



Head First Python, 2nd Edition



PREV

[E. Getting Involved: The Python Community](#)

NEXT

[About the Author](#)

Index

A NOTE ON THE DIGITAL INDEX

A link in an index entry is displayed as the section title in which that entry appears. Because some sections have multiple index markers, it is not unusual for an entry to have several links to the same section. Clicking on any link will take you directly to the place in the text in which the marker appears.

SYMBOLS

>>> (see Python Shell)

<> (angle brackets), [Examine the Raw Data with View Source](#)

= (assignment operator), [Data Structures Come Built-in](#), [Creating Lists Literally](#), [What Looks Like a Copy, But Isn't](#)

\ (backslash), [List Slices in Action](#)

^ (caret), [Understanding the Failure Messages](#)

: (colon) (see colon (:))

, (comma), [A List Is an Ordered Collection of Objects](#), [Sets Don't Allow Duplicates](#), [Watch Out for Single-Object Tuples](#)

+ (concatenation operator), [4. Formats for Strings and the Like](#)

{ } (curly braces) (see curly braces {})

-- (decrement operator), [Updating a Frequency Counter, v2.0](#)

/ (forward slash), [Decorating a Function with a URL](#)

+= (increment operator), [Updating a Frequency Counter, v2.0](#), [Adding a Method to a Class](#)

* (multiplication operator), [Python's "for" Loop Understands Slices](#)

* notation, [Accepting a List of Arguments](#)

** notation, [Accepting a Dictionary of Arguments](#)

() (parentheses) (see parentheses ())

[] (square brackets) (see square brackets [])

@ symbol, [Decorating a Function with a URL](#)

symbol, [Introducing Functions](#)

Find answers on the fly, or master something new. Subscribe today. [See pricing options.](#)

| (vertical bar), [Log a Single Line of Delimited Data](#)

A

Alt-P key combination (Linux/Windows), [Asking the Interpreter for Help, Ensuring Initialization Before Use](#)

angle brackets, [Examine the Raw Data with View Source](#)

annotations (function), [Use Annotations to Improve Your Docs](#)

append method, “Growing” a List at Runtime, What Looks Like a Copy, But Isn’t, [Processing Data: What We Already Know](#)

app.run() function, [Decorating a Function with a URL, Exposing Functionality to the Web, Rendering Templates from Flask](#)

apt-get utility, [Install Python 3 on Linux](#)

*args keyword, [Accepting a List of Arguments, The Final Step: Handling Arguments](#)

arguments

about, [Introducing Functions, Functions Can Accept Arguments](#)

adding multiple, [Making a Generically Useful Function](#)

any number and type of, [Accepting Any Number and Type of Function Arguments](#)

by-address argument passing, [Giving your code away \(a.k.a. sharing\), Demonstrating Call-by-Reference Semantics](#)

by-reference argument passing, [Giving your code away \(a.k.a. sharing\), Demonstrating Call-by-Reference Semantics](#)

by-value argument passing, [Giving your code away \(a.k.a. sharing\), Demonstrating Call-by-Reference Semantics](#)

dictionary of, [Accepting a Dictionary of Arguments](#)

function decorators, [Handling Posted Data, Accepting a List of Arguments, The Final Step: Handling Arguments](#)

interpreter processing, [What About Type Information?](#)

list of, [Accepting a List of Arguments](#)

methods and, [Method Invocation: What Actually Happens, Are You Serious About “self”?, Prefix Your Attribute Names with “self”](#)

positional versus keyword assignment, [Positional Versus Keyword Assignment](#)

specifying default values for, [Specifying Default Values for Arguments](#)

arrays (see lists)

arrow symbol, [Use Annotations to Improve Your Docs](#)

assignment operator, [Data Structures Come Built-in, Creating Lists Literally, What Looks Like a Copy, But Isn’t](#)

assignment statements, [Data Structures Come Built-in](#)

associative arrays (see dictionaries)

asterisks, [Accepting a List of Arguments](#)

asyncio module, [7. Running Your Code Concurrently](#)

async keyword, [7. Running Your Code Concurrently](#)

AttributeError exception, [Be Careful When Chaining Method Calls](#)

attributes (state)

about, [“Everything Is an Object”](#)

classes and, [An Object-Oriented Primer, Prefix Your Attribute Names with “self”](#)

dictionary lookup retrieves, [Dictionary Lookup Retrieves State](#)

displaying, [Generating Random Integers with Python](#)

Flask's session technology and, [Flask's Session Technology Adds State](#)

initializing values, [Initialize \(Attribute\) Values Before Use](#)

methods and, [Prefix Your Attribute Names with "self"](#)

objects and, [Objects Share Behavior but Not State, It's Worth Repeating Ourselves: Objects Share Behavior but Not State, Prefix Your Attribute Names with "self"](#)

authentication, [Your Webapp Is Working Well, But...](#)

automated testing, [9. It's Not Over 'Til It's Tested](#)

automatic reloading webapps, [Using Request Data in Your Webapp](#)

await keyword, [7. Running Your Code Concurrently](#)

B

backslash, [List Slices in Action](#)

BDFL (Benevolent Dictator for Life), [BDFL: Benevolent Dictator for Life](#)

Beazley, David, [7. Running Your Code Concurrently, Our Favorite Python Books](#)

behavior (see [methods \(behavior\)](#))

BIF (built-in functions), [Recalling the Built-in Data Structures](#)

binary mode, [Reading Data from an Existing File](#)

blocks of code (see [suites of code](#))

bokeh library, [4. Doing Data Science](#)

bool built-in function, [Functions Return a Result](#)

boolean values, [Avoiding KeyErrors at Runtime](#)

by-address argument passing, [Giving your code away \(a.k.a. sharing\), Demonstrating Call-by-Reference Semantics](#)

by-reference argument passing, [Giving your code away \(a.k.a. sharing\), Demonstrating Call-by-Reference Semantics](#)

by-value argument passing, [Giving your code away \(a.k.a. sharing\)](#)

C

call-by-reference semantics, [Giving your code away \(a.k.a. sharing\), Demonstrating Call-by-Reference Semantics](#)

call-by-value semantics, [Giving your code away \(a.k.a. sharing\)](#)

CamelCase, [Creating Objects from Classes](#)

caret, [Understanding the Failure Messages](#)

case sensitivity and conventions, [Avoiding KeyErrors at Runtime, Creating Objects from Classes](#)

cd command, [Running Python from the Command Line](#)

Ceder, Naomi, [A Tolerant Community: Respect for Diversit](#)

ChainMap class, [6. More from the Standard Library](#)

classes

about, [An Object-Oriented Primer](#)

attributes and, [An Object-Oriented Primer, Prefix Your Attribute Names with "self"](#)

[creating, Hooking into the “with” Statement](#)
[defining functionality of, Objects Share Behavior but Not State](#)
[empty, Creating Objects from Classes, Creating Custom Exceptions](#)
[methods and, An Object-Oriented Primer, Adding a Method to a Class](#)
[naming, Creating Objects from Classes](#)
[objects and, Creating Objects from Classes](#)
[with statement and, You’ve Seen This Pattern Before, Hooking into the “with” Statement, Consider What You’re Trying to Do, Revisited](#)
[class keyword, Creating Objects from Classes](#)
[@classmethod decorator, 3. More on Object Orientation](#)
[client error messages, Understanding HTTP Status Codes](#)
[close method, Python Supports Open, Process, Close](#)
[Code of Conduct, A Tolerant Community: Respect for Diversit](#)
[collections module, 6. More from the Standard Library](#)
[colon \(:\)](#)
[blocks of code and, What Happened to My Curly Braces?](#)
[comprehensions and, How to Spot a Comprehension](#)
[dictionaries and, How to Spot a Dictionary in Code, Sets Don’t Allow Duplicates, How to Spot a Comprehension](#)
[functions and, Naming a Chunk of Code with “def”, Use Annotations to Improve Your Docs](#)
[lists and, Lists Understand Start, Stop, and Step](#)
[combinations function \(itertools module\), 6. More from the Standard Library](#)
[comma, A List Is an Ordered Collection of Objects, Sets Don’t Allow Duplicates, Watch Out for Single-Object Tuples](#)
[command-line, running Python from, Running Python from the Command Line, Install the Testing Developer Tools](#)
[comments, Introducing Functions](#)
[comparison operators, Data Structures Come Built-in, Deciding When to Run Blocks of Code](#)
[compilation, Code Runs Immediately](#)
[comprehensions, What’s the Big Deal?](#)
[\(see also specific types of loops\)](#)
[about, What’s the Big Deal?, That Feeling You Get..., Concluding Remarks](#)
[Bahama Buzzers example, Bahamas Buzzers Have Places to Go](#)
[converting patterns into, Converting Patterns into Comprehensions](#)
[dictionary, What’s the Big Deal?, Deal with Complexity the Python Way, How to Spot a Comprehension](#)
[examining, Take a Closer Look at the Comprehension](#)
[list, What’s the Big Deal?, That Feeling You Get..., How to Spot a Comprehension, Parentheses Around Code == Generator](#)
[reading CSV data as dictionaries, Reading CSV Data As Dictionaries](#)
[reading CSV data as lists, Reading CSV Data As Lists](#)
[set, That Feeling You Get...](#)

spotting patterns, [Did You Spot the Pattern in Your Code?](#)

transforming data, [Transforming Data into the Format You Need](#)

tuples and, [That Feeling You Get... What About “Tuple Comprehensions”?](#)

concatenation operator, [4. Formats for Strings and the Like](#)

concurrency options, [Doing More Than One Thing at Once](#)

concurrent.futures module, [7. Running Your Code Concurrently](#)

connect function, [Confirm Your Table Is Ready for Data](#)

constant lists (tuples), [Ordered Collections Are Mutable/Immutable](#)

constructor methods, [Initialize \(Attribute\) Values Before Use](#)

contextlib module, [Consider What You're Trying to Do, Revisited](#)

context management protocol

about, [You've Seen This Pattern Before, Hooking into the “with” Statement, Managing Context with Methods](#)

creating a context manager, [Consider What You're Trying to Do, Revisited, You've Already Seen a Context Manager in Action](#)

creating a context manager class, [Create a New Context Manager Class](#)

exception handling and, [Handling Other Database Errors](#)

function decorators and, [Creating More Decorators](#)

initializing context manager class, [Managing Context with Methods](#)

performing set-up, [Managing Context with Methods, Perform Setup with Dunder “enter”](#)

performing tear-down, [Managing Context with Methods, Perform Teardown with Dunder “exit”](#)

readdressing webapp code, [Reconsidering Your Webapp Code, 1 of 2](#)

testing context manager, [Perform Teardown with Dunder “exit”](#)

control statements, [What Happened to My Curly Braces?](#)

copy method, [How to Copy a Data Structure](#)

Counter class, [6. More from the Standard Library](#)

CPython, [10. Alternative Implementations](#)

Cross-site Scripting (XSS), [Web Attacks Are a Real Pain, Your Function Calls Can Fail](#)

CSV data

Bahamas Buzzers example, [Bahamas Buzzers Have Places to Go](#)

reading as dictionaries, [Reading CSV Data As Dictionaries](#)

reading as lists, [Reading CSV Data As Lists](#)

csv module, [Reading CSV Data As Lists](#)

Ctrl-C key combination, [Exposing Functionality to the Web, We're Ready for a Test Run](#)

Ctrl-P key combination (Mac), [Asking the Interpreter for Help, Ensuring Initialization Before Use](#)

curly braces { }

blocks of code and, [What Happened to My Curly Braces?](#)

comprehensions and, [How to Spot a Comprehension](#)

dictionaries and, [Selecting a Frequency Count Data Structure, A Dictionary Containing a Dictionary](#)

sets and, [Sets Don't Allow Duplicates](#)

template engines, [Create the HTML, then send it to the browser](#)

current working directory, [Functions + Modules = The Standard Library, How Are Modules Found?](#)

cursor method, [Confirm Your Table Is Ready for Data, How Are You Querying Your Database?](#)

D

database-enabling webapps

creating code to work with database and tables, [Task 4: Create Code to Work with Our Webapp's Database and Tables](#)

creating database and tables, [Task 3: Create Our Webapp's Database and Tables](#)

exception handling and, [Databases Aren't Always Available, Input-Output Is \(Sometimes\) Slow, Your Function Calls Can Fail, Handling Other Database Errors](#)

installing MySQL-Connector/Python, [Install MySQL-Connector/Python](#)

installing MySQL database driver, [Task 2: Install a MySQL Database Driver for Python](#)

installing MySQL server, [Task 1: Install the MySQL Server](#)

introducing Python's DB API, [Introducing Python's DB-API](#)

reusing database code, [How Best to Reuse Your Database Code?](#)

sharing code (see context management protocol)

storing data, [Storing Data Is Only Half the Battle](#)

Data Science, [4. Doing Data Science](#)

data structures

built-in, [Data Structures Come Built-in, Meet the Four Built-in Data Structures, Recalling the Built-in Data Structures](#)

complex, [Combining the Built-in Data Structures, Does This Remind You of Anything?](#)

copying, [How to Copy a Data Structure](#)

dictionaries (see dictionaries)

lists (see lists)

sets (see sets)

tuples (see tuples)

datetime module, [Executing Code, One Statement at a Time, Functions + Modules = The Standard Library, Let's Do the Basic Conversions](#)

day attribute (date.today), [Functions + Modules = The Standard Library](#)

DB-API, [Advance Praise for Head First Python, Second Edition, Using a Database: Putting Python's DB-API to Use, Introducing Python's DB-API](#)

debugging, [Refining the Edit/Stop/Start/Test Cycle, 10. Debug, Debug, Debug](#)

decorators, function (see function decorators)

decrement operator, [Updating a Frequency Counter, v2.0](#)

default values for arguments, [Specifying Default Values for Arguments](#)

def statement

about, [Introducing Functions, Naming a Chunk of Code with "def"](#)

async keyword and, [7. Running Your Code Concurrently](#)

default values for arguments, [Specifying Default Values for Arguments](#)

positional versus keyword assignment, [Positional Versus Keyword Assignment](#)

delimiters, [List Slices in Action](#), [Log a Single Line of Delimited Data](#), [How to Spot a Comprehension](#)

describe log command, [Confirm Your Table Is Ready for Data](#), [Confirm Your Table Is Ready for Data](#)

dictionaries

- about, [An Unordered Data Structure: Dictionary, How Can a Dictionary Help Here?](#)
- of arguments, [Accepting a Dictionary of Arguments](#)
- checking for membership in, [Checking for Membership with "in"](#)
- dictionaries within, [Storing a Table of Data](#)
- dynamic, [Just How Dynamic Are Dictionaries?](#)
- easy to read, [Make Dictionaries Easy to Read](#)
- empty, [Selecting a Frequency Count Data Structure](#), [Storing a Table of Data](#), [Recalling the Built-in Data Structures](#)
- frequency counts in, [Recap: Displaying Found Vowels \(Lists\), Sets: What You Already Know](#)
- growing at run-time, [Working with Dictionaries at Runtime](#)
- iterating over, [Iterating Over a Dictionary](#)
- iterating over keys and values, [Iterating Over Keys and Values](#)
- iterating over rows of data, [Iterating Over a Dictionary with "items"](#)
- key/value pairs and, [An Unordered Data Structure: Dictionary, A Dictionary Stores Key/Value Pairs, Just How Dynamic Are Dictionaries?](#)
- of lists, [Transforming into a Dictionary Of Lists](#)
- reading CSV data as, [Reading CSV Data As Dictionaries](#)
- specifying ordering on output, [Dictionaries: What We Already Know](#)
- spotting in code, [How to Spot a Dictionary in Code](#)
- spotting pattern with, [Did You Spot the Pattern in Your Code?](#)
- square brackets and, [Insertion Order Is NOT Maintained](#)

dictionary comprehensions, [What's the Big Deal?](#), [Deal with Complexity the Python Way](#), [How to Spot a Comprehension](#)

difference method, [Taking Advantage of Set Methods](#), [difference Tells You What's Not Shared](#)

dir built-in function, [Generating Random Integers with Python](#), [Learning More About the Request Object](#), [Dunder "init" Initializes Attributes](#)

distribution packages, [Getting a Module into Site-packages](#)

Django framework, [How Does Flask Work?](#), [5. Web Development Technologies](#)

docstring

- about, [Introducing Functions](#)
- adding, [Use IDLE's Editor to Make Changes](#), [Creating Another Function, 3 of 3](#)
- adding information to, [Use Annotations to Improve Your Docs](#)
- updating, [Making a Generically Useful Function](#)

doctest module, [9. It's Not Over 'Til It's Tested](#)

documenting functions, [Use Annotations to Improve Your Docs](#)

[dot-notation syntax, Importation Confusion, “Growing” a List at Runtime, Invoking a Method: Understand the Details](#)

[dunder name, Creating a Flask Webapp Object, Preparing Your Webapp for the Cloud, Dunder “init” Initializes Attributes, Managing Context with Methods](#)

[duplicate objects, sets and, A Data Structure That Avoids Duplicates: Set, Checking for Membership with “in”](#)

[dynamic assignment of variables, Data Structures Come Built-in, Numbers, Strings...and Objects](#)

[dynamic dictionaries, Just How Dynamic Are Dictionaries?](#)

[dynamic lists, Meet the Four Built-in Data Structures, Removing Objects from a List](#)

E

[Eclipse IDE, 2. Alternatives to IDLE](#)

[edit window, Jump Right In, Use Your Editor When Working on More Than a Few Lines of Code, Invoking Your Function](#)

[elif statement, What “else” Can You Have with “if”?](#)

[else statement, What Happened to My Curly Braces?, Checking for Membership with “in”](#)

[embedded dictionaries, Storing a Table of Data](#)

[embedded suites of code, Suites Can Contain Embedded Suites](#)

[empty classes, Creating Objects from Classes, Creating Custom Exceptions](#)

[empty dictionaries, Selecting a Frequency Count Data Structure, Storing a Table of Data, Recalling the Built-in Data Structures](#)

[empty lists, Creating Lists Literally, “Growing” a List at Runtime, Recalling the Built-in Data Structures](#)

[empty sets, Returning More Than One Value](#)

[empty statements, Creating Objects from Classes](#)

[empty strings, Functions Return a Result](#)

[empty tuples, Recalling the Built-in Data Structures](#)

[Enter key, Returning to the Python Shell](#)

[__enter__ method, Managing Context with Methods, The DBcm Module, Revisited](#)

[environ attribute \(os module\), Functions + Modules = The Standard Library](#)

[environment variables, Functions + Modules = The Standard Library](#)

[escape characters, List Slices in Action, It's Time to Escape \(Your Data\)](#)

[escape function \(flask module\), It's Time to Escape \(Your Data\), Processing Data: What We Already Know](#)

[escape function \(html module\), Functions + Modules = The Standard Library](#)

[Exception class, A Lot of Things Can Go Wrong](#)

[exception handling,](#)

[built-in exceptions, A Lot of Things Can Go Wrong](#)

[catch-all exception handler, The Catch-All Exception Handler, The Catch-All Exception Handler, Revisited](#)

[context manager and, Handling Other Database Errors](#)

[creating custom exceptions, Creating Custom Exceptions](#)

[databases and, Databases Aren't Always Available, Input-Output Is \(Sometimes\) Slow, Your Function Calls Can Fail, Handling Other](#)

[Database Errors, What Else Can Go Wrong with “DBcm”?](#)

[functions and, \[Your Function Calls Can Fail\]\(#\)](#)

[import mechanisms, \[Importation Confusion\]\(#\)](#)

[indentation errors, \[Don't Forget to Try the Beer Song Code\]\(#\)](#)

[misspelled variables, \[Don't Forget to Try the Beer Song Code\]\(#\)](#)

[output display and, \[View the Log Through Your Webapp\]\(#\)](#)

[PEP 8 failure messages, \[How PEP 8-Compliant Is Our Code?\]\(#\)](#)

[run-time and, \[Just How Dynamic Are Dictionaries? Always Try to Execute Error-Prone Code, Is Your Webapp Robust Now?\]\(#\)](#)

[syntax errors, \[What Happens Next..., Use Your Editor When Working on More Than a Few Lines of Code\]\(#\)](#)

[webapps and, \[View the Log Through Your Webapp, Databases Aren't Always Available, Your Function Calls Can Fail, Getting Back to Our Webapp Code, Silently Handling Exceptions\]\(#\)](#)

[with statement and, \[The DBcm Module, Revisited, Handling SQL.Error Is Different\]\(#\)](#)

[executing code, \[Just How Dynamic Are Dictionaries?\]\(#\)](#)

[\(see also \[run-time\]\(#\)\)](#)

[Alt-P key combination for, \[Asking the Interpreter for Help, Ensuring Initialization Before Use\]\(#\)](#)

[Ctrl-P key combination for, \[Asking the Interpreter for Help, Ensuring Initialization Before Use\]\(#\)](#)

[F5 key for, \[Understanding IDLE's Windows, Press F5 to Run Your Code, Use IDLE's Editor to Make Changes\]\(#\)](#)

[interpreter processing in, \[Executing Code, One Statement at a Time\]\(#\)](#)

[invoking functions, \[Invoking Your Function\]\(#\)](#)

[pausing execution, \[Extending Our Program to Do More, Arranging to Pause Execution\]\(#\)](#)

[running concurrently, \[7. Running Your Code Concurrently\]\(#\)](#)

[running immediately, \[Code Runs Immediately, Returning to the Python Shell\]\(#\)](#)

[running multiple times, \[Extending Our Program to Do More\]\(#\)](#)

[__exit__ method, \[Managing Context with Methods, The DBcm Module, Revisited, Be Careful with Code Positioning\]\(#\)](#)

[extend method, \[Extending a List with Objects\]\(#\)](#)

[extends directive \(Jinja2\), \[Create the HTML, then send it to the browser\]\(#\)](#)

F

[F5 key, \[Understanding IDLE's Windows, Press F5 to Run Your Code, Use IDLE's Editor to Make Changes\]\(#\)](#)

[False value, \[Functions Return a Result\]\(#\)](#)

[FileNotFoundError exception, \[Always Try to Execute Error-Prone Code, The Catch-All Exception Handler, Revisited\]\(#\)](#)

[Flask class, \[Here's What Happened \\(Line by Line\\), Rendering Templates from Flask\]\(#\)](#)

[Flask framework](#)

[about, \[How Does Flask Work?, 5. Web Development Technologies\]\(#\)](#)

[accessing HTML form data, \[Accessing HTML Form Data with Flask\]\(#\)](#)

[associating function with multiple URLs, \[Functions Can Have Multiple URLs\]\(#\)](#)

creating webapp objects, [Creating a Flask Webapp Object](#)

debugging mode, [Refining the Edit/Stop/Start/Test Cycle](#)

installing, [Let's Install Flask](#)

Jinja2 template engine, [Create the HTML, then send it to the browser, Producing the Results As HTML, Generate Readable Output With HTML, Producing Readable Output with Jinja2](#)

Markup object, [It's Time to Escape \(Your Data\)](#)

rendering templates from, [Rendering Templates from Flask](#)

request object, [Accessing HTML Form Data with Flask, Learning More About the Request Object](#)

running webapps, [Running Your Flask Webapp for the First Time](#)

session mechanism, [It's Time for a Bit of a Session](#)

testing webapps, [We're Ready for a Test Run](#)

threading and, [Don't Get Bummed Out: Flask Can Help](#)

flask module

escape function, [It's Time to Escape \(Your Data\), Processing Data: What We Already Know](#)

Flask class, [Here's What Happened \(Line by Line\), Rendering Templates from Flask](#)

session dictionary, [Flask's Session Technology Adds State](#)

for loop, [What's the Big Deal?](#)

(see also [comprehensions](#))

about, [Extending Our Program to Do More, Iterating Over a Sequence of Objects, That Feeling You Get...](#)

lists and, [Python's "for" Loop Understands Lists, Reading CSV Data As Lists](#)

slices and, [Python's "for" Loop Understands Slices](#)

spotting patterns in, [Did You Spot the Pattern in Your Code?](#)

format method, [4. Formats for Strings and the Like](#)

formatting

data, [Transforming Data into the Format You Need](#)

strings, [4. Formats for Strings and the Like](#)

form dictionary (Flask), [Accessing HTML Form Data with Flask](#)

<form> tag, [Understanding HTTP Status Codes](#)

forward slash, [Decorating a Function with a URL](#)

frequency counts

about, [Recap: Displaying Found Vowels \(Lists\), Sets: What You Already Know](#)

incrementing, [Updating a Frequency Counter](#)

initializing, [Updating a Frequency Counter](#)

selecting data structure, [Selecting a Frequency Count Data Structure](#)

updating, [Updating a Frequency Counter](#)

function decorators

about, [Exposing Functionality to the Web, You've Been Using Decorators All Along](#)

adding, [Rendering Templates from Flask](#)

adjusting behaviors, [Decorating a Function with a URL](#)

arguments and, [Handling Posted Data, Accepting a List of Arguments, The Final Step: Handling Arguments](#)

components in writing, [You've Been Using Decorators All Along](#)

context managers and, [Creating More Decorators](#)

creating, [A Recipe for Creating a Function Decorator](#)

URLs and, [Decorating a Function with a URL, Exposing Functionality to the Web, Exposing Functionality to the Web, Displaying the Webapp's HTML Form, Handling Posted Data, Recap: We Need to Restrict Access to Certain URLs, Back to Restricting Access to /viewlog](#)

function objects, [Pass a Function to a Function, A Recipe for Creating a Function Decorator, Creating a Function Decorator](#)

functions (see also arguments (functions); specific functions)

about, [Functions + Modules = The Standard Library, Introducing Functions](#)

best practice for, [Follow Best Practice As Per the PEPs](#)

built-in, [Recalling the Built-in Data Structures](#)

creating, [Naming a Chunk of Code with "def", Creating Another Function, 1 of 3](#)

documenting, [Use Annotations to Improve Your Docs](#)

editing, [Invoking Your Function](#)

embedding generators within, [Using a Generator: What Just Happened?](#)

exception handling and, [Your Function Calls Can Fail](#)

importing, [Functions + Modules = The Standard Library, Arranging to Pause Execution](#)

invoking, [Invoking Your Function](#)

invoking passed functions, [Invoking a Passed Function](#)

methods and, [Invoking a Method: Understand the Details, Prefix Your Attribute Names with "self"](#)

modules and, [Functions + Modules = The Standard Library, Functions Beget Modules](#)

multiple URLs, [Functions Can Have Multiple URLs](#)

naming, [Naming a Chunk of Code with "def", Making a Generically Useful Function, Creating Objects from Classes](#)

nested, [Functions Can Be Nested Inside Functions, That's almost too easy, isn't it?](#)

passing to functions, [Pass a Function to a Function](#)

returning from functions, [Return a Function from a Function](#)

returning results, [Functions Return a Result](#)

reusing code with, [Reusing Code with Functions, Functions Beget Modules](#)

sharing, [Functions Beget Modules](#)

string quote characters, [What's the Deal with All Those Strings?](#)

troubleshooting, [Giving your code away \(a.k.a. sharing\), Demonstrating Call-by-Reference Semantics](#)

variables and, [Coping with Scoping](#)

functools module, [The Final Step: Handling Arguments, 6. More from the Standard Library](#)

G

generators, [Parentheses Around Code == Generator, Using a Generator to Process URLs](#)

getcwd function (os module), [Functions + Modules = The Standard Library](#)

GET method (HTTP), [Understanding HTTP Status Codes](#)

global variables, [Your Web Server \(Not Your Computer\) Runs Your Code](#)

H

hashes (see dictionaries)

Hellman, Doug, [6. More from the Standard Library](#)

help command, [Asking the Interpreter for Help, Asking the Interpreter for Help on a Function, What About Using Square Brackets?](#)

Homebrew package manager, [Task 1: Install the MySQL Server, Install Python 3 on Mac OS X \(macOS\)](#)

HTML forms

access with Flask, [Accessing HTML Form Data with Flask](#)

building, [Building the HTML Form](#)

displaying, [Displaying the Webapp's HTML Form](#)

producing results, [Producing the Results As HTML](#)

redirecting to avoid unwanted errors, [Adding a Finishing Touch](#)

rendering templates from Flask, [Templates Relate to Web Pages](#)

testing template code, [Preparing to Run the Template Code](#)

html module, [Functions + Modules = The Standard Library](#)

HTTP (HyperText Transfer Protocol)

status codes, [Understanding HTTP Status Codes](#)

web servers and, [Your Web Server \(Not Your Computer\) Runs Your Code](#)

I

id built-in function, [Understanding CountFromBy's Representation](#)

IDLE (Python IDE), [Jump Right In, How Does Flask Work?, 2. Alternatives to IDLE](#)

if statement, [What Happened to My Curly Braces?, Checking for Membership with "in"](#)

ImmutableMultiDict dictionary, [Logging Specific Web Request Attributes](#)

ImportError exception, [Not Found Modules Produce ImportError](#)

import statement

about, [Functions + Modules = The Standard Library, Arranging to Pause Execution](#)

Flask framework and, [Here's What Happened \(Line by Line\)](#)

interpreter search considerations, [How Are Modules Found?](#)

positioning, [What About That Import?](#)

sharing modules with, [Functions Beget Modules](#)

threading module and, [Doing More Than One Thing at Once](#)

Zen of Python, [The Zen of Python](#)

increment operator, [Updating a Frequency Counter, v2.0, Adding a Method to a Class](#)

indentation levels for suites, [Suites Can Contain Embedded Suites, Don't Forget to Try the Beer Song Code](#)

indenting suites of code

- about, [Deciding When to Run Blocks of Code, Is Indentation Driving You Crazy?](#)
- for functions, [Introducing Functions](#)
- for loops, [Iterating Over a Sequence of Objects, Indent Suites with Format...Indent Region](#)

index values, lists and, [Popping Objects Off a List, Lists Extend the Square Bracket Notation](#)

informational messages, [Understanding HTTP Status Codes](#)

`__init__` method, [Initialize \(Attribute\) Values Before Use, Providing Sensible Defaults for CountFromBy, Managing Context with Methods, The DBcm Module, Revisited](#)

inner functions, [Functions Can Be Nested Inside Functions, That's almost too easy, isn't it?](#)

in operator

- about, [Deciding When to Run Blocks of Code](#)
- dictionaries and, [Just How Dynamic Are Dictionaries?](#)
- lists and, [Putting Lists to Work, Checking for Membership with "in"](#)
- sets and, [Taking Advantage of Set Methods](#)

input built-in function, [It's Time to Update Our Code](#)

insert method, [Inserting an Object into a List](#)

INSERT statement (SQL), [Confirm Your Table Is Ready for Data, How Are You Querying Your Database?](#)

InterfaceError exception, [Always Try to Execute Error-Prone Code, Does "More Errors" Mean "More excepts"?, The DBcm Module, Revisited](#)

interpreter (Python)

- about, [Code Runs Immediately](#)
- alternative implementations, [10. Alternative Implementations](#)
- asking for help, [Asking the Interpreter for Help, Asking the Interpreter for Help on a Function](#)
- case sensitivity, [Avoiding KeyErrors at Runtime](#)
- dictionary keys and, [Iterating Over Keys and Values](#)
- functions and, [What About Type Information?](#)
- identifying operating system, [Functions + Modules = The Standard Library](#)
- identifying site-package locations, [How Are Modules Found?](#)
- internal ordering used by, [An Unordered Data Structure: Dictionary, Iterating Over Keys and Values](#)
- running from command-line, [Running Python from the Command Line](#)
- syntax errors, [What Happens Next..., Use Your Editor When Working on More Than a Few Lines of Code](#)
- whitespace and, [Is Indentation Driving You Crazy?](#)

intersection method, [Taking Advantage of Set Methods, intersection Reports on Commonality, Returning More Than One Value, Creating Another Function, 2 of 3](#)

ipython shell, [1. Alternatives to >>>](#)

IronPython project, [10. Alternative Implementations](#)

isoformat function (datetime module), [Functions + Modules = The Standard Library](#)

items method, [Iterating Over a Dictionary with "items"](#)

itertools module, [6. More from the Standard Library](#)

J

Java VM, [Code Runs Immediately](#)

Jinja2 template engine

about, [Create the HTML, then send it to the browser, Producing the Results As HTML](#)

calculating data needed, [Calculating the Data We Need](#)

readable output with, [Generate Readable Output With HTML, Producing Readable Output with Jinja2](#)

join method, [What About Using Square Brackets?, Viewing the Entire Log in Your Webapp, What's Joined Together Can Be Split Apart](#)

Jones, Brian K., [Our Favorite Python Books](#)

Jupyter Notebook IDE, [3. Jupyter Notebook: The Web-Based IDE](#)

Jython project, [10. Alternative Implementations](#)

K

keyboard shortcuts, [Indent Suites with Format...Indent Region](#)

KeyError exception, [Just How Dynamic Are Dictionaries?](#)

key/value pairs (dictionaries)

about, [An Unordered Data Structure: Dictionary, A Dictionary Stores Key/Value Pairs](#)

adding, [Working with Dictionaries at Runtime](#)

creating on the fly, [Just How Dynamic Are Dictionaries?](#)

interpreter processing and, [Iterating Over Keys and Values](#)

keyword assignment of arguments, [Positional Versus Keyword Assignment](#)

Kivy library, [9. Kivy: Our Pick for "Coolest Project Ever"](#)

**kwargs keyword, [Accepting a Dictionary of Arguments, The Final Step: Handling Arguments](#)

L

len built-in function, ["Growing" a List at Runtime](#)

level of indentation for suites, [Suites Can Contain Embedded Suites](#)

list built-in function, [Experimenting with Ranges, union Works by Combining Sets](#)

list comprehensions, [What's the Big Deal?, That Feeling You Get..., How to Spot a Comprehension, Parentheses Around Code == Generator](#)

lists

about, [Data Structures Come Built-in, Meet the Four Built-in Data Structures, A List Is an Ordered Collection of Objects, Lists: Updating What We Know](#)

assignment operators, [Data Structures Come Built-in, Creating Lists Literally, What Looks Like a Copy, But Isn't](#)

checking for membership in, [Deciding When to Run Blocks of Code, Putting Lists to Work, Checking for Membership with "in"](#)

copying existing, [What Looks Like a Copy, But Isn't](#)

creating literally, [Creating Lists Literally](#)

dictionaries of, [Transforming into a Dictionary Of Lists](#)

dynamic, [Meet the Four Built-in Data Structures, Removing Objects from a List](#)

empty, [Creating Lists Literally, “Growing” a List at Runtime, Recalling the Built-in Data Structures](#)

extending with objects, [Extending a List with Objects](#)

growing at run-time, [“Growing” a List at Runtime](#)

iterating over a sequence of objects, [Iterating Over a Sequence of Objects](#)

of arguments, [Accepting a List of Arguments](#)

popping objects off, [Popping Objects Off a List](#)

reading CSV data as, [Reading CSV Data As Lists](#)

removing objects from, [Removing Objects from a List](#)

slice notation, [List Slices in Action, Slicing a List Is Nondestructive](#)

sorted, [union Works by Combining Sets](#)

spotting in code, [A List Is an Ordered Collection of Objects](#)

spotting pattern with, [Spotting the Pattern with Lists](#)

square bracket notation, [Data Structures Come Built-in, A List Is an Ordered Collection of Objects, What About Using Square Brackets?, Square Brackets Are Everywhere, Slicing a List Is Nondestructive](#)

starting and stopping with, [Starting and Stopping with Lists](#)

stepping with, [Stepping with Lists](#)

tuples and, [Ordered Collections Are Mutable/Immutable, Making the Case for Tuples](#)

when not to use, [What’s Wrong with Lists?](#)

working with, [Putting Lists to Work, Lists: What We Know](#)

working within edit window, [What Happens Next...](#)

literal lists, [A List Is an Ordered Collection of Objects](#)

localhost, [Exposing Functionality to the Web](#)

logins/logouts, [Managing Logins with Sessions, Creating a Function Helps, But...](#)

logs and logging, [Task 4: Create Code to Work with Our Webapp’s Database and Tables](#)

(see also database-enabling webapps)

determining structure for, [Decide on a Structure for Your Log Data](#)

dir built-in function and, [Learning More About the Request Object](#)

examining raw data, [Examine the Raw Data with View Source](#)

open, process, close technique, [The “with” statement manages context, Data is logged \(behind the scenes\)](#)

single line of delimited data, [Log a Single Line of Delimited Data](#)

updating webapps, [Recalling the “log_request” Function](#)

viewing through webapps, [View the Log Through Your Webapp, Viewing the Entire Log in Your Webapp](#)

loopback address, [Exposing Functionality to the Web](#)

loop comprehensions (see comprehensions)

loops (see specific types of loops)

M

MacPorts package manager, [Install Python 3 on Mac OS X \(macOS\)](#)

maps (see dictionaries)

MariaDB, [Database-Enabling Your Webapp](#)

Markup object (Flask), [It's Time to Escape \(Your Data\)](#)

matplotlib/seaborn modules, [4. Doing Data Science](#)

memory management, [Removing Objects from a List](#)

messages, HTTP status codes, [Understanding HTTP Status Codes, Redirect to Avoid Unwanted Errors](#)

methods (behavior)

- about, ["Everything Is an Object"](#)
- arguments and, [Method Invocation: What Actually Happens, Are You Serious About "self"? Prefix Your Attribute Names with "self"](#)
- attributes and, [Prefix Your Attribute Names with "self"](#)
- chaining method calls, [Be Careful When Chaining Method Calls](#)
- classes and, [An Object-Oriented Primer, Adding a Method to a Class](#)
- decorators adjusting, [Decorating a Function with a URL](#)
- functions and, [Invoking a Method: Understand the Details, Prefix Your Attribute Names with "self"](#)
- invoking, [Invoking a Method: Understand the Details](#)
- objects and, [Objects Share Behavior but Not State, It's Worth Repeating Ourselves: Objects Share Behavior but Not State, Prefix Your Attribute Names with "self"](#)
- running webapp, [Running Your Webapp's Behavior\(s\)](#)

MicroPython project, [10. Alternative Implementations](#)

modules

- about, [Executing Code, One Statement at a Time](#)
- adding to site-packages, [Getting a Module into Site-packages](#)
- creating, [Functions Beget Modules](#)
- functions and, [Functions + Modules = The Standard Library, Functions Beget Modules](#)
- ImportError exception, [Not Found Modules Produce ImportError](#)
- importing, [Importation Confusion, Functions Beget Modules](#)
- sharing code, [Modules: What We Know Already](#)
- third party, [Batteries Included](#)

MongoDB, [7. More Data Sources](#)

month attribute (date.today), [Functions + Modules = The Standard Library](#)

multiplication operator, [Python's "for" Loop Understands Slices](#)

multiprocessing module, [7. Running Your Code Concurrently](#)

MySQL

- benefits of, [All That Remains...](#)
- DB-API and, [Introducing Python's DB-API](#)
- exception handling and, [Databases Aren't Always Available, Input-Output Is \(Sometimes\) Slow, Your Function Calls Can Fail, Handling Other Database Errors, What Else Can Go Wrong with "DBcm"?](#)
- installing MySQL-Connector/Python driver, [Task 2: Install a MySQL Database Driver for Python](#)
- installing MySQL server, [Task 1: Install the MySQL Server](#)

querying considerations, [Waiting: What to Do?](#)

MySQL console, [Task 3: Create Our Webapp's Database and Tables](#)

N

NameError exception, [Coping with Scoping](#)

namespaces, [Importation Confusion](#)

__name__ value, [Creating a Flask Webapp Object](#)

naming conventions, [What Happens Next...](#)

nested functions, [Functions Can Be Nested Inside Functions, That's almost too easy, isn't it?](#)

newsletters (Python), [Python Podcasts](#)

Not Found message, [Running Your Webapp's Behavior\(s\)](#)

not in operator, [Checking for Membership with "in", Ensuring Initialization Before Use](#)

numbers

assigning to variables, [Numbers, Strings...and Objects](#)

generating randomly, [Extending Our Program to Do More, Generating Random Integers with Python](#)

numpy package, [4, Doing Data Science](#)

O

object class, [Dunder "init" Initializes Attributes](#)

object instantiation, [Creating Objects from Classes, Initialize \(Attribute\) Values Before Use](#)

object-oriented programming (OOP), [An Object-Oriented Primer, Dunder "init" Initializes Attributes, 3, More on Object Orientation](#)

objects

about, [Numbers, Strings...and Objects](#)

attributes and, [Objects Share Behavior but Not State, It's Worth Repeating Ourselves: Objects Share Behavior but Not State, Prefix Your Attribute Names with "self"](#)

classes and, [Creating Objects from Classes](#)

creating, [Creating Objects from Classes, Initialize \(Attribute\) Values Before Use](#)

defining representation of, [Understanding CountFromBy's Representation](#)

duplicate, [A Data Structure That Avoids Duplicates: Set, Checking for Membership with "in"](#)

extending lists with, [Extending a List with Objects](#)

function, [Pass a Function to a Function, A Recipe for Creating a Function Decorator, Creating a Function Decorator](#)

key/value pairs and, [A Dictionary Stores Key/Value Pairs](#)

methods and, [Objects Share Behavior but Not State, It's Worth Repeating Ourselves: Objects Share Behavior but Not State, Prefix Your Attribute Names with "self"](#)

popping off lists, [Popping Objects Off a List](#)

removing from lists, [Removing Objects from a List](#)

sequence of, [Iterating Over a Sequence of Objects, Creating Sets Efficiently](#)

webapp, [Creating a Flask Webapp Object](#)

open function, [Python Supports Open, Process, Close](#)

opening editing window, [Jump Right In](#)

open, process, close technique

- about, [Python Supports Open, Process, Close](#)
- invoking logging function, [The “with” statement manages context, Data is logged \(behind the scenes\)](#)
- reading data from existing files, [Reading Data from an Existing File](#)
- with statement and, [Reading Data from an Existing File](#)

operating system, identifying for interpreter, [Functions + Modules = The Standard Library](#)

operators

- assignment, [Data Structures Come Built-in, Creating Lists Literally, What Looks Like a Copy, But Isn't](#)
- checking for membership with, [Deciding When to Run Blocks of Code, Putting Lists to Work, Checking for Membership with “in”, Checking for Membership with “in”](#)
- comparison, [Data Structures Come Built-in, Deciding When to Run Blocks of Code](#)
- concatenation, [4. Formats for Strings and the Like](#)
- decrement, [Updating a Frequency Counter, v2.0](#)
- increment, [Updating a Frequency Counter, v2.0, Adding a Method to a Class](#)
- multiplication, [Python’s “for” Loop Understands Slices](#)
- super, [Deciding When to Run Blocks of Code](#)
- ternary, [Checking for Membership with “in”](#)

ordered data structures, [Meet the Four Built-in Data Structures](#)

OrderedDict dictionary, [6. More from the Standard Library](#)

os module

- about, [Functions + Modules = The Standard Library](#)
- environ attribute, [Functions + Modules = The Standard Library](#)
- getcwd function, [Functions + Modules = The Standard Library](#)
- platform attribute, [Functions + Modules = The Standard Library](#)
- usage example, [Functions + Modules = The Standard Library](#)

output display

- exception handling and, [View the Log Through Your Webapp](#)
- Python Shell and, [Returning to the Python Shell](#)
- raw data to readable, [From Raw Data to Readable Output, Generate Readable Output With HTML](#)
- readable via Jinja2, [Producing Readable Output with Jinja2](#)
- specifying dictionary ordering for, [Dictionaries: What We Already Know](#)

P

pandas tools, [4. Doing Data Science](#)

parentheses ()

- comprehensions and, [How to Spot a Comprehension](#)
- function arguments in, [Naming a Chunk of Code with “def”](#)
- object instantiation and, [Creating Objects from Classes](#)

return statement and, [Returning One Value](#)

tuples in, [Making the Case for Tuples, Watch Out for Single-Object Tuples](#)

partial function, [6. More from the Standard Library](#)

pass keyword, [Creating Objects from Classes](#)

pausing execution, [Extending Our Program to Do More, Arranging to Pause Execution](#)

pdb debugger, [10. Debug, Debug, Debug](#)

pep8 plug-in, [Getting Ready to Check PEP 8 Compliance](#)

PEP (Python Enhancement Protocol)

about, [Follow Best Practice As Per the PEPs](#)

DB-API specification, [Introducing Python's DB-API](#)

line length standard, [One Final Change to Our Logging Code](#)

testing for compliance, [Can I Test for PEP 8 Compliance?. 9. It's Not Over 'Til It's Tested](#)

PermissionError exception, [try Once, but except Many Times, The Catch-All Exception Handler, Revisited](#)

permutations function (itertools module), [6. More from the Standard Library](#)

Peters, Tim, [The Zen of Python](#)

pip (Package Installer for Python)

downloading requests library, [Using a Listcomp to Process URLs](#)

installing Flask, [Let's Install Flask](#)

installing packages with, [Installing Packages with "pip"](#)

installing pep8 plug-in, [Getting Ready to Check PEP 8 Compliance](#)

installing pyreadline module, [Add to Python 3 on Windows](#)

installing pytest testing framework, [Getting Ready to Check PEP 8 Compliance](#)

platform attribute (os module), [Functions + Modules = The Standard Library](#)

podcasts (Python), [Python Podcasts](#)

pop method, [Popping Objects Off a List](#)

positional assignment of arguments, [Positional Versus Keyword Assignment](#)

PostgreSQL, [Database-Enabling Your Webapp](#)

POST method (HTTP), [Understanding HTTP Status Codes](#)

pprint function (pprint module), [Pretty-Printing Complex Data Structures](#)

pprint module, [Pretty-Printing Complex Data Structures](#)

print built-in function

about, [Deciding When to Run Blocks of Code](#)

accessing dictionary data values, [Iterating Over Keys and Values](#)

default behavior, [Reading Data from an Existing File](#)

displaying objects, [Defining CountFromBy's Representation](#)

identifying Python version, [Functions + Modules = The Standard Library](#)

optional arguments, [One Final Change to Our Logging Code](#)

product function (itertools module), [6. More from the Standard Library](#)

programming tools, [8. Programming Tools](#)

prompt (Python Shell) (see Python Shell)

protocol port number, [Running Your Flask Webapp for the First Time, Exposing Functionality to the Web](#)

PSF (Python Software Foundation), [BDFL: Benevolent Dictator for Life](#)

ptpython REPL, [2. Alternatives to IDLE](#)

PyCharm IDE, [2. Alternatives to IDLE](#)

py command, [Running Python from the Command Line, Install the Testing Developer Tools](#)

PyCon (Python Conference), [BDFL: Benevolent Dictator for Life](#)

PyLint tool, [8. Programming Tools](#)

pymongo database driver, [7. More Data Sources](#)

PyPI (Python Package Index), [Modules: What We Know Already, Let's Install Flask, 6. Working with Web Data](#)

PyPy project, [10. Alternative Implementations](#)

pyreadline module, [Add to Python 3 on Windows](#)

pytest testing framework, [Getting Ready to Check PEP 8 Compliance](#)

py.test tool, [9. It's Not Over 'Til It's Tested, 8. Programming Tools](#)

Python, [Breaking with Tradition, 1. What About Python 2?](#)

Python, [Jump Right In](#)

about, [Hooking into the "with" Statement](#)

installing on Linux, [Install Python 3 on Linux](#)

installing on Mac OS X, [Install Python 3 on Mac OS X \(macOS\)](#)

installing on Windows, [Install Python 3 on Windows](#)

usage recommendations, [1. What About Python 2?](#)

PythonAnywhere

about, [Pythonanywhere: Deploying Your Webapp](#)

configuring webapps, [Step 5: Configure Your Webapp](#)

creating starter webapp, [Step 4: Create a Starter Webapp, 1 of 2](#)

extracting and installing code, [Step 3: Extract and Install Your Code](#)

preparing webapps, [Step 0: A Little Prep](#)

signing up for, [Step 1: Sign Up for PythonAnywhere](#)

testing deployed webapp, [Step 6: Take Your Cloud-Based Webapp for a Spin!](#)

uploading files to the cloud, [Preparing Your Webapp for the Cloud, Exploiting Dunder Name Dunder Main, Step 2: Upload Your Files to the Cloud](#)

Python community, [Advance Praise for Head First Python, Second Edition](#)

Python Core Developers, [10. Alternative Implementations](#)

Python Packaging Authority, [Modules: What We Know Already](#)

Python Shell

about, [Understanding IDLE's Windows](#)

accessing prompt within, [Functions + Modules = The Standard Library, Both approaches work with Python](#)

alternatives to, [1. Alternatives to >>>](#)

asking for help, [Asking the Interpreter for Help](#), [Asking the Interpreter for Help on a Function](#)

copying code to editor, [Use Your Editor When Working on More Than a Few Lines of Code](#)

experimenting at, [Both approaches work with Python](#), [Experimenting at the Shell](#)

recalling last commands typed, [Asking the Interpreter for Help](#)

running interpreter from, [Running Python from the Command Line](#)

terminating statements with Enter key, [Iterating Over a Sequence of Objects](#)

Q

quit command, [Running Python from the Command Line](#)

quotation marks

comments and, [Introducing Functions](#)

strings and, [List Slices in Action](#), [What's the Deal with All Those Strings?](#)

R

Ramalho, Luciano, [Our Favorite Python Books](#)

randint function (random module), [Extending Our Program to Do More, Generating Random Integers with Python](#), [How Are Modules Found?](#)

random module, [Extending Our Program to Do More, Generating Random Integers with Python](#), [How Are Modules Found?](#)

random number generation, [Extending Our Program to Do More, Generating Random Integers with Python](#)

range built-in function, [Iterating a Specific Number of Times](#), [Is Indentation Driving You Crazy?](#)

reading

CSV data as dictionaries, [Putting Slices to Work on Lists](#)

CSV data as lists, [Reading CSV Data As Lists](#)

data from existing files, [Reading Data from an Existing File](#)

README.txt file, [Creating the Required Setup Files](#)

redirect function (Flask), [Adding a Finishing Touch](#)

redirection messages, [Understanding HTTP Status Codes](#)

remove method, [Removing Objects from a List](#)

render_template function (Flask), [Rendering Templates from Flask](#), [Adding a Finishing Touch](#)

REPL tool, [Understanding IDLE's Windows](#), [2. Alternatives to IDLE](#)

request object (Flask), [Accessing HTML Form Data with Flask](#), [Learning More About the Request Object](#), [Dunder "init" Initializes Attributes](#)

requests library, [Using a Listcomp to Process URLs](#), [6. Working with Web Data](#)

requests module, [6. Working with Web Data](#)

results, functions returning, [Functions Return a Result](#)

return keyword, [Introducing Functions](#)

return statement

about, [Functions Return a Result](#)

parentheses and, [Returning One Value](#)
 returning multiple values, [Returning More Than One Value](#)
 returning one value, [Returning One Value](#)
 return values (functions)
 about, [Functions Return a Result](#)
 interpreter processing, [What About Type Information?](#)
 variable scope and, [Prefix Your Attribute Names with “self”](#)
 route decorator
 about, [Exposing Functionality to the Web](#)
 adding, [Rendering Templates from Flask](#)
 adjusting behaviors, [Decorating a Function with a URL](#)
 optional arguments, [Handling Posted Data](#)
 run-time
 exception handling, [Just How Dynamic Are Dictionaries?, Always Try to Execute Error-Prone Code, Is Your Webapp Robust Now?](#)
 growing dictionaries at, [Working with Dictionaries at Runtime](#)
 growing lists at, [“Growing” a List at Runtime](#)
 RuntimeError exception, [Always Try to Execute Error-Prone Code](#)

S
 scikit-learn tools, [4. Doing Data Science](#)
 scipy modules, [4. Doing Data Science](#)
 scope of variables, [Coping with Scoping](#)
 SELECT statement (SQL), [Database INSERTs and SELECTs Are Different](#)
 self argument, [Method Invocation: What Actually Happens, Are You Serious About “self”?, Prefix Your Attribute Names with “self”](#)
 sequence of objects, [Iterating Over a Sequence of Objects, Creating Sets Efficiently](#)
 server error messages, [Understanding HTTP Status Codes](#)
 session dictionary (flask module), [Flask’s Session Technology Adds State](#)
 sessions
 about, [It’s Time for a Bit of a Session](#)
 managing logins/logouts with, [Managing Logins with Sessions](#)
 state and, [Flask’s Session Technology Adds State](#)
 set built-in function, [Creating Sets Efficiently, Returning More Than One Value, Creating Another Function, 2 of 3](#)
 set comprehensions, [That Feeling You Get...](#)
 setdefault method, [Substituting “not in” for “in”](#)
 sets
 about, [A Data Structure That Avoids Duplicates: Set, Sets Don’t Allow Duplicates](#)
 combining, [Taking Advantage of Set Methods](#)
 commonality between, [Taking Advantage of Set Methods, intersection Reports on Commonality](#)
 creating efficiently, [Creating Sets Efficiently](#)
 creating from sequences, [Creating Sets Efficiently](#)

[difference between, Taking Advantage of Set Methods, difference Tells You What's Not Shared](#)

[duplicate objects and, A Data Structure That Avoids Duplicates: Set, Checking for Membership with "in", Sets Don't Allow Duplicates](#)

[empty, Returning More Than One Value](#)

[spotting in code, Sets Don't Allow Duplicates](#)

[setup function \(setuptools module\), Creating the Required Setup Files](#)

[setuptools module, Getting a Module into Site-packages](#)

[single-object tuples, Watch Out for Single-Object Tuples](#)

[site-packages, How Are Modules Found?, ImportErrors Occur No Matter the Platform](#)

[sleep function \(time module\), Extending Our Program to Do More, Arranging to Pause Execution](#)

[slice notation](#)

[for loop and, Python's "for" Loop Understands Slices](#)

[lists and, List Slices in Action, Slicing a List Is Nondestructive](#)

[__slots__ directive, 3. More on Object Orientation](#)

[sorted built-in function](#)

[about, 5. Getting Things Sorted](#)

[dictionaries and, Dictionaries: What We Already Know](#)

[sets and, Sets Don't Allow Duplicates, union Works by Combining Sets](#)

[spaces versus tabs, Is Indentation Driving You Crazy?](#)

[split method, What's Joined Together Can Be Split Apart, Processing Data: What We Already Know, Reading CSV Data As Lists, Stripping, Then Splitting, Your Raw Data](#)

[sqlalchemy module, 7. More Data Sources](#)

[SQLException exception, Raising an SQLException](#)

[SQL injection \(SQLi\), Web Attacks Are a Real Pain, Your Function Calls Can Fail](#)

[SQLite, Database-Enabling Your Webapp](#)

[square brackets \[\]](#)

[comprehensions and, How to Spot a Comprehension](#)

[dictionaries and, Insertion Order Is NOT Maintained, Accessing a Complex Data Structure's Data](#)

[lists and, Data Structures Come Built-in, A List Is an Ordered Collection of Objects, What About Using Square Brackets?, Square Brackets Are Everywhere, Slicing a List Is Nondestructive](#)

[tuples and, Tuples Are Immutable](#)

[standard library](#)

[about, Functions + Modules = The Standard Library, Functions + Modules = The Standard Library, Reusing Code with Functions](#)

[additional information, Batteries Included, 6. More from the Standard Library, 8. GUIs with Tkinter \(and Fun with Turtles\)](#)

[cautions adding/removing modules, Getting a Module into Site-packages](#)

[concurrency options, Doing More Than One Thing at Once](#)

[identifying locations, How Are Modules Found?](#)

[usage examples, Executing Code, One Statement at a Time, Functions + Modules = The Standard Library](#)

[start value, Asking the Interpreter for Help on a Function, Lists Understand Start, Stop, and Step, Starting and Stopping with Lists](#)

state (see [attributes \(state\)](#))

statements

- [assignment, Data Structures Come Built-in](#)
- [control, What Happened to My Curly Braces?](#)
- [displaying output, Returning to the Python Shell](#)
- [empty, Creating Objects from Classes](#)
- [reusability of, Consider What You're Trying to Reuse](#)
- [terminating with Enter key, Returning to the Python Shell](#)
- [try...except, Catching an Error Is Not Enough, Silently Handling Exceptions, Does "More Errors" Mean "More excepts"?](#)

[@staticmethod decorator, 3. More on Object Orientation](#)

[status codes \(HTTP\), Understanding HTTP Status Codes](#)

[step value, Asking the Interpreter for Help on a Function, Lists Understand Start, Stop, and Step, Stepping with Lists](#)

[stop value, Asking the Interpreter for Help on a Function, Lists Understand Start, Stop, and Step, Starting and Stopping with Lists](#)

storing data

- [in databases and tables, Advance Praise for Head First Python, Second Edition](#)
- [in data structures, Advance Praise for Head First Python, Second Edition](#)
- [in text files, Python Supports Open, Process, Close](#)

[strftime function \(time module\), Functions + Modules = The Standard Library](#)

strings

- [assigning to variables, Numbers, Strings...and Objects](#)
- [empty, Functions Return a Result](#)
- [formatting, 4. Formats for Strings and the Like](#)
- [iterating a specific number of times, Iterating Over a Sequence of Objects](#)
- [joining together, What About Using Square Brackets?, Viewing the Entire Log in Your Webapp, What's Joined Together Can Be Split Apart](#)
- [key/value pairs and, A Dictionary Stores Key/Value Pairs](#)
- [quotation marks and, List Slices in Action, What's the Deal with All Those Strings?](#)
- [splitting apart, What's Joined Together Can Be Split Apart](#)
- [turning into list of letters, Starting and Stopping with Lists](#)
- [whitespace and, Stripping, Then Splitting, Your Raw Data](#)

[strip method, Stripping, Then Splitting, Your Raw Data](#)

[submodules, Executing Code, One Statement at a Time](#)

[success messages, Understanding HTTP Status Codes, Redirect to Avoid Unwanted Errors](#)

[sudo command, Install the Testing Developer Tools, Let's Install Flask, Install Python 3 on Linux](#)

suites of code

- [comments in, Introducing Functions](#)

embedded suites within, [Suites Can Contain Embedded Suites](#)

functions and, [Introducing Functions](#)

indentation levels and, [Suites Can Contain Embedded Suites](#)

indenting, [Deciding When to Run Blocks of Code, Iterating Over a Sequence of Objects, Indent Suites with Format...Indent Region, Is Indentation Driving You Crazy?](#)

looping, [Extending Our Program to Do More, Iterating Over a Sequence of Objects](#)

running multiple times, [Extending Our Program to Do More](#)

unindenting, [Indent Suites with Format...Indent Region](#)

super operators, [Deciding When to Run Blocks of Code](#)

syntax errors, [What Happens Next..., Use Your Editor When Working on More Than a Few Lines of Code, Creating Objects from Classes](#)

sys module, [Functions + Modules = The Standard Library, Haven't We Just Lost Something?](#)

T

tables, [Decide on a Structure for Your Log Data](#)

tables, [Task 4: Create Code to Work with Our Webapp's Database and Tables](#)

(see also dictionaries)

<table> tag, [Generate Readable Output With HTML](#)

tabs versus spaces, [Is Indentation Driving You Crazy?](#)

<td> tag, [Generate Readable Output With HTML](#)

template engines

about, [Building the HTML Form](#)

embedding display logic in, [Embed Display Logic in Your Template](#)

preparing to run code, [Preparing to Run the Template Code](#)

relating to web pages, [Templates Relate to Web Pages](#)

rendering from Flask, [Rendering Templates from Flask](#)

ternary operator, [Checking for Membership with "in"](#)

testing developer tools, [Getting Ready to Check PEP 8 Compliance, 9, It's Not Over 'Til It's Tested](#)

text files, saving data to, [Python Supports Open, Process, Close](#)

text mode, [Reading Data from an Existing File](#)

Thread class, [Doing More Than One Thing at Once](#)

threading library, [Doing More Than One Thing at Once](#)

threading module, [Doing More Than One Thing at Once, Don't Get Bummed Out: Use Threads, 7, Running Your Code Concurrently](#)

<th> tag, [Generate Readable Output With HTML](#)

time module, [Functions + Modules = The Standard Library, Extending Our Program to Do More, Arranging to Pause Execution](#)

tkinter library, 8, [GUIs with Tkinter \(and Fun with Turtles\)](#)

today function (datetime module), [Functions + Modules = The Standard Library](#)

trailing whitespace, [Is Indentation Driving You Crazy?](#)

<tr> tag, [Generate Readable Output With HTML](#)

True value, [Functions Return a Result](#)

try...except statements, [Catching an Error Is Not Enough, Silently Handling Exceptions, Does “More Errors” Mean “More excepts”?](#)

tuples

about, [Ordered Collections Are Mutable/Immutable, Making the Case for Tuples](#)

comprehensions and, [That Feeling You Get..., What About “Tuple Comprehensions”?](#)

empty, [Recalling the Built-in Data Structures](#)

lists and, [Ordered Collections Are Mutable/Immutable, Making the Case for Tuples](#)

single-object, [Watch Out for Single-Object Tuples](#)

spotting in code, [Making the Case for Tuples](#)

turtle module, [8. GUIs w ith Tkinter \(and Fun w ith Turtles\)](#)

type built-in function, [Making the Case for Tuples, Understanding CountFromBy's Representation](#)

TypeError exception, [Are You Serious About “self”? Pass any amount of argument data to dunder “init”, Providing Sensible Defaults for CountFromBy](#)

type hints (annotations), [Use Annotations to Improve Your Docs](#)

U

unindenting suites of code, [Indent Suites with Format...Indent Region](#)

union method, [Taking Advantage of Set Methods](#)

unittest module, [9. It's Not Over 'Til It's Tested](#)

unordered data structures, [An Unordered Data Structure: Dictionary](#)

URLs

function decorators and, [Decorating a Function with a URL, Exposing Functionality to the Web, Exposing Functionality to the Web, Displaying the Webapp's HTML Form, Handling Posted Data, Recap: We Need to Restrict Access to Certain URLs, Back to Restricting Access to /viewlog](#)

functions with multiple, [Functions Can Have Multiple URLs](#)

processing with generators, [Using a Generator to Process URLs](#)

processing with list comprehensions, [Using a Listcomp to Process URLs](#)

restricting access to, [Can We Now Restrict Access to URLs?, Recap: We Need to Restrict Access to Certain URLs, Back to Restricting Access to /viewlog](#)

V

van Rossum, Guido, [Python Code Is Easy to Read, BDFL: Benevolent Dictator for Life](#)

variables

assigning objects to, [Numbers, Strings...and Objects](#)

assigning values to, [Data Structures Come Built-in, Numbers, Strings...and Objects](#)

displaying values of, [Returning to the Python Shell](#)

dynamic assignment of, [Data Structures Come Built-in, Numbers, Strings...and Objects](#)

environment, [Functions + Modules = The Standard Library](#)

functions and, [Coping with Scoping](#)

global, [Your Web Server \(Not Your Computer\) Runs Your Code](#)

initializing values, [Initialize \(Attribute\) Values Before Use](#)

misspelling, [Don't Forget to Try the Beer Song Code](#)

scope of, [Coping with Scoping](#)

usage example, [Data Structures Come Built-in](#)

venv technology, [2. Virtual Programming Environments](#)

version, identifying, [Functions + Modules = The Standard Library](#)

vertical bar, [Log a Single Line of Delimited Data](#)

vim text editor, [2. Alternatives to IDLE](#)

virtualenv module, [2. Virtual Programming Environments](#)

virtual programming environments, [2. Virtual Programming Environments](#)

W

web applications, [Task 4: Create Code to Work with Our Webapp's Database and Tables](#)

(see also database-enabling webapps)

adding finishing touch, [Adding a Finishing Touch](#)

adding robustness to, [A Quick Recap: Adding Robustness](#)

automatic reloading, [Using Request Data in Your Webapp](#)

calculating data needed, [Calculating the Data We Need](#)

creating Flask webapp object, [Creating a Flask Webapp Object](#)

deploying with PythonAnywhere, [Pythonanywhere: Deploying Your Webapp](#)

edit/stop/start/test cycle, [Refining the Edit/Stop/Start/Test Cycle](#)

exception handling and, [View the Log Through Your Webapp. Databases Aren't Always Available, Your Function Calls Can Fail, Getting Back to Our Webapp Code, Silently Handling Exceptions](#)

exposing functionality to the Web, [Exposing Functionality to the Web](#)

function decorators, [Decorating a Function with a URL, Exposing Functionality to the Web, Exposing Functionality to the Web, Displaying the Webapp's HTML Form, Handling Posted Data](#)

functions with multiple URLs, [Functions Can Have Multiple URLs](#)

global variables and, [Your Web Server \(Not Your Computer\) Runs Your Code](#)

handling posted data, [Handling Posted Data](#)

how they work, [Let's Build Something](#)

HTML forms, [Building the HTML Form, Accessing HTML Form Data with Flask](#)

HTTP status codes, [Understanding HTTP Status Codes](#)

installing and using Flask, [Let's Install Flask](#)

preparing for the cloud, [Preparing Your Webapp for the Cloud](#)

producing results as HTML, [Producing the Results As HTML](#)

redirecting to avoid unwanted errors, [Redirect to Avoid Unwanted Errors](#)

request data in, [Using Request Data in Your Webapp](#)

restarting, [Exposing Functionality to the Web, We're Ready for a Test Run](#)

running behaviors, [Running Your Webapp's Behavior\(s\)](#)

running for first time, [Running Your Flask Webapp for the First Time](#)

stopping, [Exposing Functionality to the Web, We're Ready for a Test Run](#)

testing, [Exposing Functionality to the Web, We're Ready for a Test Run](#)

updating, [Reconsidering Your Webapp Code, 1 of 2](#)

viewing logs, [View the Log Through Your Webapp](#)

web development technologies, [5. Web Development Technologies](#)

what do we want them to do, [What Do We Want Our Webapp to Do?, Recall What We're Trying to Build](#)

what happens on the web server, [What Happens on the Web Server?](#)

web development technologies, [5. Web Development Technologies](#)

web servers

about, [The Web Is Stateless](#)

HTTP status codes and, [Understanding HTTP Status Codes](#)

webapp process and, [Let's Build Something, What Happens on the Web Server?, Your Web Server \(Not Your Computer\) Runs Your Code](#)

while loop, [Iterating Over a Sequence of Objects](#)

whitespace, [Is Indentation Driving You Crazy?, Understanding the Failure Messages, Stripping, Then Splitting, Your Raw Data, One Final Question](#)

WingWare IDE, [2. Alternatives to IDLE](#)

with statement

classes and, [You've Seen This Pattern Before, Hooking into the "with" Statement, Consider What You're Trying to Do, Revisited](#)

context management protocol and, [You've Already Seen a Context Manager in Action](#)

exception handling and, [The DBcm Module, Revisited, Handling SQLError Is Different](#)

open, process, close technique and, [Reading Data from an Existing File](#)

split method and, [Stripping, Then Splitting, Your Raw Data](#)

viewing logs through webapps, [View the Log Through Your Webapp](#)

wonder name, [Creating a Flask Webapp Object](#)

wraps function (functools module), [The Final Step: Handling Arguments](#)

X

xkcd webcomic, [The Zen of Python](#)

XSS (Cross-site Scripting), [Web Attacks Are a Real Pain, Your Function Calls Can Fail](#)

Y

year attribute (date.today), [Functions + Modules = The Standard Library](#)

Z

Zen of Python, [The Zen of Python](#)

ZIP files, [Creating the Distribution File](#)





PREV

[E. Getting Involved: The Python Community](#)

NEXT

[About the Author](#)[Recommended](#) / [Queue](#) / [History](#) / [Topics](#) / [Tutorials](#) / [Settings](#) / [Blog](#) / [Get the App](#) / [Sign Out](#)© 2017 Safari. [Terms of Service](#) / [Privacy Policy](#)