python Built-In String Object Methods](#)
- split-apply-combine group operations, [How to Think About Group Operations](#)
- Spyder IDE, [Integrated Development Environments and Text Editors](#)
- SQL query results into DataFrame, [Interacting with Databases](#)-[Interacting with Databases](#)
- SQLAlchemy project, [Interacting with Databases](#)
- SQLite3 database, [Interacting with Databases](#)
- sqrt() (NumPy ufunc), [Universal Functions: Fast Element-Wise Array Functions](#)
- square brackets ([ ])
  - arrays, [The NumPy ndarray: A Multidimensional Array Object](#)
    - (see also arrays)
  - arrays returned in reverse order, [More About Sorting](#)
    - (see also arrays)
  - intervals closed (inclusive), [Discretization and Binning](#)
  - list definitions, [List](#)
    - slicing lists, [Slicing](#)
  - loc and iloc operators, [Indexing, Selection, and Filtering](#), [Indexing, Selection, and Filtering](#)
  - series indexing, [Indexing, Selection, and Filtering](#)
  - string element index, [String Functions in pandas](#)
  - string slicing, [String Functions in pandas](#)
  - tuple elements, [Tuple](#)
- stable sorting algorithms, [Alternative Sort Algorithms](#)

- stack(), [Hierarchical Indexing](#), [Reshaping with Hierarchical Indexing](#)-  
[Reshaping with Hierarchical Indexing](#), [Pivoting “Long” to “Wide” Format](#)
- stacking, [Concatenating Along an Axis](#)-[Concatenating Along an Axis](#),  
[Concatenating and Splitting Arrays](#)-[Stacking helpers: r\\_ and c\\_](#)
  - r\_ and c\_ objects, [Stacking helpers: r\\_ and c\\_](#)
  - vstack() and hstack(), [Concatenating and Splitting Arrays](#)
- standardize() (Patsy), [Data Transformations in Patsy Formulas](#)
- statistical methods
  - categorical variable into dummy matrix, [Computing Indicator/Dummy Variables](#)
  - frequency table via crosstab(), [Bar Plots](#)
  - group weighted average and correlation, [Example: Group Weighted Average and Correlation](#)-[Example: Group Weighted Average and Correlation](#)
  - group-wise linear regression, [Example: Group-Wise Linear Regression](#)
  - groupby() (see groupby())
  - histogram of bimodal distribution, [Histograms and Density Plots](#)
  - mean(), [How to Think About Group Operations](#)
    - grouping by key, [Group Transforms and “Unwrapped” GroupBys](#)
    - missing data replaced with mean, [Example: Filling Missing Values with Group-Specific Values](#)-[Example: Filling Missing Values with Group-Specific Values](#)
    - pivot table default aggregation, [Pivot Tables and Cross-Tabulation](#)
  - moving window functions, [Moving Window Functions](#)-[User-Defined Moving Window Functions](#)
    - binary, [Binary Moving Window Functions](#)
    - decay factor, [Exponentially Weighted Functions](#)
    - expanding window mean, [Moving Window Functions](#)
    - exponentially weighted functions, [Exponentially Weighted Functions](#)
    - rolling operator, [Moving Window Functions](#), [Moving Window Functions](#)
    - span, [Exponentially Weighted Functions](#)

- user-defined, [User-Defined Moving Window Functions](#)
- NumPy arrays, [Mathematical and Statistical Methods](#)
- pandas objects
  - correlation and covariance, [Correlation and Covariance-Correlation and Covariance](#)
  - summary statistics, [Summarizing and Computing Descriptive Statistics-Summarizing and Computing Descriptive Statistics](#)
  - summary statistics by level, [Summary Statistics by Level](#)
- permutation of data, [Permutation and Random Sampling](#)
  - example, [Example: Random Sampling and Permutation](#)
  - itertools function, [itertools module](#)
  - NumPy random generator method, [Pseudorandom Number Generation](#)
- random sampling, [Permutation and Random Sampling, Example: Random Sampling and Permutation](#)
- statsmodels, [Introduction to statsmodels-Estimating Time Series Processes](#)
  - about, [statsmodels, Introduction to statsmodels](#)
  - about Patsy, [Creating Model Descriptions with Patsy](#)
  - conda install statsmodels, [Example: Group-Wise Linear Regression, Introduction to statsmodels](#)
    - Patsy installed, [Creating Model Descriptions with Patsy](#)
  - email list, [Community and Conferences](#)
  - import statsmodels.api as sm, [Import Conventions, Example: Group-Wise Linear Regression, Estimating Linear Models](#)
  - import statsmodels.formula.api as smf, [Estimating Linear Models](#)
  - linear regression models, [Estimating Linear Models-Estimating Linear Models](#)
  - missing data not allowed, [Introduction to scikit-learn](#)
  - time series analysis, [Estimating Time Series Processes](#)
- stdout from shell command, [Interacting with the Operating System](#)
- str (strings) scalar type, [Strings-Strings](#)
  - about, [Scalar Types](#)
    - immutable, [Strings](#)
    - NumPy string\_type, [Data Types for ndarrays](#)
    - sequences, [Strings](#)
  - backslash () to escape, [Strings](#)

- built-in string object methods, [Python Built-In String Object Methods](#)
- converting objects to, [Strings](#)
- data preparation, [String Functions in pandas-String Functions in pandas](#)
- datetime to and from, [Dates and times, Converting Between String and Datetime-Converting Between String and Datetime](#)
- decoding UTF-8 to, [Bytes and Unicode with Files](#)
- formatting, [Strings](#)
  - datetime as string, [Dates and times, Converting Between String and Datetime-Converting Between String and Datetime](#)
  - documentation online, [Strings](#)
  - f-strings, [Strings](#)
- get\_dummies(), [Computing Indicator/Dummy Variables](#)
- missing data, [String Functions in pandas-String Functions in pandas](#)
- multiline strings, [Strings](#)
- regular expressions, [Regular Expressions-Regular Expressions](#)
- string methods, [String Functions in pandas](#)
- substring methods, [Python Built-In String Object Methods](#)
  - element retrieval, [String Functions in pandas](#)
- type casting, [Type casting](#)
  - NumPy ndarray type casting, [Data Types for ndarrays](#)
- str(), [Strings](#)
  - datetime objects as strings, [Converting Between String and Datetime](#)
  - type casting, [Type casting](#)
- strftime(), [Dates and times, Converting Between String and Datetime](#)
- striding information of ndarrays, [ndarray Object Internals](#)
- strip() to trim whitespace, [Python Built-In String Object Methods](#)
- strptime(), [Dates and times, Converting Between String and Datetime](#)
- structured data, [What Kinds of Data?](#)
- structured ndarrays
  - about, [Structured and Record Arrays](#)
  - memory maps working with, [Memory-Mapped Files](#)
  - nested data types, [Nested Data Types and Multidimensional Fields](#)
  - why use, [Why Use Structured Arrays?](#)

- subplots() (matplotlib), [Figures and Subplots](#)
- subplots\_adjust() (matplotlib), [Adjusting the spacing around subplots](#)
- substring methods, [Python Built-In String Object Methods](#)
- subtraction (see minus (-) operator)
- summary statistics with pandas objects, [Summarizing and Computing Descriptive Statistics-Summarizing and Computing Descriptive Statistics](#)
  - (see also pivot tables)
- swapaxes() (NumPy), [Transposing Arrays and Swapping Axes](#)
- swaplevel(), [Reordering and Sorting Levels](#)
- symmetric\_difference() for sets, [Set](#)
- symmetric\_difference\_update() for sets, [Set](#)
- sys module for getdefaultencoding(), [Files and the Operating System](#)

## T

- tab completion in IPython, [Tab Completion](#)
  - object attributes and methods, [Attributes and methods](#)
- take() (NumPy), [Fancy Indexing Equivalents: take and put](#)
- Taylor, Jonathan, [statsmodels](#)
- tell() position in a file, [Files and the Operating System](#)
- templating strings, [Strings](#)
  - documentation online, [Strings](#)
  - f-strings, [Strings](#)
- text data read from a file, [Reading and Writing Data in Text Format-Reading and Writing Data in Text Format](#)
  - CSV files, [Reading and Writing Data in Text Format-Reading and Writing Data in Text Format](#)
    - defining format and delimiter, [Working with Other Delimited Formats](#)
    - delimiters separating fields, [Reading and Writing Data in Text Format](#)
  - JSON, [JSON Data](#)
  - missing data, [DataFrame](#), [Reading and Writing Data in Text Format](#)
  - other delimited formats, [Working with Other Delimited Formats](#)
  - parsing, [Data Loading, Storage, and File Formats](#)
  - reading in pieces, [Reading Text Files in Pieces](#)

- type inference, [Reading and Writing Data in Text Format](#)
- XML and HTML, [XML and HTML: Web Scraping](#)
- text data written to a file
  - CSV files, [Writing Data to Text Format](#)
  - JSON data, [JSON Data](#)
  - missing data, [Writing Data to Text Format](#)
  - other delimited format, [Working with Other Delimited Formats](#)
  - subset of columns, [Writing Data to Text Format](#)
- text editors, [Integrated Development Environments and Text Editors](#)
- text mode default file behavior, [Bytes and Unicode with Files](#)
- text() (matplotlib), [Annotations and Drawing on a Subplot](#)
- TextFileReader object from read\_csv(), [Reading and Writing Data in Text Format](#), [Reading Text Files in Pieces](#)
- tilde (~) as NumPy negation operator, [Boolean Indexing](#)
- tile() (NumPy), [Repeating Elements: tile and repeat](#)
- %time(), [Timing Code: %time and %timeit](#)-[Timing Code: %time and %timeit](#)
- time series
  - about, [Time Series](#), [Date Ranges](#), [Frequencies](#), and [Shifting](#)
  - about frequencies, [Frequencies and Date Offsets](#)
  - about pandas, [pandas](#)
  - aggregation and zeroing time fields, [Dates and times](#)
  - basics, [Time Series Basics](#)-[Time Series with Duplicate Indices](#)
    - duplicate indices, [Time Series with Duplicate Indices](#)
    - indexing, selecting, subsetting, [Indexing](#), [Selection](#), [Subsetting](#)
  - data types, [Date and Time Data Types and Tools](#)
    - converting between, [Converting Between String and Datetime](#)-[Converting Between String and Datetime](#)
    - locale-specific formatting, [Converting Between String and Datetime](#)
  - date ranges, frequencies, shifting
    - about, [Date Ranges](#), [Frequencies](#), and [Shifting](#)
    - frequencies and date offsets, [Frequencies and Date Offsets](#)
    - frequencies chart, [Generating Date Ranges](#)
    - generating date ranges, [Generating Date Ranges](#)-[Generating Date Ranges](#)

- shifting, [Shifting \(Leading and Lagging\) Data](#)-[Shifting dates with offsets](#)
- shifting dates with offsets, [Shifting dates with offsets](#)
- week of month dates, [Week of month dates](#)
- fixed frequency, [Time Series](#)
- interpolation when reindexing, [Reindexing](#)
- long or stacked format, [Pivoting “Long” to “Wide” Format](#)-[Pivoting “Long” to “Wide” Format](#)
- moving window functions, [Moving Window Functions](#)-[User-Defined Moving Window Functions](#)
  - binary, [Binary Moving Window Functions](#)
  - decay factor, [Exponentially Weighted Functions](#)
  - expanding window mean, [Moving Window Functions](#)
  - exponentially weighted functions, [Exponentially Weighted Functions](#)
- rolling operator, [Moving Window Functions](#), [Moving Window Functions](#)
- span, [Exponentially Weighted Functions](#)
- user-defined, [User-Defined Moving Window Functions](#)
- periods
  - about, [Periods and Period Arithmetic](#)
  - converting timestamps to and from, [Converting Timestamps to Periods \(and Back\)](#)
  - PeriodIndex from arrays, [Creating a PeriodIndex from Arrays](#)
  - quarterly period frequencies, [Quarterly Period Frequencies](#)
- resampling and frequency conversion, [Resampling and Frequency Conversion](#)
  - downsampling, [Downsampling](#)-[Open-high-low-close \(OHLC\) resampling](#)
  - grouped time resampling, [Grouped Time Resampling](#)
  - open-high-low-close resampling, [Open-high-low-close \(OHLC\) resampling](#)
  - upsampling, [Upsampling and Interpolation](#)
- statsmodels for estimating, [Estimating Time Series Processes](#)
- stock price percent change, [Correlation and Covariance](#)
- time zones (see time zones)

- time type, [Dates and times-Dates and times](#), [Date and Time Data Types and Tools](#)
- time zones, [Time Zone Handling-Operations Between Different Time Zones](#)
  - about, [Time Zone Handling](#)
  - between different time zones, [Operations Between Different Time Zones](#)
  - Bitly links dataset
    - counting time zones in pandas, [Counting Time Zones with pandas-Counting Time Zones with pandas](#)
    - counting time zones in Python, [Counting Time Zones in Pure Python](#)
  - DST, [Time Zone Handling](#), [Operations with Time Zone-Aware Timestamp Objects](#)
  - localization and conversion, [Time Zone Localization and Conversion-Time Zone Localization and Conversion](#)
  - pytz library, [Time Zone Handling](#)
  - time zone-aware objects, [Operations with Time Zone-Aware Timestamp Objects](#)
  - UTC, [Time Zone Handling](#), [Operations with Time Zone-Aware Timestamp Objects](#)
- timedelta type, [Dates and times](#), [Date and Time Data Types and Tools](#)
- timedelta() (datetime), [Date and Time Data Types and Tools](#)
- %timeit(), [Timing Code: %time and %timeit-Timing Code: %time and %timeit](#)
- Timestamp (pandas)
  - formatting, [Converting Between String and Datetime-Converting Between String and Datetime](#)
  - shifting dates with offsets, [Shifting dates with offsets](#)
  - time series basics, [Time Series Basics](#)
  - time zone-aware, [Operations with Time Zone-Aware Timestamp Objects](#)
- timestamps, [Time Series](#), [Time Series Basics](#)
  - normalized to midnight, [Generating Date Ranges](#)
- timezone() (pytz), [Time Zone Handling](#)
- Titanic passenger survival dataset, [Introduction to scikit-learn](#)
- to\_csv(), [Writing Data to Text Format](#)

- `to_datetime()`, [Converting Between String and Datetime](#)
- `to_excel()`, [Reading Microsoft Excel Files](#)
- `to_json()`, [JSON Data](#)
- `to_numpy()`, [Interfacing Between pandas and Model Code](#)
  - convert back to DataFrame, [Interfacing Between pandas and Model Code](#)
- `to_period()`, [Converting Timestamps to Periods \(and Back\)](#)
- `to_pickle()`, [Binary Data Formats](#)
  - caution about long-term storage, [Binary Data Formats](#)
- `to_timestamp()`, [Pivoting “Long” to “Wide” Format](#)
- trace function for debugger, [Other ways to use the debugger](#)
- `transform()`, [Group Transforms and “Unwrapped” GroupBys](#)
- `transpose()` with T attribute, [Transposing Arrays and Swapping Axes](#)
- True, [Booleans](#)
- try/except blocks, [Errors and Exception Handling](#)
- `tuple()`, [Tuple](#)
- tuples, [Tuple-Tuple methods](#)
  - exception types, [Errors and Exception Handling](#)
  - methods, [Tuple methods](#)
  - mutable and immutable, [Tuple](#)
  - rest elements, [Unpacking tuples](#)
  - set list-like elements to, [Set](#)
  - SQL query results, [Interacting with Databases](#)
  - string slicing, [Strings](#)
  - unpacking, [Unpacking tuples](#)
- type (Windows) to print file to screen, [Reading and Writing Data in Text Format](#)
- type casting, [Type casting](#)
  - NumPy ndarrays, [Data Types for ndarrays](#)
    - ValueError, [Data Types for ndarrays](#)
- type inference in reading text data, [Reading and Writing Data in Text Format](#)
- tzinfo type, [Date and Time Data Types and Tools](#)
- `tz_convert()`, [Time Zone Localization and Conversion](#)
- `tz_localize()`, [Time Zone Localization and Conversion](#), [Operations with Time Zone-Aware Timestamp Objects](#)

# U

- ufuncs (universal functions) for ndarrays, [Universal Functions: Fast Element-Wise Array Functions](#)
  - methods, [Universal Functions: Fast Element-Wise Array Functions](#), [ufunc Instance Methods-ufunc Instance Methods](#)
  - pandas objects and, [Function Application and Mapping](#)
  - writing new in Python, [Writing New ufuncs in Python](#)
    - custom compiled via Numba, [Creating Custom numpy.ufunc Objects with Numba](#)
    - faster with Numba, [Writing Fast NumPy Functions with Numba](#)
- UInt16Dtype extension data type, [Extension Data Types](#)
- UInt32Dtype extension data type, [Extension Data Types](#)
- UInt64Dtype extension data type, [Extension Data Types](#)
- UInt8Dtype extension data type, [Extension Data Types](#)
- unary ufuncs, [Universal Functions: Fast Element-Wise Array Functions](#), [Universal Functions: Fast Element-Wise Array Functions](#)
- underscore (`_`)
  - data types with trailing underscores, [NumPy Data Type Hierarchy](#)
  - tab completion and, [Tab Completion](#)
  - unwanted variables, [Unpacking tuples](#)
- Unicode characters
  - backslash (\) to escape in strings, [Strings](#)
  - bytes objects and, [Bytes and Unicode](#)
  - strings as sequences of, [Strings](#)
  - text mode default file behavior, [Bytes and Unicode with Files](#)
- union(), [Set](#)
  - DataFrame method, [Index Objects](#)
- unique and other set logic
  - `is_unique()` property of indexes, [Axis Indexes with Duplicate Labels](#)
  - ndarrays, [Unique and Other Set Logic](#)
  - pandas, [Unique Values, Value Counts, and Membership-Unique Values, Value Counts, and Membership](#)
  - repeated instances, [Background and Motivation](#)
- US baby names dataset, [US Baby Names 1880–2010-Boy names that became girl names \(and vice versa\)](#)

- naming trends analysis, [Analyzing Naming Trends-Boy names that became girl names \(and vice versa\)](#)
  - gender and naming, [Boy names that became girl names \(and vice versa\)](#)
  - increase in diversity, [Measuring the increase in naming diversity](#)
  - last letter revolution, [The “last letter” revolution](#)
- USDA food database, [USDA Food Database-USDA Food Database](#)
- universal functions (see ufuncs)
- Unix
  - cat to print file to screen, [Reading and Writing Data in Text Format](#), [Writing Data to Text Format](#)
  - time zone-aware Timestamps, [Operations with Time Zone-Aware Timestamp Objects](#)
- unstack(), [Hierarchical Indexing](#), [Reshaping with Hierarchical Indexing](#)-[Reshaping with Hierarchical Indexing](#)
- update() for sets, [Set](#)
- upsampling, [Resampling and Frequency Conversion](#), [Upsampling and Interpolation](#)
  - target period as superperiod, [Resampling with Periods](#)
- UTC (coordinated universal time), [Time Zone Handling](#), [Operations with Time Zone-Aware Timestamp Objects](#)
- UTF-8 encoding
  - bytes encoding, [Bytes and Unicode](#)
  - open(), [Files and the Operating System](#)

## V

- value\_counts(), [Unique Values, Value Counts, and Membership](#), [Background and Motivation](#)
  - bar plot tip, [Bar Plots](#)
  - categoricals
    - new categories, [Categorical Methods](#)
    - performance of, [Better performance with categoricals](#)
- VanderPlas, Jake, [Conclusion](#)
- variables, [Variables and argument passing](#)
  - binary operators, [Binary operators and comparisons](#)

- command history input and output variables, [Input and Output Variables](#)
- dynamic references, strong types, [Dynamic references, strong types](#)
  - duck typing, [Duck typing](#)
- module imports, [Imports](#)
- namespace, [Namespaces, Scope, and Local Functions](#)
- None tested for via is operator, [Binary operators and comparisons](#)
- output of shell command, [Shell Commands and Aliases](#)
- str immutable, [Strings](#)
- underscore (\_) for unwanted, [Unpacking tuples](#)
- vectorization with NumPy arrays, [Arithmetic with NumPy Arrays](#), [Array-Oriented Programming with Arrays](#), [Array-Oriented Programming with Arrays](#)
- vectorize() (NumPy), [Writing New ufuncs in Python](#)
- version of Python used by book, [Installation and Setup](#)
- vertical bar (|)
  - OR, [Binary operators and comparisons](#)
    - NumPy ndarrays, [Boolean Indexing](#)
    - union of two sets, [Set](#)
- visualization
  - about, [Plotting and Visualization](#)
  - book on data visualization, [Other Python Visualization Tools](#)
  - matplotlib
    - about, [matplotlib](#), [Plotting and Visualization](#), [Other Python Visualization Tools](#)
    - API primer, [A Brief matplotlib API Primer](#)
    - configuration, [matplotlib Configuration](#)
    - documentation online, [Figures and Subplots](#)
    - invoking, [A Brief matplotlib API Primer](#)
    - patch objects, [Annotations and Drawing on a Subplot](#)
    - plots saved to files, [Saving Plots to File](#)
    - two-dimensional NumPy array, [Array-Oriented Programming with Arrays](#)
  - other Python tools, [Other Python Visualization Tools](#)
  - seaborn and pandas, [Plotting with pandas and seaborn-Facet Grids and Categorical Data](#)
    - about seaborn, [Plotting with pandas and seaborn](#), [Bar Plots](#)

- bar plots, [Bar Plots-Bar Plots](#)
- box plots, [Facet Grids and Categorical Data](#)
- density plots, [Histograms and Density Plots-Histograms and Density Plots](#)
- documentation online, [Facet Grids and Categorical Data](#)
- facet grids, [Facet Grids and Categorical Data-Facet Grids and Categorical Data](#)
- histograms, [Histograms and Density Plots-Histograms and Density Plots](#)
- line plots, [Line Plots-Line Plots](#)
- scatter or point plots, [Scatter or Point Plots-Scatter or Point Plots](#)
- vstack() (NumPy), [Concatenating and Splitting Arrays](#)

## W

- web API interactions, [Interacting with Web APIs-Interacting with Web APIs](#)
- website for book
  - book materials, [Data for Examples](#)
  - installation instructions, [Installation and Setup](#)
- week of month (WOM) dates, [Week of month dates](#)
- where() (NumPy), [Combining Data with Overlap](#)
- while loops, [while loops](#)
  - NumPy array vectorization instead, [NumPy Basics: Arrays and Vectorized Computation](#), [Arithmetic with NumPy Arrays](#), [Array-Oriented Programming with Arrays](#), [Array-Oriented Programming with Arrays](#)
  - performance tip, [Performance Tips](#)
- whitespace
  - Python indentation, [Indentation, not braces](#)
  - strip() to trim, [Python Built-In String Object Methods](#)
  - text file delimiter, [Reading and Writing Data in Text Format](#)
- Wickham, Hadley, [How to Think About Group Operations, US Baby Names 1880–2010](#)
- Wilke, Claus O., [Other Python Visualization Tools](#)
- Williams, Ashley, [USDA Food Database-USDA Food Database](#)
- Windows

- exit() to exit Python shell, [Miniconda on Windows](#), [The Python Interpreter](#)
- Miniconda installation, [Miniconda on Windows](#)
- Python Tools for Visual Studio, [Integrated Development Environments and Text Editors](#)
- type to print file to screen, [Reading and Writing Data in Text Format](#)
- writable() file, [Files and the Operating System](#)
- write() to a file, [Files and the Operating System](#)
- writelines() to a file, [Files and the Operating System](#)
- writing CSV files, [Writing Data to Text Format](#)
- writing data to a file
  - binary data
    - Excel format, [Reading Microsoft Excel Files](#)
    - HDF5 format, [HDF5 and Other Array Storage Options](#)
    - memory-mapped files, [Memory-Mapped Files](#)
    - ndarrays saved, [File Input and Output with Arrays](#)
    - pickle format, [Binary Data Formats](#)
    - pickle format caution, [Binary Data Formats](#)
    - plots saved to files, [Saving Plots to File](#)
  - text data
    - CSV files, [Writing Data to Text Format](#)
    - JSON data, [JSON Data](#)
    - missing data, [Writing Data to Text Format](#)
    - other delimited format, [Working with Other Delimited Formats](#)
    - subset of columns, [Writing Data to Text Format](#)

## X

- xlim() (matplotlib), [Ticks, Labels, and Legends](#)
- XML file format, [XML and HTML: Web Scraping](#)
  - reading, [Parsing XML with lxml.objectify](#)

## Y

- yield in a function, [Generators](#)

## Z

- zip files of datasets on GitHub, [Data for Examples](#)
- zip() for list of tuples, [zip](#)

[Support](#) | [Sign Out](#)

©2022 O'REILLY MEDIA, INC. [TERMS OF SERVICE](#) | [PRIVACY POLICY](#)