

Index

Symbols

- = assignment operator, 192
 - += augmented assignment operator, 115, 192–193
 - [] (brackets)
 - list literal, 120
 - listcomps (list comprehensions), 262
 - regular expressions, 303
 - ^ (caret)
 - regular expressions, 304
 - syntax errors, 53
 - , (comma)
 - in separating arguments, 279
 - in tuple creation, 122
 - + concatenation operator, 93–94
 - @ (decorator), 267–268
 - { } dict literal, 140
 - (difference) operator, 155
 - / (division) operator, 50, 112
 - \$ (dollar sign) in regular expressions, 304
 - . (dot) character in regular expressions, 303
 - _ _ (double underscore characters), 219
 - () empty tuple, 121–122, 163
 - == equality test, 48, 166
 - = (equals sign), 22, 40, 51
 - \ (escape character), 91–92, 304
 - ** exponentiation operator, 47
 - > (greater than) operator, 48
 - // integer division operator, 50, 113
 - % interpolation operator for string formatting, 101–105
 - & intersection operator, 156
 - < less than operator, 48
 - % modulo remainder operator, 50
 - * multiplication operator, 47
 - ** operator, for dictionaries, 195
 - * operator, lists and tuples, 195
 - != or <> not equal to operator, 48
 - () parentheses
 - grouping operator, 47
 - regular expression groups, 304, 306–307
 - + (plus) addition operator, 47
 - # (pound sign), 51, 84
 - r. *See* raw strings
 - * repeating operator, 93–94
 - ; (statement separator), 41
 - ^ symmetric difference operator, 53, 156
 - """ or ''' (triple quote), 90
 - u. *See* Unicode strings
 - _ (underscore character), 20–21, 40
 - | union operator, 156
- ## • A •
- absolute importing, 211–212
 - absolute pathnames, 285
 - accessing module attributes/code, 202–203
 - ActiveState Programmer Network, 379
 - _ _add_ _(), 232
 - add_header(), 351
 - addition operator (+), 47
 - all(), 164
 - allow_fragments parameter, 341
 - and operator, 49, 164–165
 - anonymous functions, 258, 269
 - any(), 164
 - append(), 46, 125, 132, 134–136
 - *args, 194, 287–288. *See also* positional parameters
 - arguments. *See also* parameters
 - classes, 220
 - defined, 184
 - exception, 251
 - functions, 23
 - instances, 220
 - keyword arguments (**kwargs), 150, 190, 194
 - methods, 46
 - passing, 189–190
 - positional, 190
 - separating, 279
 - unpacking, 195–196
 - arithmetic operators, 47
 - as. *See* import statement; with statement
 - assignment (=) operator, 192
 - associative arrays, 141
 - associative memories, 141
 - as_string(), 353
 - attach(), 352
 - AttributeError message, 279
 - attributes
 - classes, 218, 221–222, 224–226
 - getting, 278–279
 - instances, 218, 222, 224–226

attributes (*continued*)

modules, 31, 203

`__mro__`, 245

new-style classes, 240–241

objects, 214, 275, 278–279

parameter attributes, 220

private, 226

setting, 278–279

`__slots__` class attribute, 244

viewing, 278

augmented assignment operator (`+=`), 115, 192–193

automatic conversion of numbers, 113

• B •

backslash (`\`) escape character, 91–92

bare except, 251

bare functions or methods, 66

base classes, 215, 227–229. *See also* classes

`BaseException` class, 265–266

`basicConfig()`, 297–298

Beautiful Soup tool, 344

Berkeley DB database library, 333

`bin` parameter, 333

binary floating point, 113

binding names to objects, 40

blocks. *See* code blocks

body. *See* code blocks

Boolean data type, 162–164

Boolean expression errors, 76

Boolean operators, 47–49, 162, 164–165

Borg Pattern, 373

braces (`{ }`) dict literal, 140

brackets (`[]`)

list literal, 120

listcomps (list comprehensions), 262

regular expressions, 303

`break` keyword, 177

break points, 86

bugs (defined), 73, 75

building

dictionary, 149–151

list, 134–135

built-in functions, 23

built-in help system, 27–28

`__builtin__` module, 274

`__builtins__` module, 24–25

• C •

cache storage with dictionaries, 152–153

calculators, 17, 22–23, 116–118

`call()`, 287–288

call stack, 249

calling

class methods, 242

classes, 222

functions, 23–24, 184

generators, 260

methods, 45–46, 222–223

unbound methods, 223

`capitalize()`, 97

caret (`^`)

regular expressions, 304

syntax errors, 53

case manipulation in strings, 97

`center()`, 97

CGI (Common Gateway Interface), 356

`cgi` module, 356–361

CGI scripts, 356–361

`cgitb` module, 361

`cgi.test()`, 360

character encoding, 107

checking elements in sets, 155

Cheese Shop, 377–378

child classes. *See* subclasses

class attributes, 218, 221–222, 224–226

class factory, 246

class methods, 58, 241–243

`class` statement, 40, 63

classes. *See also* instances

arguments, 220

base classes, 215, 227–229

`BaseException`, 265–266

Borg Pattern, 373

calling, 222

conventions, 219, 226

creating, 213, 219, 234

data-only, 369

defined, 58, 216

`Exception`, 255–256, 265–266

inheritance, 215, 228–230

`__init__()`, 219–220

metaclasses, 245–246

namespaces, 58, 224, 230–231

new-style classes, 235–246, 265

old-style classes, 235

operator interception, 232–233

operator overloading, 232–233

`self` parameter, 219, 221, 223, 243

Spider, 63–64

subclasses, 215, 227–231

subclassing, 65

`type()`, 217

type/class unification, 236

`classobj()`, 246

`class_with_method()`, 246

- `close()`, 261, 277, 315, 322, 325
- closing files, 277
- `cls` parameter, 243
- `cmath` module, 115–116
- code
 - comments, 51–52, 81
 - decorator syntax, 267–268
 - docstrings, 52, 69, 81, 90–91
 - formatting, 81
 - indentation, 8, 25, 54
 - modules, 202
 - organizing, 201
 - pseudo-code, 73
 - pythonic code, 12
 - refactoring, 77–79
 - unpythonic code, 12
- code blocks
 - compound statements, 161
 - control structures, 54–55, 161
 - defined, 8, 53
 - interactive mode, 26
 - loops, 54–55
 - namespaces, 56–58
 - rules, 54
 - syntax, 54
- coding idioms. *See* idioms
- `collapse_rfc2231_value()`, 355
- combining strings, 93, 97
- comma (,)
 - in separating arguments, 279
 - in tuple creation, 122
- command line, 29–30
- comments, 51–52, 81, 84
- Common Gateway Interface (CGI), 356
- community. *See* Python community
- comparing
 - sequences, 122–123
 - sets, 155
 - strings, 95
 - words, 370
- comparison operators, 47–48, 93, 95, 165–166
- comparisons, 162, 165–167
- `compile()`, 307–309
- compiling regular expressions, 307–309
- `comp.lang.python` newsgroup, 76, 377, 379
- `comp.lang.python.announce` newsgroup, 380
- complex numbers, 42, 114–116
- compound data types, 119
- compound statement, 161
- compressed files, 321–326
- concatenation (+) operator, 93–94
- conditional expressions, 266–267
- conditional operators, 47, 49
- conditions, 162
 - configuring loggers, 297–298
 - connections to databases, 327
 - constants, 25, 372
 - container objects, 119
 - context managers, 265
 - `continue` statement, 178
 - continuous loops, 178
 - control structures
 - code blocks, 54–56, 161
 - comparisons, 162, 165–167
 - conditions, 162
 - defined, 161
 - false objects, 163
 - nested, 161
 - syntax, 161–162
 - conventions
 - for classes, 219, 226
 - for code formatting, 81
 - for comments, 51–52
 - for docstrings, 187
 - for instances, 226
 - for modules, 205, 226
 - for names, 80–81
 - conversion type, 101
 - converting
 - data to a string, 96
 - formats of pathnames, 285
 - literal objects into tuples, 136–137
 - numeric data types, 113
 - objects to strings, 233
 - sequence into a dictionary, 149–150
 - tuples into a dictionary, 149
 - Unicode to an encoded string, 108
- `copy()`, 147, 320
- `copy2()`, 320
- copying
 - dictionary, 147
 - directories, 320
 - files, 320
 - list, 131–132
- `copymode()`, 320
- `copytree()`, 320
- coroutine. *See* generator
- cost of, 11
- `count()`, 96–97
- counting
 - occurrence of values in a list, 125
 - strings, 97
- `count(value)`, 125
- `cPickle` module, 330–333
- creating. *See also* writing
 - class attributes, 221–222
 - class methods, 242
 - classes, 213, 219, 234

- creating (*continued*)
 - dictionary, 140
 - directories, 284
 - exceptions, 255–256
 - generators, 263–264
 - instance attributes, 222
 - instances, 216–218, 222
 - iterators, 258, 260–261
 - list, 124, 134, 262–263
 - methods, 220–221
 - modules, 205
 - regular expressions, 302
 - StringIO object, 314
 - subclasses, 227
 - tables (databases), 328
 - cStringIO module, 313–314
 - curly braces ({}) dict literal, 140
 - current working directory, 283
 - custom exceptions, 265–266
- D •
- Daily Python URL blog, 379
 - data
 - converting to a string, 96
 - pickling, 330–333
 - sparse data, 139
 - unpickling, 333
 - data types. *See also* dictionary; files; list; numbers; sequential data types; sets; strings
 - Boolean, 162–164
 - compound, 119
 - defined, 41
 - high-level data types, 8
 - methods, 45–46
 - pickling restrictions, 331
 - spider.py program, 67
 - type/class unification, 236
 - values, 41
 - databases
 - connections, 327
 - DBM-style databases, 333–336
 - persistent databases, 330–331
 - Python Database API, 330
 - SQLite library, 326–330
 - table operations, 328–330
 - data-only classes, 369
 - date() object, 289
 - datetime module, 289–291
 - datetime() object, 289–290
 - DBM-style databases, 333–336
 - debugging
 - Boolean expressions, 76
 - break points, 86
 - bugs (defined), 73, 75
 - CGI scripts, 359–361
 - comments, 84
 - dir(), 82
 - IDLE, 37
 - importance of, 71
 - infinite loops, 76
 - logging module, 83–84, 296–299
 - misspelled names, 75
 - modules, 202
 - pdb module, 84–86
 - print statement, 83
 - PyChecker tool, 95
 - regular expressions, 312
 - repr(), 82–83
 - strategies, 82–83
 - syntax errors, 75
 - tracebacks, 83
 - type(), 82–83
 - values, 76
 - Decimal(), 117. *See also* numbers
 - decimal module, 112, 116–118
 - decimal numbers, 42, 116–118
 - decode(), 97, 108
 - decode_params(), 355
 - decode_rfc2231(), 355
 - decoding strings into Unicode, 108–109
 - decorate, sort, undecorate sort, 370–371
 - decorator (@), 267–268
 - dedent(), 315
 - deep copying, 131–132
 - def keyword, 269
 - def statement, 40, 63, 185–186, 220. *See also* functions; methods
 - default_scheme parameter, 341
 - default-value parameters, 66, 190–194
 - defining functions, 185–186
 - del keyword, 130, 147
 - deleting
 - directories, 284, 321
 - indentation from strings, 315
 - items from a list, 125
 - items from dictionaries, 147
 - depth-first, left-to-right searching, 230–231
 - designing programs, 71–75
 - dice-roller function, 366
 - dict(). *See* dictionary
 - dict keyword, 44
 - dict literal ({}), 140
 - dictionary
 - adding items, 148–149
 - building, 149–151
 - cache storage, 152–153
 - comparison with other data types, 140

- converting sequences into, 149–150
 - converting tuples into, 149
 - copying, 147
 - creating, 140
 - defined, 42, 44, 67
 - deleting items, 147
 - empty dictionary, 140
 - keyword arguments (****kwargs**), 150
 - loops, 145–147, 150–151
 - `os.environ` dictionary, 286
 - removing items, 147
 - retrieving items from, 144–145
 - shallow copying, 147–148
 - storing constants in dictionaries, 372
 - string formatting, 104–105
 - testing, 146
 - unpacking, 195–196
 - updating, 151
 - uses for, 139
 - values, 143–144, 151–152
 - dictionary keys, 140–144, 153
 - `difference()` operator, 155
 - `difflib` module, 370
 - `dir()`, 24, 82, 207, 275, 323
 - directories, 283–284, 320–321, 365
 - `disable()`, 299
 - disabling message logging, 299
 - disappearing list, 132
 - division, 50, 112. *See also* `Decimal()`
 - division (`/`) operator, 50, 112
 - docstrings, 52, 69, 81, 90–91, 187
 - `doctest` module, 293–296
 - Document Object Model (DOM), 348
 - documentation, 28–29, 207, 236, 274, 376, 384
 - dollar sign (\$) in regular expressions, 304
 - dot (.) character in regular expressions, 303
 - double underscore characters (`_ _`), 219
 - downloading Python, 383
 - Dr. Dobbs' Python-URL Web site, 379
 - DSU sort, 370–371
 - `dump()`, 332
 - `dumps()`, 330, 332
 - duplicate dictionary keys, 153
- E •**
- EAFP (Easier to Ask Forgiveness than Permission)
 - method of exception handling, 254
 - ElementTree XML implementation, 344–348
 - `elif` statement, 168–170
 - `else` clause
 - for loop, 177–178
 - if/else, 55, 169
 - try/else, 56
 - try/except/else, 251–252
 - while loop, 177–178
 - email module, 349–353
 - `email.Utils` module, 354–355
 - embedded Python, 389
 - embedding, 11
 - empty dictionary, 140
 - empty tuple (`()`), 121–122, 163
 - `encode()`, 97, 108
 - `encode_base64()`, 352–353
 - `encode_quopri()`, 352–353
 - `encode_rfc2231()`, 355
 - `Encoders` module, 352–353
 - encoding, 107
 - `endswith()`, 96–97
 - entering dictionary keys, 141
 - `enumerate()`, 135, 173, 179–180
 - environment variables
 - changing, 286
 - `os.environ` dictionary, 286
 - `PYTHONDOCS`, 28, 384
 - `PYTHONPATH`, 205–206
 - values, 286
 - `_eq_ _()`, 232
 - equality test (`==`), 48, 166
 - equals sign (`=`), 22, 40, 51
 - error checking, 69
 - error handling. *See* exception handling
 - errors
 - Boolean expressions, 76
 - bugs (defined), 73, 75
 - infinite loops, 76
 - misspelled names, 75
 - overflow errors, 114
 - syntax errors, 53, 75
 - values, 76
 - escape character (`\`), 91–92, 304
 - escape codes, 91–92, 304–305
 - `eval()`, 281
 - `except` clause, 56, 249–252 *See also* try statement
 - Exception class, 255–256, 265–266
 - exception handling
 - arguments, 251
 - EAFP method, 254
 - example, 253
 - functionality, 248
 - LBYL method, 254
 - multiple exceptions, 250
 - raise statement, 254–255
 - `spider.py` program, 69–70
 - tracebacks, 249
 - try/except block, 248–252
 - try/finally block, 248–249, 252–253
 - user input, 248

exceptions

- arguments, 251
- `BaseException` class, 265–266
- creating, 255–256
- custom exceptions, 265–266
- defined, 25, 247
- `Exception` class, 255–256, 265–266
- hierarchy, 256
- `KeyboardInterrupt`, 266
- logging details of, 251
- new-style classes, 265
- `PendingDeprecationWarning`, 266
- printing details of, 251
- raising, 143, 247, 254–255
- re-raising, 255
- saving, 255
- string exceptions, 266
- `SystemExit`, 266
- Web page errors, 338–339
- `execute()`, 329
- `expandtabs()`, 97
- exponentiation operator (`**`), 47
- expressions. *See also* statements
 - Boolean expressions, 76
 - conditional expressions, 266–267
 - defined, 41
 - interactive mode, 20–21
 - lambda expressions, 268–269
 - seeing the result of the last expression, 20
 - values, 20–21
- `extend()`, 125
- extending a parent class, 215, 228–229
- extensibility, 8
- Extensible Markup Language, 344–348
- extension module, 201
- Extreme Programming (XP), 78

● F ●

- false, 49
- false objects, 163
- `fetchall()`, 329
- files
 - closing, 277
 - compressed files, 321–326
 - copying, 320
 - file data type, 42, 45
 - HTML files, 342–344
 - listing, 365
 - moving, 320
 - opening, 275–276
 - permissions, 320
 - reading, 277
 - saving, 277
 - string buffers, 313
 - temporary files, 284
 - unpickling, 333
 - writing to, 277–278
 - XHTML files, 342–344
 - Zip files, 321–326
- `fill()`, 316–317
- `filter()`, 268, 270
- filtering a list, 263
- finally clause, 56, 249, 252–253. *See also* try statement
- `find()`, 96–97
- Find commands, 36
- `findall()`, 309
- `finditer()`, 309
- `find_links()`, 64–65
- first-class functions, 268
- flags for regular expressions, 309
- `float()`, 114. *See also* numbers
- floating point numbers, 42, 112–114
- floor division, 112
- `flush()`, 277
- for loop, 55, 170–173, 175, 177–178. *See also* generator; listcomps (list comprehensions)
- form data
 - CGI scripts, 357–358
 - security risks, 358
 - submitting, 340–341
- `formataddr()`, 354
- `formatdate()`, 355
- formatting
 - code, 81
 - strings, 97, 101–105
 - text, 315–318
- framework, 230
- from. *See* import statement
- `fromkeys()`, 149–150
- frozen sets, 156–157
- function decorator. *See* decorator (@)
- function wrapper. *See* decorator (@)
- functional programming, 214, 268
- functions. *See also* def statement; methods
 - anonymous, 258, 269
 - arguments, 23, 184, 189–190, 195–196
 - bare functions, 66
 - benefits of, 183–184
 - built-in, 23
 - call stack, 249
 - calling, 23–24, 184
 - decorators, 267–268
 - def statement, 185–186
 - defined, 23, 57, 185
 - defining, 185–186

- differences from methods, 66
- docstrings, 187
- first-class, 268
- generators, 258, 260–261
- help, 23
- higher-order, 268
- interactive mode, 23–24
- lambda expressions, 268
- namespaces, 57, 196–200
- naming, 186–187
- parameters, 184, 188–194
- references, 198
- regular expressions, 308
- return statement, 188
- values, 188
- `__future__` module, 112–113

• G •

- generator, 258, 260–264. *See also* `def` statement
- genexps (generator expressions), 261–264
- `getaddresses()`, 354
- `getattr()`, 278–279
- `__getattr__()`, 232, 238
- `__getattrattribute__()`, 238
- `get_close_matches()`, 370
- `getfirst()`, 357
- `getinfo()`, 323
- `__getitem__()`, 232
- `getlist()`, 357
- `get_payload()`, 353
- getting
 - attributes, 278–279
 - field values from tables, 329–330
- `geturl()`, 339
- `getvalue()`, 152, 315
- `glob()`, 365
- `glob` module, 365
- global namespaces, 197–200
- greater than (>) operator, 48
- `group()`, 312
- `groupdict()`, 312
- grouping operator (), 47
- grouping regular expressions, 304, 306–307
- `groups()`, 312
- `gzip` module, 325–326

• H •

- handling exceptions. *See* exception handling
- `hasattr()`, 278
- hashes, 141

- `has_key()`, 146
- help
 - built-in help system, 27–28
 - documentation, 28–29
 - functions, 23
 - IDLE, 35
 - interactive mode, 27–28
 - modules, 33, 207
 - online resources, 377
 - packages, 209
 - strings, 97
- hierarchy of exceptions, 256
- higher-order functions, 268
- high-level data types, 8
- high-level languages, 11
- hooks, 232
- HTML files, 342–344
- `htmllib` module, 342–344

• I •

- idioms, 257–258, 365–373
- IDLE (Interactive DeveLopment Environment)
 - debugging, 37
 - defined, 34
 - features, 15–16
 - Find commands, 36
 - help, 35
 - interactive mode, 34–35
 - Path Browser, 36
 - starting, 34, 385
 - text editor, 36
 - time-saving features, 35
- `if` statement, 55, 163, 167–168, 170. *See also*
 - conditional expressions
- `if/else` statement, 169
- imaginary numbers, 42, 114
- `import` statement, 62
- importing
 - absolute importing, 211–212
 - modules, 30–32, 57, 202–203, 274
 - package items, 209–212
 - relative importing, 211–212
- `in`. *See* `for` loop
- `in` comparison operator, 166
- `in` keyword, 96, 124
- indentation
 - code, 8, 25, 54
 - strings, 315
- `IndentationError` message, 54
- `index()`, 97, 129–130
- `IndexError` message, 99, 128

- indexing
 - list, 126–130
 - strings, 98–100
 - tuples, 126
 - Industrial Light & Magic, 10
 - info(), 339
 - infolist(), 323
 - inheritance, 215, 228–230, 237–238
 - __init__(), 64, 66, 219–220, 233
 - initializers, 219
 - initializing
 - modules, 202–203
 - programs, 62–63
 - input(), 279–280
 - insert(), 130
 - installing
 - documentation, 384
 - Python, 383–388
 - instances
 - arguments, 220
 - attributes, 218, 222, 224–225
 - calling methods, 222–223
 - class methods, 58
 - conventions, 226
 - creating, 216–218, 222
 - defined, 58, 64, 216
 - namespaces, 58, 224
 - new-style classes, 239
 - int(), 94. *See also* numbers
 - integer division, 50, 112–113
 - integer division (/) operator, 50, 113
 - integers, 42, 111–113
 - Interactive DeveLopment Environment. *See* IDLE
 - interactive mode
 - calculator, 17, 22–23
 - code blocks, 26
 - defined, 15–16
 - exiting, 26–27
 - expressions, 20–21
 - features, 16–17
 - functions, 23–24
 - help, 27–28
 - IDLE, 34–35
 - lists, 21–22
 - modules, 32–33
 - namespace, 24–25
 - object information, 19–20
 - quitting, 26–27
 - rules, 18
 - starting, 17
 - strings, 21–22
 - writing programs, 25–26
 - internal representation of strings, 93
 - interpolation operator (%), 101–105
 - interpreter. *See* interactive mode
 - intersection (&) operator, 156
 - intersection of sets, 156
 - inverting items in a list, 125
 - is comparison operator, 166
 - is not comparison operator, 166
 - isalnum(), 97
 - isalpha(), 97
 - isdigit(), 97
 - islower(), 97
 - is_multipart(), 353
 - isspace(), 97
 - istitle(), 97
 - isupper(), 97
 - items(), 144–146
 - __iter__(), 233
 - iterables, 124, 171, 259
 - iterators
 - creating, 258, 260–261
 - defined, 258–259
 - enumerate(), 135
 - iteritems(), 180–181
 - itertools library, 258–260
- **I** •
- Java, 10
 - join(), 97
 - joining
 - paths, 285
 - strings, 93, 97
 - URLs, 342
 - Jython Wiki, 378
- **K** •
- key parameter, 371
 - KeyboardInterrupt exception, 266
 - keys(), 144–146
 - keyword arguments. *See* arguments
 - Kodos Debugger, 312
 - **kwargs parameter, 150, 190, 194
- **L** •
- lambda expressions, 268–269
 - largest item, finding, 279
 - LBYL (Look Before You Leap) method of
 - exception handling, 254
 - len(), 21, 124, 173
 - less than (<) operator, 48
 - lib/pythonXX/site-packages/ directory, 209

- libraries
 - Berkeley DB database library, 333
 - itertools library, 258–260
 - SQLite library, 326–330
 - standard library, 273–274
 - Zip files, 324–325
 - lib/site-python/ directory, 208
 - Linux operating system, 388
 - Linux Weekly News*, 10
 - list. *See also* sequences
 - adding items to the end of a list, 125
 - building, 134–135
 - copying, 131–132
 - counting occurrence of values in, 125
 - creating, 124, 134, 262–263
 - deep copying, 131–132
 - defined, 43, 67, 120
 - deleting items from, 125
 - differences from tuples, 121
 - disappearing, 132
 - elements, 124
 - filters, 263
 - indexing, 126–130
 - interactive mode, 21–22
 - inverting items in a list, 125
 - iterators, 258
 - methods, 124–125
 - modifying, 125
 - mutability, 120
 - operators, 123–124
 - performance problems, 133
 - queues, 135–136
 - range(), 173
 - repeating, 132–133
 - reversing, 125–126
 - shallow copying, 131
 - simultaneous test/add/delete, 131
 - slicing, 126–129
 - sorting, 120, 125–126
 - stacks, 135
 - sys.argv, 282
 - uniquely ordered list, 367–368
 - unpacking, 195
 - list(), 124, 134
 - listcomps (list comprehensions), 134, 261–263
 - listing
 - contents of a module, 204
 - files, 365
 - modules, 32–33, 205–206
 - literals, 41, 89
 - ljust(), 97
 - load(), 333
 - local namespaces, 197–199
 - local user groups, 380
 - locale, 107
 - log(), 299
 - loggers, 296–298
 - logging module, 83–84, 296–299
 - logging system, 251, 298–299
 - logical operators, 47–49, 162, 164–165
 - long integers, 42, 111–112
 - long strings, 93
 - Look Before You Leap (LBYL) method of exception handling, 254
 - loops
 - continuous loops, 178
 - defined, 54, 170
 - dictionaries, 145–147, 150–151
 - else clause, 177–178
 - enumerate(), 179–180
 - for loop, 55, 170–173, 175
 - generators, 260–261
 - infinite loops, 76
 - spider.py program, 66–67
 - stopping, 177
 - strings, 95
 - while loop, 55, 170, 173–174, 176
 - lower(), 97
 - lstrip(), 97
- M •
- Mac OS X operating system, 386–387
 - macostools module, 319
 - mailing lists, 377, 380
 - maintenance of programs, 76–79
 - map(), 268–270
 - mapping key, 104
 - match(), 310
 - matching regular expressions, 304–305, 310, 312–313
 - math module, 115
 - max(), 279
 - measuring strings, 21
 - membership testing with sets, 154
 - memoizing, 267–268
 - memory files, 313
 - message logging, 299
 - message_from_file(), 349
 - message_from_string(), 349
 - __metaclass__, 237, 246
 - metaclasses, 245–246
 - Method Resolution Order (MRO), 244–245
 - methods. *See also* def statement; functions
 - arguments, 46
 - bare methods, 66
 - calling, 45–46, 222–223

methods *(continued)*

- class methods, 58, 241–243
- classes, 58
- creating, 220–221
- data types, 45–46
- defined, 185
- differences from functions, 66
- extending in parent classes, 228–229
- hooks, 232
- initializers, 219
- lists, 124–125
- passing information to a method, 46
- regular expressions, 308
- special methods, 232–233
- `spider.py` program, 66
- static methods, 243
- strings, 68, 96–97
- unbound methods, 223
- underscore characters (`_`) in names, 219

MIME objects, 349–352

`min()`, 279

misspelled names, 75

modifying lists, 125

modules

- accessing code, 202
- attributes, 31, 203
- benefits, 10, 202
- Cheese Shop, 377–378
- code, 202
- conventions, 205, 226
- creating, 205
- debugging, 202
- defined, 12, 29, 57, 201
- documentation, 207
- examining, 24–25
- extension module, 201
- help, 33, 207
- importing, 30–32, 57, 202–203, 274
- initializing, 202–203
- interactive mode, 32–33
- listing, 32–33, 205–206
- listing contents, 204
- namespaces, 57, 200
- naming, 204–206
- `.py` suffix, 29, 201
- reloading, 280–281
- rules, 205
- testing, 295–296
- writing, 205

modulo (%) remainder operator, 50

`move()`, 320–321

moving

- directories, 321
- files, 320

- `_mro_` attribute, 245
- MRO (Method Resolution Order), 244–245
- multiple inheritance, 229–230
- multiple-line strings, 92–93
- multiplication operator (*), 47

• N •

name mangling, 226

`NameError` message, 253

`namelist()`, 323

names

- binding to objects, 40
- case-sensitivity, 40
- conventions, 80–81
- defined, 39
- keywords, 40
- naming, 80–81
- reserved words, 40
- rules, 40, 80–81
- `spider.py` program, 67–68
- values, 39–40

namespaces

- child classes, 230–231
- classes, 58, 224, 230–231
- code blocks, 56–58
- functions, 57, 196–200
- global, 197–200
- instances, 58, 224
- interactive mode, 24–25
- local, 197–199
- modules, 57, 200
- nested scopes, 198
- subclasses, 230–231

naming

- functions, 186–187
- loggers, 298
- modules, 204–206
- names, 81
- packages, 208

negative indexing, 100

`_neq_()`, 232

nested conditional expressions, 266–267

nested control structures, 161

nested scopes, 198

`_new_()`, 242–243

new features, 274

newsgroups, 76, 377, 379–380

new-style classes

- attribute management, 240–241
- class methods, 241–243
- controversy about, 236
- defined, 235–236

- documentation, 236
- exceptions, 265
- `__getattr__()`, 238
- inheritance, 237–238
- instances, 239
- metaclasses, 245–246
- MRO (Method Resolution Order), 244–245
- `__new__()`, 242–243
- properties, 240–241
- rules, 237
- `__slots__`, 244
- static methods, 243
- `super()`, 239–240
- tutorial, 236
- not equal to (`!=` or `<>`) operator, 48
- not in comparison operator, 166
- not operator, 49, 164
- numbers
 - automatic conversion of, 113
 - complex numbers, 42, 114
 - decimal numbers, 42
 - defined, 42–43
 - floating point numbers, 42, 112–114
 - integers, 42, 111–113

● ○ ●

- `object()`, 261
- object composition, 228
- objects
 - attributes, 214, 275, 278–279
 - binding names to objects, 40
 - container objects, 119
 - converting to strings, 233
 - defined, 39, 214–215
 - evaluation of, 20
 - finding an object's type, 280
 - iterables, 124
 - MIME objects, 349–352
 - new-style classes, 235–246, 265
 - persistent objects, 330–331
 - polymorphism, 215–216
 - regular expression object, 307–308
 - seeing information about, 19
 - singleton objects, 372–373
- old-style classes, 235
- online resources, 375–380
- `open()`, 275–276, 325
- operating systems, 273, 282–283, 383–388
- operator interception, 232–233
- operators
 - arithmetic, 47
 - comparison, 47–48, 93, 95, 165–166

- conditional, 47, 49
- defined, 47
- dictionaries, 195
- lists, 123–124, 195
- logical (Boolean), 47–49, 162, 164–165
- order of precedence, 50
- overloading, 232–233
- sequences, 123–124
- strings, 93–95
- tuples, 123–124, 195
- or operator, 49, 164–165
- organizing code, 201
- os module, 282–286
- `os.environ` dictionary, 286
- `OSError` message, 320
- overflow errors, 114
- overloading operators, 232–233
- overriding parent classes, 215, 227–228

● p ●

- packages
 - defined, 201, 207
 - features, 207
 - help, 209
 - importing items, 209–212
 - `lib/pythonXX/site-packages/` directory, 209
 - `lib/site-python/` directory, 208
 - naming, 208
 - requirements, 208
 - structure, 209
 - Tcl/TkAqua, 387
 - Tkinter, 312, 387
- packing tuples, 137
- pair programming, 78
- parameter attributes, 220
- parameters. *See also* arguments
 - default-value, 66, 190–194
 - defined, 184
 - positional, 190
 - specifying, 188–190
- parent classes. *See* base classes
- parentheses ()
 - grouping operator, 47
 - regular expression groups, 304, 306–307
- `parseaddr()`, 354
- `parsedate()`, 355
- Parser module, 353–354
- `partition()`, 97
- `pass` keyword, 178–179
- passing
 - arguments, 189–190
 - information to a method, 46

- Path Browser, 36
- paths, 284–286
- pdb module, 84–86
- PendingDeprecationWarning exception, 266
- PEP (Python Enhancement Proposal), 376
- performance problems, 133
- permissions for files/directories, 284, 320
- persistent databases, 330–331
- pickle module, 330–333
- pickling data, 330–333
- PIG (Python Interest Group), 380
- plus (+) addition operator, 47
- polymorphism, 215–216
- pop(), 130, 135–136, 144–145, 367
- Popen(), 286–287
- popitem(), 144–145
- populating database tables, 328–329
- positional parameters, 190
- pound sign (#), 51, 84
- precedence of operators, 50
- print statement, 19, 83
- printdir(), 323
- printing
 - exception details, 251
 - strings, 21, 90, 93
- printme(), 52
- private attributes, 226
- process_page(), 64–65
- programs
 - debugging, 71, 74–76, 82–86
 - design principles, 74–75
 - designing, 71–74
 - docstrings, 69, 81
 - documentation, 71–72
 - error checking, 69
 - initializing, 62–63
 - maintenance, 76–79
 - pseudo-code, 73
 - running, 63
 - spider.py, 59–70
 - structure of, 62
 - testing instructions, 62
 - writing, 12–14
- prompt, 17
- properties
 - extending in parent classes, 215
 - new-style classes, 240–241
- protocol parameter, 332
- prototyping, 9
- pseudo-code, 73
- PUG (Python User Group), 380
- .py suffix, 29–30, 201
- PyChecker tool, 95
- pysqlite module, 326
- Python community
 - controversies within, 236
 - news about, 379
 - philosophy of, 12
- Python Cookbook Web site, 365, 379
- Python Database API Specification v2.0, 330
- Python Enhancement Proposal (PEP), 376
- Python Interest Group (PIG), 380
- Python interpreter. *See* interactive mode
- Python Library Reference, 207
- Python User Group (PUG), 380
- Python Wiki, 378, 380
- PYTHONDOCS, 28, 384
- Python.org Web site, 375–376
- PYTHONPATH, 205–206

• Q •

- quantize(), 118
- queues, 135–136
- quitting interactive mode, 26–27
- quotation marks in strings, 89–91
- quote(), 354
- quote_plus(), 339–340

• R •

- raise statement, 254–255
- raising exceptions, 143, 247, 254–255
- random module, 116, 366–367
- random numbers, 116
- random.choice(), 366–367
- range(), 172–173
- ratio(), 370
- raw strings, 92. *See also* strings
- raw_input(), 279–280
- re module, 301–302, 307–309
- read(), 277, 322
- readability of regular expressions, 313
- reading files, 277
- readinto(), 325
- readlines(), 277
- real numbers, 42, 114–116
- recursive directories, 284
- reduce(), 268, 270
- refactoring the code, 77–79
- references to functions, 198
- regexes. *See* regular expressions
- regression testing, 293
- regular expression object, 307–308

- regular expressions
 - compiling, 307, 309
 - creating, 302
 - debugging, 312
 - escape codes, 304–305
 - finding in a string, 309
 - finding repeats, 305–306
 - flags, 309
 - functions, 308
 - grouping, 304, 306–307
 - matching, 304–305, 310, 312–313
 - methods, 308
 - readability, 313
 - rules, 303
 - special characters, 303–304
 - splitting, 310–311
 - text searches, 301–303
 - tutorial, 312
- regular integers, 42, 111–112
- relative importing, 211–212
- relative pathnames, 285
- reload(), 280–281
- reloading modules, 280–281
- remainder (modulo %) operator, 50
- remove(), 125
- removing. *See* deleting
- repeating (*) operator, 93–94
- repeats in regular expressions, 305–306
- replace(), 97
- repr(), 20, 82–83
- __repr__(), 233
- requirements for packages, 208
- re-raising exceptions, 255
- reserved words, 40
- resources, 375–380
- resumable function. *See* generator
- return codes, 287
- return statement, 188
- reverse(), 125
- reversed(), 126, 181
- reversing lists, 125–126
- rfind(), 97
- rindex(), 97
- rjust(), 97, 105
- rmtree(), 321
- roll(), 366
- root logger, 296–297
- round(), 23–24, 113
- rpartition(), 97
- rstring(), 97
- rstrip(), 277
- running
 - programs, 63
 - scripts, 29–30
- S •
 - saving
 - exceptions, 255
 - files, 277
 - SAX (Simple API for XML), 348
 - scripts
 - CGI scripts, 356–361
 - defined, 8, 29
 - .py suffix, 29–30
 - running, 29–30
 - writing, 30
 - search(), 310
 - searches
 - depth-first, left-to-right, 230–231
 - glob, 365
 - regular expressions, 301–303, 306
 - strings, 97
 - self parameter, 219, 221, 223, 243
 - send(), 261
 - sendmail(), 355–356
 - sequences. *See also* list; strings; tuples
 - comparing, 122–123
 - converting into a dictionary, 149–150
 - elements, 43
 - operators, 123–124
 - uses, 42
 - sequential data types, 42–44, 119
 - setattr(), 278–279
 - __setattr__(), 233
 - set_charset(), 351
 - setdefault(), 148–149, 153
 - __setitem__(), 233
 - set_param(), 352
 - set_payload(), 352
 - sets, 42, 44–45, 67, 154–157
 - setting attributes, 278–279
 - shallow copying, 131, 147–148
 - shelve module, 334–336
 - shuffle(), 366–367
 - shutdown(), 299
 - shutil module, 319–320
 - shutting down logging system, 299
 - Simple API for XML (SAX), 348
 - Simple Mail Transfer Protocol (SMTP), 355–356
 - simultaneous test/add/delete in lists, 131
 - singleton objects, 372–373
 - sleep(), 289, 292–293
 - slicing
 - list, 126–129
 - strings, 98–100
 - tuples, 126
 - __slots__(), 244

- smallest item, finding, 279
- smtplib module, 349, 355–356
- software architects, 73
- sort(), 96–97, 125, 132, 142, 182, 371
- sorted(), 126, 181–182, 371
- sorting
 - dictionary keys, 141–142
 - DSU (decorate, sort, undecorate) sorts, 370–371
 - list, 120, 125–126
 - strings, 96–97
- span(), 312
- sparse data, 139
- special characters
 - in regular expressions, 303–304
 - in URLs, 339–340
- special methods, 232–233
- special package directories, 208
- specifying parameters, 188–190
- spider.py program
 - class statements, 63
 - code listing, 60–61
 - data types, 67
 - def statements, 63
 - error handling, 69–70
 - find_links(), 64–65
 - functionality, 59
 - functions, 66
 - import statements, 62
 - __init__(), 64, 66
 - initializing, 62–63
 - loops, 66–67
 - methods, 66
 - missing features, 69
 - names, 67–68
 - process_page(), 64–65
 - running, 63
 - Spider class, 63–64
 - strings, 68–69
 - structure of, 62
 - url_in_site(), 65
- split(), 21–22, 97, 310
- split extension, 286
- splitlines(), 97
- splitting
 - pathnames, 286
 - paths, 285–286
 - regular expressions, 310–311
 - strings, 21–22, 97
 - URLs, 341–342
- SQLite library, 326–330
- sqlite3 module, 326–330
- stacks, 135
- standard library, 273–274
- StandardError message, 369
- starting
 - IDLE, 34, 385
 - interactive mode, 17
- startswith(), 68, 96–97
- statements. *See also* expressions
 - defined, 21, 41
 - ; (statement separator), 41
- static methods in new-style classes, 243
- stopping loops, 177
- str(), 20, 96, 113
- __str__(), 233
- strftime(), 290
- string buffers, 313
- string exceptions, 266
- string module, 97
- StringIO module, 313–314
- StringIO object, 314–315
- strings. *See also* sequences
 - case manipulation, 97
 - combining, 93, 97
 - comparing, 95
 - concatenating, 94
 - converting data to string, 96
 - converting objects to string, 233
 - converting Unicode to string, 108
 - counting, 97
 - cStringIO module, 313–314
 - decoding into Unicode, 108–109
 - defined, 43, 89
 - deleting indentation, 315
 - docstrings, 52, 69, 81, 90–91
 - elements, 98–100
 - escape character (\), 91–92, 304
 - escape codes, 91–92
 - evaluating, 281
 - finding regular expressions, 309
 - formatting, 97, 101–105
 - help, 97
 - immutability, 101
 - indexing, 98–100
 - interactive mode, 21–22
 - internal representation of, 93
 - joining, 93, 97
 - literals, 89
 - long strings, 93
 - loops, 95
 - measuring, 21
 - memory files, 313
 - methods, 68, 96–97
 - multiple-line strings, 92–93
 - operators, 93–95
 - printing, 21, 90, 93
 - quotation marks, 89–91
 - raw strings, 92

- repeating, 93–94
- search and replace, 97
- slicing, 98–100
- sorting, 96–97
- `spider.py` program, 68–69
- splitting, 21–22, 97
- string module, 97
- `StringIO` module, 313–314
- testing contents of, 96–97
- text encoding, 97
- Unicode strings, 97, 105–109
- unpickling, 333
- uses, 42
- `strip()`, 97
- `sub()`, 311
- subclasses. *See also* classes
 - creating, 227
 - extending a parent class, 215, 228–229
 - inheritance, 215, 228
 - multiple inheritance, 229–230
 - namespaces, 230–231
 - overriding parent class, 215, 227–228
- subclassing, 65
- submitting form data, 340–341
- `subprocess` module, 282–283, 287–288
- subsets, 155
- suite. *See* code blocks
- `sum()`, 270
- `super()`, 239–240
- superclasses. *See* base classes
- supersets, 155
- `swapcase()`, 97
- swapping values, 138
- symmetric difference (^) operator, 53, 156
- symmetric difference of sets, 156
- syntax errors, 53, 75
- `SyntaxError` message, 248
- `sys` module, 281–282
- `sys.argv` list, 282
- `sys.exit()`, 282
- `SystemExit` exception, 266

● T ●

- tables (databases), 328–330
- Tcl/TkAqua package, 10, 387
- `tempfile` module, 284
- temporary files, 284
- test-driven development, 78
- testing
 - contents of strings, 96–97
 - dictionaries, 146
 - `doctest` module, 293–296
 - modules, 295–296

- regression testing, 293
 - `unittest` module, 293
- testing instructions, 62
- `testzip()`, 323
- text
 - formatting, 315–318
 - regular expression text searches, 301–303
 - wrapping, 315–318
- text editor, 36
- text encoding, 97
- `textwrap` module, 315–318
- `TextWrapper` object, 317–318
- `throw()`, 261
- `time` module, 289, 291–292
- `time()` object, 289, 292
- `timedelta()` object, 290–291
- `title()`, 97
- Tkinter package, 312, 387
- tracebacks, 53, 83, 249, 361
- `translate()`, 97
- triple quote (""") or (`` `), 90
- true division, 112
- `truncate()`, 325
- `try` statement, 56, 248–253
- `try/except` block, 248–252
- `try/finally` block, 248–249, 252–253
- `tuple()`, 136–137
- tuples. *See also* sequences
 - comma (,) in tuple creation, 122
 - converting into a dictionary, 149
 - converting literal objects into, 136–137
 - defined, 43, 120–122
 - empty tuple (), 121–122, 163
 - examples, 121
 - immutability, 120
 - indexing, 126
 - operators, 123–124
 - packing, 137
 - slicing, 126
 - swapping values, 138
 - tuple literal, 121
 - unpacking, 137–138, 195
- Tutor mailing list, 377
- type. *See* classes; new-style classes
- `type()`, 82–83, 217, 280
- type/class unification, 236
- type-testing, 368–369

● U ●

- unbound methods, 223
- underscore characters (_,), 20–21, 40
- Unicode strings, 97, 105–109. *See also* strings
- union (|) operator, 156

`unique()`, 367–368
 uniquely ordered list, 367–368
`unittest` module, 293
 Universal Naming Convention (UNC) paths, 284–285
 Unix operating system, 388
 unpacking

- arguments, 195–196
- tuples, 137–138

 unpickling data, 333
`unquote()`, 354
 updating dictionaries, 151
`upper()`, 45–46, 97
`urlencode()`, 341
`url_in_site()`, 65
`urljoin()`, 68
`urllib` module, 337, 340
`urllib2` module, 337, 340
`urlopen()`, 337–338, 340
`urlparse()`, 341–342
`urlparse` module, 341–342
 URLs, 337–342
 Usenet newsgroups, 76, 377, 379–380
 user groups, 380
 user input, 248, 279–280

• U •

`ValueError` message, 125, 254
 values

- class attributes, 224–225
- counting occurrence of in a list, 125
- data types, 41
- dictionaries, 143–144, 151–152
- environment variables, 286
- errors, 76
- expressions, 20–21
- functions, 188
- instance attributes, 224–225
- names, 39–40
- swapping, 138

`values()`, 144–146
 variables, 39
 versions of Python, 391–394
 viewing attributes, 278

• W •

Web form data. *See* form data
 Web page errors, 338–339
 Web site resources, 375–380
`while` loop, 55, 170, 173–174, 176–178
 wikis, 378, 380
 Windows operating system, 383–385
 Windows UNC paths, 284–285
 WinZip, 321
`with` statement, 56, 264–265
 word comparisons, 370
`wrap()`, 315–316
 wrapping text, 315–318
`write()`, 277–278, 323
`writeln()`, 324
`writelnstr()`, 324
 writing. *See also* creating

- modules, 205
- programs, 12–14, 25–26
- scripts, 30

 writing to files, 277–278
 WxPython installer, 312

• X •

XHTML files, 342–344
 XML, 344–348
`xml.dom` module, 348
`xml.dom.minidom` module, 348
`xml.sax` module, 348
 XP (Extreme PFmming), 78
`xrange()`, 173

• Y •

`yield` statement, 260. *See also* generator

• Z •

`zfill()`, 97
`zip()`, 134, 137, 180, 268, 270
 Zip files, 321–326
`zipfile` module, 321–325