

Symbols

' (single quoted string delimiter) 39
 != (not equal) 100
 .pth files (additions to sys.path) 121
 .py files (Python source code files) 117
 .pyc files (compiled bytecode files) 118
 " character in strings 227
 () (empty tuple) 58
 {} (empty dictionary) 82
 @abstractmethod decorator 260
 @abstractproperty decorator 260–261
 @property decorator 199
 @staticmethod decorator 192
 *
 list multiplication operator 53
 multiplication operator 40
 unpacking tuples 59
 ' (single-quote character) 64
 \ 65
 \n (newline character) 39
 # (comment header) 37
 % operator 24
 + (concatenation operator), for lists 53
 = (assignment operator)
 for assigning to shelves 170
 for creating/assigning to a dictionary 82

Numerics

2to3 conversion tool 276–277
 diff file produced 278
 fixers for specific features 278
 options 278
 -w option to write to file 279
 3to2.py, converting Python 3.x back to 2.x 281

A

ABCMeta 260
 absolute pathnames. *See* pathnames
 absolute
 abstract base classes 258–262
 @abstractmethod decorator 260
 @abstractproperty decorator 260
 abstract method, instantiating class with 261
 creating 260
 MutableSequence 259
 type checking with 259
 abstract collection type
 Hashable 259
 Iterable 259
 Mapping 259
 MutableMapping 259
 MutableSequence 259
 MutableSet 259
 Sequence 259
 Sized 259
 abstract methods 261
 containing
 implementation 261
 instantiating class with 261
 abstract property 261
 __add__ special method
 attribute 251
 __all__ attribute, packages 240
 Alt-/, keyboard shortcut (keyword completion) 14
 Alt-N, keyboard shortcut (next line) 14
 Alt-P, keyboard shortcut (previous line) 14
 and (logical operator) 100, 154
 append method lists 48
 applications, distributing 145
 arguments. *See* command line or functions 105
 arithmetic operations 40
 arithmetic operators 19
 involving only integers 38
 arithmetic precedence 38
 array module 46
 ASCII
 characters, including in a string 65
 special characters 65
 assert statement (debug statement) 181, 266
 assert_ TestCase class 271
 assertAlmostEqual TestCase class 271
 assertEquals TestCase class 271
 assertFalse TestCase class 271

assertNotAlmostEqual TestCase class 271
 assertNotEqual TestCase class 271
 assertRaises TestCase class 271
 assignment 37–38
 associative arrays. *See* dictionaries 81
 attributes 117

B

BaseHTTPRequestHandler urllib.request module 294
 BaseHTTPRequestHandler http.server module 294
 __bases__ (finding what classes an object inherits from) 244
 batteries included 282
 binary data, reading 165–167
 binary mode, opening files in 163
 binary records (for reading and writing data) 166
 bindings (names to objects in namespaces) 123
 block structure 35–37, 96–99 indentation 36
 blocks. *See* block structure 96
 Boolean expressions 25, 99, 101
 Boolean operators 100
 Booleans 99–101 examples 20 introduction 40
 bound methods 188
 braces in block structure 36
 break statement 27 in for loops 92, 94 in while loops 91
 buffering, definition 161
 built-in namespace 123
 built-in operators 43
 __builtins__ (built-in module) 123 dictionary of built-in identifiers 126
 Button (widget) in Tkinter 219
 bytecode (.pyc or .pyo files) 118
 bytes 80 reading from a file (read) 162 writing a string to a file (write) 162
 bytes type 275

C

C compiler, needed for freeze tool 145
 C/C++ reading data files generated by (struct)struct 167 writing data files for (struct)struct 167
 cache definition 88 implementation example 89, 169
 Canvas (widget) in Tkinter 221
 capitalize function 72
 center function 72
 CGI, dictionary in WSGI application 296
 CGIHTTPRequestHandler http.server module 294
 Cheese Shop. *See* Python Package Index
 __class__ attribute, obtaining the class of an instance 191
 class keyword 187
 class methods 192–194
 class variables access 191 creating 190 in class inheritance 197 using if instance variable not found 191 using to initialize instance variables 191
 classes abstract base classes 258–262 capitalization 187 class methods 192–194 class variables 190–192 creating 190 in inheritance 197 using if instance variable not found 191 using to initialize instance variables 191 constructor 187 cyclical references in instances of 205 defining and using 187–188 destructor methods for (__del__) 203–206 documentation strings for (__doc__) 193 dot notation, to access members 187 duck-typing 245 explicitly creating using a metaclass 257 finding the base classes of an instance (__bases__) 244 __getitem__ as marker of mutable sequence 259 inheritance 194–196 instance variables in 196 need to call __init__ explicitly 195 inheritance, multiple 207–208 initializing (__init__ method) 187 initializing with default parameters (__init__ method) 189 initializing with parameters (__init__ method) 189 instance variables 188 instantiation 187 isinstance function 244 subclass keyword 258 metaclasses 256–258 custom 258 method definition (def statement) 188 method invocation 188–189 methods 188 multiple inheritance hierarchy 207 namespaces 200 obtaining the class of an instance 191 private methods 197 private variable name mangling 198 private variables 197 begin with __ 198 properties marked with the @property decorator 199 setter decorator 199 shadowing class variable with instance variable 191 special method attributes 248 static methods 192–194 storing the class of an instance 244 subclassing built-in types 254–256 super function 195

- classes (*continued*)
 - superclass namespace 200
 - use of self 189
 - user-defined, type of 243
 - using superclass name explicitly instead of super 195
 - using to define structures 187
 - using to manage Tkinter application 219
 - close
 - for files 160
 - for shelves 170
 - cmath module 20
 - complex numbers and 42
 - collections library, abstract collection types 259
 - collections module, in standard library 283
 - comments
 - defined 29
 - differentiating 37
 - comparison operators 26
 - comparisons, compound 100
 - compile-time variable typing, lack of 8
 - complex numbers 41
 - advanced functions 42
 - examples 20
 - introduction 40
 - compound statements 96
 - context managers 184
 - continue statement 27
 - in for loops 94
 - in while loops 91
 - control flow structures 25–28
 - copy module 56
 - count method 70
 - lists 54
 - current working directory 149
 - cyclical references, breaking 205
- D**
-
- data members. *See* classes, instance variables 188
 - data serialization. *See* pickling 167
 - data type modules, in standard library 283
 - data types 19
 - converting to strings 23
 - dictionaries 24
 - file objects 25
 - lists 21–22
 - numbers 19–21
 - sets 24
 - strings 23
 - tuples 22
 - databases
 - accessing 291–293
 - DB-API 2.0 standard 291
 - sqlite3 library 291–293
 - close 292
 - commit 292
 - common operations 293
 - connect 293
 - connecting 291
 - creating cursor 292
 - cursor 293
 - execute 293
 - fetchall 293
 - fetchmany 293
 - fetchone 293
 - inserting 292
 - querying 292
 - DB-API 2.0 standard database interface 291
 - debug statements (assert) 181
 - `__debug__` variable 181, 266
 - setting false with PYTHONOPTIMIZE environment variable 266
 - decorator functions 113
 - deepcopy function, lists 56
 - def statement 28
 - def (method definition statement) 188
 - def keyword
 - defining methods 31
 - del (deletion statement)
 - deleting module bindings 124
 - deleting namespace entries 124
 - deleting variables 37
 - deletion of dictionary entries 84
 - del method
 - lists 49
 - `__delitem__` special method attribute 253
 - destructors 203–206
 - pitfalls 205
 - Python compared to C++ 204
 - development and debugging tools and runtime services, in standard library 286
 - dictionaries 24, 81–89
 - adding multiple entries to (update) 85
 - comparison to lists 82
 - comprehension 95
 - copying (copy, copy.deepcopy) 85
 - creating (=)
 - with initial entries 83
 - definition 82
 - deleting entries from (del) 84
 - delimiters [] 84
 - efficiency of 89
 - get method 24
 - implementing a cache with 88
 - key membership (in) 84
 - keys 24, 82–83
 - indices of 84
 - sorting 84
 - valid values 87
 - keys view (object) 84
 - key-value pairs in (items) 84
 - len function 24
 - methods 24
 - number of entries in (len) 83
 - order of values 82
 - representing sparse matrices with 88
 - retrieving values from (get) 85
 - table of operations 85
 - tuples as keys for 87
 - using to count words 86
 - values 24
 - values in (values) 84
 - view (object) 84
 - vs. shelves 171
 - why called dictionaries 83
 - dictionary, tuples, as keys 22
 - dir (display names in a module) 126
 - function 16
 - directives doctests, tweaking with 269
 - directories
 - changing (os.chdir) 150
 - creating (os.mkdir, os.makedirs) 156
 - deleting (os.rmdir) 156
 - getting the current working directory (os.getcwd) 150
 - listing the files in (os.listdir) 150, 155
 - nonempty, deleting (shutil.rmtree) 156

directories (*continued*)
 processing files in a directory
 tree
 (os.path.walk) 156–157
 renaming (os.rename) 155
 distutils package 145
 installing libraries 288
 division
 returning float 38, 40
 returning truncated
 integer 38, 40
 Django 295
 use of decorators in 114
 __doc__ (documentation string)
 definition 29, 104
 for built-in functions 127
 for classes 193
 vs. comment 104
 docstrings, testing code
 with 267
 doctest module, testmod 268
 doctests 267–270
 avoiding traps 269
 directives 269
 ELLIPSIS 269
 NORMALIZE_WHITESPACE 269
 pros and cons 270
 vs. unit tests 273
 documentation,
 Python 305–308
 downloading 308
 help command 306
 HTML, using pydoc 306
 on your computer 305
 online 305
 using pydoc 306
 web-based, using pydoc 307
 Windows Help file 308
 duck typing 245, 258

E

EAFP, easier to ask
 forgiveness 258
 else (statement)
 with exceptions 179
 with for loops 92
 with if-else constructs 91
 with while loops 91
 Emacs Python mode 13
 endian, converting when read-
 ing and writing data 167
 endswith method 71
 enumerate function 94

error handling
 example program 182
 exception mechanism 175
 possible approaches 173–175
 returning error status 173
 escape sequences 65–66
 hexadecimal 65
 numeric 65
 octal 65
 event handling in Tkinter 220
 events, virtual, in Tkinter 220
 except statement
 (exceptions) 28, 180
 exceptions 28, 172–184
 accessing multiple
 arguments 181
 AssertionError 177
 AttributeError 177
 BaseException 177
 BufferError 177
 BytesWarningException 177
 catching 176
 catching and handling (try-
 except-else-finally) 179
 defining new types of 180
 DeprecationWarning 177
 else statement 179
 EnvironmentError 177
 EOFError 177
 example application 183
 except statement 179
 Exception 177
 formal definition 175
 FutureWarning 177
 general concepts 173–176
 generating with raise
 statement 176
 GeneratorExit 177
 handlers 175
 handling, examples 28
 hierarchy of, in Python 177
 ImportError 177
 ImportWarning 177
 in Python 176–184
 IndentationError 177
 IndexError 177
 inheritance hierarchy, effect
 on catching of 182
 introduction 38
 IOError 177
 KeyboardInterrupt 177
 KeyError 177
 LookupError 177
 MemoryError 177

NameError 177
 NotImplementedError 177
 OverflowError 177
 OSError 177
 PendingDeprecationWarning
 177
 raising (raise) 178
 ReferenceError 177
 RuntimeError 177
 RuntimeWarning 177
 StopIteration 177
 SyntaxError 177
 SyntaxWarning 177
 SystemError 177
 SystemExit 177
 TabError 177
 try statement 179
 TypeError 177
 types of 177
 UnboundLocalError 177
 UnicodeDecodeError 177
 UnicodeEncodeError 177
 UnicodeError 177
 UnicodeTranslateError 177
 UnicodeWarning 177
 use of string argument 179
 user defined 176
 UserWarning 177
 ValueError 177
 VMSError (VMS) 177
 Warning 177
 where to use 184
 WindowsError
 (Windows) 177
 ZeroDivisionError 177
 executables, creating
 with py2app 145
 with py2exe 145
 with the freeze tool 145
 expandtabs function 72
 expressions
 Boolean 99, 101
 introduction 38
 extend method, lists 49

F

fail TestCase class 271
 false Boolean values 99
 Fast Fourier Transform 41
 file objects 25
 closing 160
 input function 25
 open statement 25
 opening 160

- file objects (*continued*)
 - os module 25
 - Pickle module 25
 - struct module 25
 - sys module 25
 - file objects. *See also* files 160
 - file path functions in standard library 284
 - filehandles. *See* file objects 159
 - fileinput module 133–134
 - fileinput.input (iterate over lines of input files)
 - fileinput 133
 - fileinput.lineno (total lines read in) 134
 - files 159
 - closing 160
 - obtaining the extension of (os.path.splitext) 152
 - opening (open) 160
 - opening in binary mode 163
 - processing files in a directory tree (os.path.walk) 157
 - reading with context managers 184
 - reading a line of (readline) 160
 - reading all lines of
 - file object as iterator 161
 - readlines 161
 - removing (os.remove) 155
 - renaming (os.rename) 155
 - write (write a string to a file) 162
 - writing a list of bytes to (writelines) 163
 - writing a list of strings to (writelines) 163
 - writing binary data to (write) 162
 - files and storage modules, in standard library 284
 - filesystems 147–158
 - See also* pathnames, files, directories 158
 - finally statement 179, 180, 206
 - find method 70
 - float (conversion function), for converting a string to a float 69
 - floating-point numbers. *See* floats
 - floats
 - arithmetic operations 40
 - examples 20
 - introduction 40
 - for loops 27, 92–94
 - break statement 27
 - continue statement 27
 - syntax for 92
 - formal string representation (repr) 75
 - format method 76
 - format specifiers 77
 - format string, as a template for a binary record (struct) 166
 - Frame (widget class), in Tkinter 214
 - freeze tool 145
 - C compiler required 145
 - from import * 42
 - from ... import *, controlled by __all__ 240
 - frozensets 61
 - creating (frozenset) 61
 - functions 103–114
 - accessing variable outside local scope 111
 - arguments 28
 - assigning to variables 111
 - built-in
 - list of 127
 - obtaining documentation strings of 127
 - overriding 127
 - decorator functions 113
 - definition of (def) 27, 103
 - documentation string of (__doc__) 104
 - generator functions 112
 - global variables in 109
 - local variables in 109
 - parameters 28, 105–109
 - default values 105
 - indefinite number of, by keyword 108
 - mixed passing techniques 108
 - mutable objects as 108
 - passing, by parameter name 106
 - positional 105
 - variable number of 107
 - return statement 28
 - return value of (return) 104
 - testing in interactive mode 108
 - variable scope 111
 - vs. procedures 104
 - __future__ module 280
- G**
-
- generator functions 112
 - geometry management in Tkinter 213
 - get (dictionary value retrieval method) 85
 - get method 24
 - __getitem__ special method
 - attribute 249, 253
 - as marker of mutable sequence 259
 - using to mimic list 250
 - GIMP Toolkit 222
 - Glade (graphical tool) 222
 - glob module, glob.glob (path-name pattern expansion) 139, 155
 - global namespace 123
 - global variables 109
 - globals function 16
 - graphics libraries, use of decorators in 114
 - Grayson, John, *Python and Tkinter Programming* 219
 - grid command in Tkinter 217
 - GUI development. *See also* Tkinter 209
 - GUI libraries for Python
 - cross-platform 221
 - Gtk (GIMP Toolkit) 222
 - Qt Package (of the KDE) 221
 - Tkinter 209
 - wxPython and wxWidgets 222
 - GUI principles, using Tkinter 212–214
- H**
-
- hashtables. *See* dictionaries 81
 - Hello, World program 15
 - help function 15
 - with variable name 15
 - help pages, generating 306
 - hexadecimal character representation (\xFF) 66
 - home scheme
 - installing libraries with 288
 - setup.py option 289
 - HTML formatting, for web applications 304
 - HTML wrapper 303
 - HTTP client, writing 294
 - HTTP status, in WSGI application 296
 - http.server module 293

I**IDLE**

Alt-/ keyboard shortcut (key-word completion) 14
 Alt-N keyboard shortcut (next line) 14
 Alt-P keyboard shortcut (previous line) 14
 choosing, vs. basic shell 14
 exiting a session 15
 for creating/editing a module 116
 indentation 37
 introduction to 13
 keyboard shortcuts 14
 Python Shell window
 starting on Mac OS X 13
 starting on UNIX/
 Linux 13
 starting on Windows 13
 using 14
 vs. basic interactive mode 12–14
 if statement 26
 if-elif-else statements 91
 imag 42
 imaginary numbers. *See* complex numbers 41
 immutability (non-modifiability), of numbers 87
 import packages 238
 import (bring in from a module) 119
 import statement 20, 29, 119
 from module import name 119
 imports, relative 239
 in for dictionaries 84
 in (key existence test), for shelves 171
 in (membership operator), for strings 74
 indentation 5, 96–99
 in block structure 35–37
 tabs vs. spaces in 97
 index method 70
 lists 53
 index notation
 for lists 46–48
 IndexError exception 251
 informal string representation (str) 75
 inheritance 194–196
`__init__` method 31

`__init__.py`, required for packages 238
`__init__.py` file
 executed on package import 239
 required for packages 240
 input (prompt for and read in a string) 163
 input function 25
 getting user input 43
 insert method
 lists 49
 vs. slice assignment 49
 installing Python modules 287
 other options 289
 prebuilt packages 287
 using home scheme 288
 using `setup.py` 288
 instance variables
 in class inheritance 196
 in classes 188
 int as dictionary key 87
 int (conversion function), for converting a string to an integer 69
 integer division 38, 40
 integers
 examples 19
 introduction 40
 Integrated Development Environment (IDLE). *See* IDLE
 interactive mode 12
 command history 13
 command prompt in (`>>>`) 15
 for Mac OS X 12
 for UNIX 13
 for Windows 12
 session
 exiting 13
 session, starting 12
 interactive prompt
 dir function 16
 help function 15
 using to explore Python 15
 internet protocols and formats, in standard library 286
 io library, StringIO 298
 is (identity operator) 101
 is not (identity operator) 101
`__isabstractmethod__` function
 attribute 261
 isinstance (built-in function) 244, 252, 259
 issubclass (built-in function) 245

items (dictionary contents method) 84
 iteration. *See* for loops, while loops 90

J

Java, difference in memory management for 205
 join method 67

K

keys (dictionary indices) 84
 values valid for 87
 keyword passing 106
 Kuchling, Andrew, regular expression tutorial 233

L

Label (widget) in Tkinter 219
 lambda expressions 111
 LBYL, look before you leap 258
 len (length function)
 for dictionaries 83
 for lists 46
 use in for loops with the range function 93
 len function 24
`__len__` special method
 attribute 253
 libraries
 adding 287
 in Python 8
 installing with home scheme 288
 installing with `setup.py` 288
 library modules 122
 Linux/UNIX
 absolute pathnames in 149
 relative pathnames in 149
 list, subclassing 254
 list (conversion function) 23
 converting string to list 60
 converting tuple to list 60
 for converting a string to a list 73
 list multiplication operator (*) 53
 lists 21–22, 46–57
 adding together 49
 appending element to (append) 48

- lists (*continued*)
 - bulk initialization of (*) 53
 - comparison to dictionaries 82
 - comprehension 95
 - concatenating 53
 - converting to tuples 23, 60
 - copying 48, 56
 - creating (=) 46
 - creating with the range function 93
 - custom sorting 51–52
 - disadvantages of 52
 - deep copy 56
 - deleting elements
 - by position (del) 49
 - by value (remove) 50
 - element types 21
 - empty 47, 58
 - finding the maximum value of (max) 53
 - finding the minimum value of (min) 53
 - functionality, implementing
 - with special method attributes 251
 - index notation 46–48
 - indexing 21
 - inserting elements into (insert) 49
 - matches in (count) 54
 - membership, determining (in) 52
 - methods, calling 22
 - modifying 48–50
 - nested 55–57
 - operations, summary of 54
 - reverse parameter 52
 - reversing (reverse) 50
 - searching (index) 54
 - shallow copy 56
 - slice notation 21, 47
 - sorting 50–52
 - in descending order 52
 - with a custom key function 51
 - typed 252
 - types of elements 46
 - writing to a file (pickle.dump) 167–170
 - ljust function 72
 - local namespace 123
 - locals (obtain the local namespace) 16, 124
 - logical operators 26, 100
 - look before you leap 258
 - loops. *See* for loops, while loops 90
 - lower function 72
 - lstrip method 69
- M**
-
- Mac OS X
 - Python launcher app for scripts 135
 - relative pathnames in 149
 - writing administrative scripts for 135
 - Macintosh
 - installing Python on 11
 - __main__ (script or interactive session name) 124, 125
 - map objects. *See* dictionaries 81
 - Maple 235
 - math functions, in standard library 284
 - math module 20, 41
 - complex numbers and 42
 - Mathematica 235
 - MATLAB 235
 - matrix
 - definition 88
 - representation using lists 55
 - sparse 88
 - memory management 203–206
 - message wall, creating 297–304
 - metaclasses 256–258
 - custom 258
 - explicitly creating a class using 257
 - methods
 - abstract 261
 - containing implementation 261
 - instantiating class with 261
 - basics 188
 - bound 189
 - invoking 188
 - unbound 189
 - modules 115–123
 - abc (Abstract Base Class) 260
 - accessing other definitions in same 117
 - collections library, abstract collection types 259
 - combining with scripts 141
 - creating 29
 - creating in IDLE 116
 - definition 115, 234
 - doctests 267
 - grouping, in packages 30, 235
 - importing from (import) 119
 - installing 287
 - library modules 122
 - module name vs. file name 118
 - private names in 121
 - reloading (reload) 118
 - search path for (sys.path) 119
 - using __all__ to control imports 119
 - using in scripts 118
 - using to eliminate name clashes 116
 - using underscore to make names private in 121
 - vs. programs and scripts 140–145
 - where to place 120
 - __mul__ special method attribute 253
 - multiline statements 98
 - multiple inheritance 207–208
 - addins 208
 - hierarchy 207
 - mixins 208
 - multiplication operator (*), for numbers 40
 - MutableSequence abstract base class 259
- N**
-
- __name__ (finding the class name of an object) 124, 244
 - named attributes
 - in Tkinter 212
 - in Tkinter, default values 213
 - NameError exception 38
 - namespaces 123–128
 - bindings in 123
 - built-in 123, 200
 - class namespace (self) 200, 203
 - displaying the built-in namespace (dir) 126
 - for class instances 199–203
 - for functions in interactive sessions 125
 - for functions in modules 125
 - for interactive sessions 123–124
 - for modules 125

namespaces (*continued*)
 global 123, 200
 instance namespace
 (self) 200, 202
 local 123, 200
 obtaining the local namespace
 (locals) 124
 overriding built-in
 functions 127
 superclasses (self) 200, 203
 with modules 116
 network programming 293–304
 newline character (\n) 39
 None value 43
 nonlocal keyword 110
 numbers 19–21, 40–43
 complex 42
 integer division operator (//) 40
 types, in Python 40
 numeric and mathematical mod-
 ules, in standard
 library 284
 numeric computation 41
 numeric functions
 built in 41
 in math module 41
 NumPy extension for numeric
 operations 41, 46

O

-O command-line option 266
 object reference counting 203
 object-oriented programming in
 Python 186–208
 See also classes 208
 objects
 converting strings to 74
 duck-typing 258
 finding class name of
 (__name__) 244
 giving full list capability 252
 making behave like
 lists 249–251
 type of (type function) 243
 writing to a file
 (cPickle.dump) 167–170
 OOP (object oriented
 programming) 30
 open (open a file) 159–161
 additional arguments 161
 create a new file object 159
 statement 25
 use of newline parameter 162
 open source software 7

operating system services, in
 standard library 285
 operators, built-in 43
 optparse module (parse com-
 mand-line arguments) 132
 or (logical operator) 100
 os module 25
 os.chdir (change
 directory) 150
 os.curdir (current directory
 indicator) 150, 153
 os.environ (environment
 variables) 154
 os.getcwd (get the current
 working directory) 150
 os.listdir (list the files in a
 directory) 150, 153, 155
 os.mkdir (create a
 directory) 156
 os.name (operating system
 name) 153
 os.path.basename (obtain the
 base file/directory) 152
 os.path.commonprefix (find
 the common prefix in a set
 of pathnames) 152
 os.path.exists (test the exis-
 tence of a pathname) 154
 os.path.expanduser (expand
 the user name
 variable) 153
 os.path.expandvars (expand
 the system variables) 153
 os.path.getatime (get atime of
 object pointed to by
 path) 155
 os.path.getmtime (get mtime
 of object pointed to by
 path) 155
 os.path.isdir (test if the path-
 name is a directory) 154
 os.path.isfile (test if the path-
 name is a file) 154
 os.path.ismount (test if the
 pathname is a filesystem
 mount point) 154
 os.path.issamefile (test if two
 pathnames point to same
 file) 155
 os.path.join (creating
 pathnames) 150–153
 os.path.split (splitting
 pathnames) 152
 os.path.splitext (obtain the
 file extension) 152

os.remove (delete a file) 155
 os.rmdir (delete a
 directory) 156
 os.walk (process files in a
 directory tree) 156

P

packages 234–241
 __all__ attribute of 240
 basic use of 234
 collections of related
 modules 235
 controlling imports with
 __all__ 240
 directory structure 236
 example of 235–240
 import statements in 239
 __init__.py file in 239
 loading subpackages and sub-
 modules of 238
 nesting 241
 private names vs. __all__ 241
 proper use of 241
 relative imports 239
 similarity to modules 238
 submodules 239
 using 238
 packages, defined 30
 packing binary data
 (struct.pack) 167
 pass statement 92
 PATH_INFO, web server gateway
 interface (WSGI) 301
 pathnames 148–155
 absolute 148
 in Linux/UNIX 149
 in Mac OS X 149
 in Windows 148, 151
 creating
 (os.path.join) 150–152
 expanding environment vari-
 ables in
 (os.path.expandvars) 153
 expanding username short-
 cuts in
 (os.path.expanduser) 153
 expanding wildcard charac-
 ters in 155
 expansion of (glob.glob) 155
 getting information about
 files 154
 manipulating 150–153
 obtaining common prefix of a
 set of
 (os.path.commonprefix) 152

- pathnames (*continued*)
 - obtaining the base of
 - (`os.path.basename`) 152
 - relative 148
 - in Linux/UNIX 149
 - in Mac OS X 149
 - in Windows 149, 151
 - separators 148
 - specialized queries 154
 - splitting 152
 - table of functions 157
 - testing existence of
 - (`os.path.exists`) 154
 - to network resources 152
- paths. *See* pathnames 147
- PEP (Python Enhancement Proposal) 8 309–321
- PEP 20, Zen of Python 321
- PEP-8 44
- Perl vs. Python 5
- persistent data. *See* pickling and shelves 171
- Peters, Tim
 - metaclasses 258
 - Zen of Python 241, 322
- Pickle module 25
 - in standard library 285
 - `pickle.dump` (write a Python object to a file) 167–170
 - `pickle.load` (read a Python object from a file) 167–170
- pickling (reading and writing Python objects from files) 167–170
- piping I/O between commands (`|`) 132
- Plone 295
- porting
 - Python 2.x to Python 3 274–276
 - problems 279
 - test coverage during 275, 279
- precedence
 - arithmetic 38
 - rules of 100
- print function 8, 15, 23
 - controlling output 79
 - redirecting output to file 164
- print statement 66
- printing
 - formatted strings with the `%` operator (`%`) 77
 - formatting sequences with named parameters 76
- private methods, in classes 197
- private names, in modules 121
- private variables
 - in classes, begin with `_` 198
 - methods of classes 197
- procedures
 - None value and 43
 - vs. functions 104
- properties creating 199
- PSF (Python Software Foundation) 7
- py2exe 145
- pydoc module
 - HTML help pages 306
 - using from command line 306
 - web-based
 - documentation 307
- PyPi 289
- Python
 - advantages of 3–6
 - coding style 43
 - contributing to 7
 - control flow 19
 - cross-platform 6
 - disadvantages 7–8
 - ease of use 4
 - expressiveness 4
 - high level of abstraction 4
 - indentation 5
 - installing 10–12
 - more than one version 11
 - on Macintosh 11
 - on UNIX 12
 - on Windows 95/98/NT 11
 - IronPython 6
 - legal restrictions on use, lack thereof 7
 - libraries included 6
 - library support 8
 - licensing 6
 - open source 6
 - origin 4
 - programs
 - distributing 145
 - Python 3
 - advantages of 8
 - incompatible with earlier versions 8
 - vs. earlier versions 8
 - readability 5
 - simple syntax 4
 - speed 7
 - support for GUIs 209
 - synopsis 19–30
 - variable typing, lack of 8
 - vs. Java, variable swap 5
 - vs. Perl 5
- Python 2.6 and `-3` switch, needed for porting 275–276
- Python 2.x
 - 2to3 conversion tool and Python 3 276–277
 - converting Python 3.x code back to 2.x 281
 - `dict.keys()` returns list 275
 - integer division 275
 - long integer type 275
 - migrating to Python 3.x 274–281
 - porting to Python 3 276
 - common problems 279
 - print statement 275
 - `raw_input` 275
 - StandardError 275
 - unicode type 275
 - using same code for, and Python 3 280
 - vs. Python 3 275
 - vs. Python 3.x
 - integer vs. float division 275
 - methods returning lists vs. dynamic views 275
 - normal and long ints vs. all ints long 275
 - print statement vs. print function 275
 - `raw_input` vs. `input` 275
 - StandardError exception vs. Exception exception class 275
 - unicode string type 275
- Python 3.x
 - bytes type 275
 - using same code for, and Python 2 280
- Python 3000 8
- Python and Tkinter Programming* (John Grayson) 219
- Python manual of style 309–321
- Python Package Index (PyPI) 289
- Python, Zen of 321
- PYTHONOPTIMIZE environment variable 181, 266
- PYTHONPATH environment variable 121
- PYTHONUNBUFFERED (binary unbuffered IO) 138
- PyUnit 270

Q

Qt package 221
 qualification 117

R

raise (statement) 178
 range (create a list sequence) 93
 setting starting and stepping values 93
 use in for loops 93
 raw strings 228
 re (regular expression) library 23
 re module 23, 70
 re.compile (create and compile a regular expression) 226
 read
 a binary record from a file 166
 a fixed amount from a file 162–163
 read (read contents of file as bytes object) 162
 readline (read a line from a file) 160
 readlines (read all lines of a file) 161
 real 42
 regular expressions 225–233
 advantages of using re.compile 226
 definition of 226
 example of 226
 extracting matched text with 229–231
 extracting text from a string with (?P) 230
 function for substituting text in strings with (sub) 232
 grouping () 226
 in standard library 283
 matching a digit \d 229
 matching one or more times + 229
 matching optional ? 230
 or | 226
 raw strings and 227
 special characters in 226
 splitting into sections 230
 strings, searching with 70

 substituting text in strings with (sub) 232
 relative pathnames. *See* pathnames, relative
 reload (reload a module) 118
 using to reload imported module 30
 remove method, lists 50
 replace method 72
 repr (convert an object to a string) 74, 76
 return statement 28
 reverse method, lists 50
 rfind method 70
 rindex method 70
 rjust function 72
 rowconfigure (command), in Tkinter 218
 rstrip method 69
 rules of precedence 100

S

scoping rules for Python 123–128, 199–203
 scripts 130–146
 combining with modules 141
 command-line arguments for 131
 controlling functions purpose of 144
 to catch exceptions 144
 to check command-line parameters 144
 to handle special modes 144
 to map output 144
 double-click starts in interpreter directory on Windows 139
 execution options on Windows 135–138
 making executable on Mac OS X 135
 making executable on UNIX 135
 on UNIX/Linux
 grp module for accessing group database 135
 pwd module for accessing group database 135
 resource module for accessing group database 135
 stat module for accessing group database 135
 syslog module for accessing group database 135
 redirecting input and output for 131
 standard structure of 130
 starting from a command line 130
 starting from a command window 137
 starting from a Mac OS X command line 130
 starting from a Windows command prompt 130
 starting from the Windows Run box 137
 starting in Windows by opening (double-click) 136, 140
 UNIX vs. Windows 138–140
 use as modules 142
 use for module regression testing 144
 use of `__main__` 142
 Scrollbar (widget), in Tkinter orientation 218
 placement 217
 sticky attributes 218
 search (search a string for a regular expression match) 226, 230–231
 self variable 31, 200
 use of in classes 189
 sequence creation (range) 93
 sequence object types, immutable. *See* strings, tuples, and sets 45
 setdefault (dictionary method) 85
`__setitem__` special method attribute 251
 sets 24, 60–61
 creating (set) 24, 61
 frozenset type 61
 in keyword 24
 operations 60
 setter decorator (`@method.setter`), in classes 199
 setup.py
 installing libraries with 288
 use with distutils 145
 shelve module
 close to ensure data written to file 171
 only strings a keys for 171
 shelve.open (open/create a shelve) 170

- shelves 170–171
 - closing (close) 170
 - key membership (has_key) 171
 - valid keys for (only strings) 170
 - vs. dictionaries 171
- slice, defined 21
- slicing, defined 47
- sort method, lists 50
- sorted function 52
- special characters 64–66, 227
 - in ASCII 65
 - in regular expressions 226
- special characters for regular expressions 227
- special method attributes 248
 - __getitem__ 249
 - making objects behave like lists 249–251
 - when to use 256
- speed, of Python 7
- split function 67, 73
- SQLite interface, in standard library 285
- sqlite3 database. *See* databases
 - sqlite3 library
- standard error (sys.stderr) 163
- standard input (sys.stdin) 164
 - redirecting 164
- standard library 282–287
 - data types 283
 - development and debugging
 - tools and runtime services 286
 - files and storage 284
 - http.server module 293
 - internet protocols and formats 286
 - internet protocols, handling 293
 - numeric and mathematical modules 284
 - operating system services 285
 - string services 283
- standard output (sys.stdout), redirecting 164
- startswith method 71
- statements
 - compound 96
 - splitting across multiple lines (\) 98
- static methods 192–194
- str (convert an object to a string) 75
 - __str__ method 31
 - __str__ special method
 - attribute 248
 - defining 248
- string module
 - constants 73
 - string.capitalize (convert a string to uppercase) 72
 - string.center (centers a string) 72
 - string.expandtabs (remove tab characters from a string) 72
 - string.find (find a substring in a string) 70
 - string.index (find a substring in a string) 71
 - string.join (join strings) 67
 - string.ljust (left-justify a string) 72
 - string.lower (convert a string to lowercase) 72
 - string.maketrans (translate characters in a string) 72
 - string.replace (replace substrings in a string) 72
 - string.rfind (find a substring in a string) 71
 - string.rjust (right-justify a string) 72
 - string.split (split a string) 67–68
 - string.strip (strip white space off both ends of a string) 70
 - string.swapcase (swap character case in a string) 72
 - string.title (capitalize all words in a string) 72
 - string.translate (translate characters in a string) 72
 - string.zfill (pads a numeric string with zeros) 72
- string modulus (%)
 - operator 77–80
 - formatting sequences 77
- string services modules, in standard library 283
- string.digits constant 73
- string.hexdigits constant 73
- string.letters constant 74
- string.lowercase constant 74
- string.maketrans function 72
- string.octdigits constant 73
- string.translate function 72
- string.uppercase constant 74
- string.whitespace constant 73
- strings 23, 39, 63–77
 - automatic concatenation
 - of 99
 - basic 39
 - basic operations 64
 - concatenation (+) 64
 - concatenation (string.join) 67
 - conversion to a number (int, long, float) 69
 - converting objects to (repr, str) 74, 76
 - counting occurrences in (string.count) 71
 - delimiters
 - double quoted 39
 - single quoted 39
 - triple quoted 39
 - evaluating 67–74
 - extracting matched text from 229–231
 - format method 76–77
 - with named parameters 76
 - formatting 78
 - formatting sequences 77–80
 - with named parameters 76, 78–79
 - formatting with % 77–80
 - function to create text when substituting text in with regular expressions (sub) 232
 - immutability of 23, 64
 - including ASCII characters in 65
 - including Unicode characters in 66
 - introduction 39
 - length of (len) 64
 - matching single element in a regular expression in (search) 227
 - methods 23, 67
 - modifying 71
 - with list manipulations 73
 - multiplication (*) 64
 - options for delimiting 23
 - raw 228
 - regular expression
 - grouping () 226
 - or | 226
 - optional match (?) 230

- strings (*continued*)
 - to extract text from
 - (?P) 230
 - to match a digit (\d) 229
 - to match one or more times
 - + 229
 - reporting qualities of 73
 - searching for a regular expression in (search) 230
 - searching (string.find, string.rfind, string.index, string.rindex) 70
 - searching, with re module 70
 - slice notation 63
 - splitting across lines 39, 99
 - splitting apart
 - (string.split) 67
 - table of operations 74
 - whitespace removal
 - (string.strip, string.lstrip, string.rstrip) 70
 - writing a string to a file
 - (write) 162
 - strip method 69
 - struct module (read and write
 - binary data) 25, 165–167
 - format string 166
 - struct.calcsize (calculate the size of a format string) 166
 - struct.pack (pack a binary record) 167
 - struct.unpack (parse data based on a format string) 166
 - structures
 - creating 187
 - using 187
 - style, Python
 - blank lines 310
 - code layout 310
 - comments 314–315
 - block 314
 - inline 314
 - documentation strings 314
 - imports 311
 - indentation 310
 - maximum line length 310
 - naming conventions 315–319
 - PEP (Python Enhancement Proposal) 8 309–321
 - programming
 - conventions 319–321
 - version bookkeeping 315
 - whitespace 311
 - subclassing
 - built-in types 254–256
 - UserDict 256
 - UserList 255
 - UserString 256
 - subpackages in packages 238
 - super function 195
 - swapcase function 72
 - symbol tables. *See* namespaces 123
 - SyntaxError, indentation errors 97
 - sys module 25
 - sys.path (search path for modules) 119
 - sys.platform (what platform are we on) 153
 - sys.prefix (module search path prefix) 121
 - sys.__stderr__ (original standard error) 164
 - sys.__stdin__ (original standard input) 164
 - sys.stdin, standard input 131
 - sys.stdout, standard output 131
 - syspath 119
- T**
-
- Tcl, Tk as GUI extension 210
 - test coverage, needed for porting 275, 279
 - TestCase class
 - assert_ 271
 - assertAlmostEqual 271
 - assertEqual 271
 - assertFalse 271
 - assertNotAlmostEqual 271
 - assertNotEqual 271
 - assertRaises 271
 - fail 271
 - unittest class 270
 - testing 265–273
 - assert statement 266
 - avoiding doctest traps 269
 - doctests 267–270
 - need for 266
 - unit tests 270–273
 - with Python 2.6 and -3 276
 - TestLoader class 272
 - TestRunner class 272
 - TestSuite class 272
 - unittest class 270
 - Text (widget) in Tkinter 221
 - text file, analyzing, example program 101
 - text input, prompting for (raw_input) 163
 - third-party modules 123
 - title function 72
 - Tk, GUI extension of Tcl 210
 - Tkinter 209–221
 - advantages 210
 - alternatives to 221
 - cross-platform support 210
 - direct mapping of Tk widgets to Python classes 212
 - event handling in 220
 - example application 214–215
 - Frame (widget class) 214
 - geometry management for 213
 - grid command 217
 - grid geometry manager 214
 - GUI development library for Python 209
 - installing 210
 - integration into Python 210
 - mouse events in 220
 - named attributes 212
 - default values 213
 - pack geometry manager 214
 - place geometry manager 214
 - principles 212–214
 - quick development time 210
 - rowconfigure (command) 218
 - sources of further information 219
 - Tk interface module 211
 - Toplevel (widget class) 214
 - ttk widgets 210
 - using classes to manage 219
 - virtual events in 220
 - widgets 212
 - attributes 215
 - Button 219
 - Canvas 221
 - constructor arguments 213
 - creating 215
 - hierarchy 215
 - Label 219
 - parent 216
 - placement 214–218
 - relative placement 217
 - Scrollbar
 - orientation 218

Tkinter (*continued*)
 placement 217
 sticky attributes for 218
 Text 221
 window events in 220
 Tkinter package 107
 Toplevel (widget class) in
 Tkinter 214
 tracebacks 38
 True (Boolean value) 26, 99
 try statement 28, 179
 try-except statement 179
 try-except-finally-else
 statement 28
 try-finally statement
 (finalizer) 206
 ttk widget set in Tkinter 210
 tuple (conversion function) 23, 73
 converting list to tuple 60
 tuples 22, 57–60
 as dictionary keys 22
 as keys for dictionaries 87
 concatenating (+) 57
 converting to lists 23, 60
 copying 58
 creating (=) 57
 element types 22
 immutability of 57
 index notation 57
 methods 22
 one-element, comma in 58
 packing and
 unpacking 59–60
 with list delimiters 60
 parentheses and 58
 reading from a file
 (pickle.load) 167
 unpacking of, in for loops 94
 unpacking to make for loops
 cleaner 94
 unpacking, extended (*) 59
 writing to a file
 (pickle.dump) 168
 TurboGears 295
 type (type finding function) 242
 type coercions 38
 type conversions
 any Python object to a string
 (repr, str) 74
 integer to float (float) 40
 string to float (float,
 string.atof) 69
 typed list 252
 TypedList 252, 267

types
 as objects 242–246
 built-in, subclassing 254–256
 bytes 275
 checking 243
 comparing 243
 duck-typing 245
 in the standard library 283
 obtaining the class of an
 object (`__class__`) 244
 of user-defined classes 243
 type objects 243
 typing, dynamic 8

U

Unicode
 character representation
 (`\N{LATIN SMALL LETTER A}`) 66
 characters, including in
 strings 66
 unit tests 270–273
 creating 270
 running 272
 running multiple tests 272
 vs. doctests 273
 unittest module 270
 UNIX
 installing Python on 12
 writing scripts for 135
 update (dictionary update
 method) 85
 upper function 72
 URL, parsing 300
 urllib.parse, `uquote_plus`
 function 301
 urllib.request module 294
 urllopen, urllib.request
 module 294
 user input 43
 converting to int or float 43
 prompt string 43
 user-defined classes, type of 243
 UserDict, subclassing 256
 UserList, subclassing 255
 UserString, subclassing 256

V

ValueError exception 68
 values (dictionary method) 84
 van Rossum, Guido, PEP 8 309

variables
 assigning (=) 37–38
 creating (=) 37
 deleting (del) 37
 global 109
 local 109
 names 38
 view (object), with
 dictionaries 84
 virtual events, in Tkinter 220

W

Warsaw, Barry, PEP 8 309
 web application
 advanced, creating with
 frameworks 296
 basic, creating with wsgi
 library 295
 creating in Python 295–297
 WSGI 295
 web frameworks 297
 web programming 293
 web server gateway interface
 (WSGI)
 environ,
 CONTENT_LENGTH 298
 handle_request 296
 HTML formatting 304
 make_server 296
 PATH_INFO 301
 REQUEST_METHOD 298
 sample application 297–304
 serve_forever 296
 web server gateway interface
 (WSGI) specification 295
 schematic diagram 295
 wsgiref module 296
 web2py 295
 When 266
 which 154
 while loops 26, 90
 break statement 26
 whitespace
 in block structure 35
 removing 69
 widgets, in Tkinter 212
 constructor arguments 213
 creating 215
 placement 215–218
 relative placement 217
 Windows
 absolute pathnames in 148,
 151

- Windows (*continued*)
 - adding .py as recognized extension 138
 - creating shortcut for script in 136
 - relative pathnames in 149, 151
 - starting script with double click 140
 - using .pyw extension to avoid opening command window 136
 - Windows 95/98/NT, installing Python on 11
 - with statement 184
 - word-counting example 86
 - working directory, current 149
 - wrapper class
 - UserDict 256
 - UserList 255
 - UserString 256
 - write (write a string to a file) 162
 - writelines (write a list of strings to a file) 163
 - WSGI 295
 - wsgi.input, reading from 298
 - wsgiref module,
 - simple_server 295
 - wxPython toolkit 222
 - wxWidgets framework 222
-
- Z**
- Zen of Python (Tim Peters) 321
 - flat is better than nested 241
 - zfill function 72
 - zip function 95
 - Zope 295
 - Zope 3 project, use of doctests 270