

# Index

## SYMBOLS/NUMERICS

**!** = (exclamation and equals), unequal comparison, 53

**'** (single quotes), 9–11

**“** (double quotes), 9–11

**“ “ “** (triple quotes), 7, 10

**#** character, 78

**%** sign (format specifiers)

- as remainder operator, 370, 375
- as string formatting operator, 21, 370
- strings, 12

**%d** conversions, 371

**%f** format specifier

- floating-point numbers, 370–371
- program files, 24

**%o** and  **%#o** conversion, 371

**%w.pf** conversion, 371

**%x** conversion, 371

**( )** (parentheses), types, 34

**\*** (asterisk)

- floating-point conversions, 372
- for glob patterns, 141
- importing from modules, 167
- modules' contents, 120
- multiplication, 21
- in queries, 248

**\*\*** (exponentiation operator), 375

**+** (plus sign)

- to combine strings, 11
- number types, 20

**,** (commas)

- recursive functions, 133
- in tuples, 36–37

**.** (periods)

- creating modules, 113, 114
- as general wildcard, 199

**\** (backslash)

- directory names, 131
- regular expressions, 199

- special text, 192
- in strings, 127–128

**/** (forward slash), 21, 131, 192, 272

**//** (forward slashes), floor division, 375

**=** (equals sign), names and values, 32

**==** (double equals), equality comparison, 52

**?** (question mark), for glob patterns, 141

**{ }** (curly braces), types, 34

**[ ]** (square braces)

- glob patterns, 141
- lists, 37
- types, 34

**2to3** tool, 558

## A

**\_\_all\_\_** list, 120–121

**\_\_all\_\_** variable, 167

**abs** built-in function, 375

**absolute paths**, defined, 129, 134

**ADA programming language**, 290

**add\_some\_text()** function, 129

**addition, testing**, 210–212

**addresses (Internet)**, 292

**administrative panel (web applications)**, 388

**aliases in SQL queries**, 249

**Amazon.com Web service**

- responses, 444–445
- REST quick start, 443–445

**anonymous functions**, 143–144

**Apilevel** global, 261

**APIs (Application Programming Interfaces)**, 252–262

- AWS, 443
- basics of, 239
- swing APIs, Jython, 492–493

**application layer**, 291

**applications. See also Web applications**

- Django, creating, 403–405
- Jython-based, packaging, 488–489
- vs. projects (Django), 403

## architecture

- client-server, 333, 409
- Django, 390–396
- MVC, 390–391
- peer-to-peer, 333–334
- of Web, 408–409

## args data list, 150

## arguments

- defined, 26, 116
- format specifiers, 26
- Jython, 486

## argv (argument vector), 116

## arithmetic

- order of evaluation, 24–25
- program files, 21–23
- in Python, 374–375
- testing, 210–212

## ArithTest class, 210, 211, 215

## ArithTest2 class, 216

## ArithTestSuper class, 215

## array module, 382–383

## arrays, 380–383

## assert language feature, 208–209

## AssertionErrors, 208

## assertions, 208–209

## assignment operator (the = sign), 83

## attachments (MIME), 298

## authentication (users), 388

## axis (XPath), 272

# B

## base 10, defined, 27

## base 16, defined, 27

## base 8, defined, 27

## Base64 encoding, 295–297

## BaseRequestHandler subclass, 320

## baz function, 343, 344

## Beginning XML, 3rd Edition (Wrox Press), 267

## binding to external hostnames, 316–317

## BitTorrent, 334

## BittyWiki

- API documents, 473
- API through XML-RPC, 460–463
- core library, 429–432
- exposing SOAP interface to, 468–470
- manipulating through SOAP, 470
- manipulating through WSDL proxies, 477–478
- REST API, 448–451, 473

## BittyWiki Web interface, 432–441

- markup, 435–441
- request structure, 433
- resources, 433–435

## bittywiki.delete(string pageName) method, 473

## bittywiki.getPage(string pageName)

### method, 473

## BittyWikiRestAPI class, 453–454

## bittywiki.save(string pageName, string text)

### method, 473

## blocks, programming in, 4

## borders of widgets, customizing, 234

## boundaries (e-mail), defined, 300

## braces, enclosing, 34

## break statements, 63, 64

## bug reports, 224–225

## built-ins

- docstrings (documentation strings), 375–378
- math functions, 375–378
- Python 3.1 changes in, 557

# C

## c

- Java, 481
- Python, 337, 481
- Python interpreter, 290–291

## C, extension programming with, 337–366

- C vs. Python, 337
- exercises, 366
- extension modules, building and installing, 340–342
- extension modules outline, 338–340
- LAME extension module, 350–363
- LAME project, 346–350
- parameters, passing to C, 342–345
- Python objects from C code, 363–365
- returning values from C, 345–346
- summary, 366

## C++

- characters as numbers, 373
- Java, 481

## caching, web frameworks, 389

## calling functions, 88

## case sensitivity

- globbing, 140
- SQL keywords, 248

## CGI (Common Gateway Interface), 417–422

- basics of, 417–418
- environment variables, 420–422

- scripts, running, 418–419
- user input with HTML forms, 422
- web interface to BittyWiki, 435–436
- Web servers and scripts, 419–420
- CGIXMLRPCRequestHandler function, 474**
- channels (chats and), 323**
- characters as numbers, 373–374**
- Chat Server. See Python Chat Server**
- checkboxes, creating, 235**
- child processes, 152**
- children, defined, 163**
- children classes**
  - creating, 281–282
  - defined, 163
- class keyword, 96**
- classes, 95–107. See also specific classes**
  - children classes, creating, 281–282
  - code, making into objects, 96–103
  - defined, 111, 163
  - defining/creating, 96–107, 163–164
  - documenting, 168–169
  - element classes (XML), 281–283
  - exceptions, 97
  - extending, 165–166
  - interface methods, writing, 100–101
  - internal methods, writing, 99
  - Java classes, using in Jython, 489–494
  - JNDI, 507
  - mail, 300
  - objects, creating from, 96–99
  - objects, 93–94, 104–107
  - overview, 111
  - Python 3.1 changes in, 555
  - scope of objects, 104–107
  - servlet classes in Jython, creating, 503
  - SmartMessage and MailServer classes, 302–305
  - widget classes, 237
- clients**
  - chat clients, 329–331
  - mirror clients, 318–320
  - Web clients, interacting with, 408
  - WikiSpiderSOAP.py client, 470–473
- client-server architecture, 333, 409**
- clipping logs, 191**
- close method (dbm modules), 242, 244**
- cmath module, 380**
- code. See also source code**
  - creating modules from pre-existing, 113–115
  - defining, def, 73
  - grouping under names, 73–74
  - making into objects, 96–103
  - saving in files, 71–72
- Code Editor, saving program files with, 71**
- color**
  - background, setting (Java), 493
  - customizing (widgets), 234
- command line, starting modules from, 115–117**
- commands. See also specific commands**
  - four basic (HTTP), 411
  - Jython, executable, 186–187
  - servers, 325
- commas (,)**
  - recursive functions, 133
  - in tuples, 36–37
- comments**
  - basics of, 78–79
  - web applications, 388
- Common Gateway Interface (CGI). See CGI (Common Gateway Interface)**
- comparison of values**
  - difference comparison, 53
  - equality comparison, 51–53
  - more than one comparison, 56–60
  - Python 3.1 changes in, 555
- compiling, .pyc Files, 122**
- complex joins, writing, 257–258**
- complex numbers, 16–17, 378–380**
- concatenation, 11–13**
- configuring**
  - database settings (Django models), 401–403
  - GUI widgets, 231, 234
- conjugate method, 379**
- connections (databases)**
  - Connection object, 253
  - transactions, 260
- content types (MIME), 297**
- context, defined, 6**
- continue statement, 64–65**
- controller, MVC architecture, 391**
- copying**
  - data, 33
  - files, 138
- C-Python**
  - basics of, 483
  - vs. Jython, 483
  - Jython, handling differences, 510–511
- CRUD (Create, Read, Update, Delete), 247, 248, 433**

## **Cunningham, Ward, 428**

### **curly braces ( { } ), types, 34**

#### **cursors**

- databases, 253–255
- defined, 253
- dynamic, 500
- static database cursors, 500
- widgets, customizing, 234

### **customizing widgets, 233–234**

## **D**

### **\_\_doc\_\_, 76**

#### **data**

- changing through names, 33
- copying, 33
- dictionaries as indexed groupings of, 39–41
- lists as changeable sequences of, 37–39
- names for, 31–34
- representation of in XML-RPC, 457–458
- storage, relational databases, 245
- storing using lists, 45–46
- tuples for unchanging sequences of, 34–37

### **data link layer, 291**

#### **database APIs**

- complex joins, writing, 257–258
- connections, creating, 253
- cursors, 253–255
- documenting. *See* Web service APIs, documenting
- employees, removing, 259–260
- errors, handling, 261–262
- managers, updating, 258–259
- module capabilities, 261
- modules, downloading, 252–253
- simple query, writing, 256–257
- transactions, 260–261

### **databases. *See also* DBM persistent dictionaries; relational databases; text processing**

- accessing, 239–240
- basics of, 189
- connectivity (web applications), 388
- exercises, 263
- setting up, 250–251, 496–500
- settings for Django models, 401–403
- summary, 262–263

### **databases, accessing from Jython, 494–500**

- basics of, 494–495
- databases, setting up, 496–500

- Python DB API, 495
- tables, creating, 497–500

#### **DB API**

- basics of, 252
- modules, downloading, 252–253

### **dbm module, 240, 245**

### **DBM persistent dictionaries, 240–245**

- accessing, 243–244
- creating, 241–242
- DBM modules, choosing, 240–241
- vs. relational databases, 245

### **dbm.dumb module, 240**

### **dbm.gnu module, 240**

### **decimal points in formatting numbers, 371, 372**

### **decisions in Python, 57–60**

### **def, defining code, 73**

### **defining/creating classes, 96–107, 163–164**

### **delete(string pageName) method, 473**

### **deleting. *See also* removing**

- CRUD, 247, 248, 433
- files, 138
- QUID, 247
- rows, 249

### **dereference feature, tuples, 35**

### **dialog boxes, creating, 236–237**

#### **dictionaries**

- defined, 39
- dictionary parameters, 222
- getting keys from, 40–41
- making, 39–40
- string substitution using, 148–149

### **difference comparison, 53**

#### **dir function**

- methods, objects and strings, 94–95
- modules, 158–159
- print\_dir function, 137

### **directories. *See also* files and directories**

- contents, 135–136
- creating and removing, 140
- navigation of, text processing, 190
- packages, 118–119
- recursive listings, 136–137
- types of entries, 136

#### **distutils package**

- distributing modules, 341
- for installing modules, 184, 185, 186

### **division, 21–22**

### **Django, 387–406**

- applications, creating, 403–405
- apps vs. projects, 403

architecture of, 390–396  
 exercises, 406  
 installing, 389–390  
 origination of, 389  
 project setup, 391–394  
 summary, 405–406  
 templates, 396–398  
 templates and views. *See* templates and views (Django)  
 URLconf, creating, 395–396  
 views, creating, 394–396  
 web application frameworks, 387–389

**docstrings (documentation strings), 75, 96**

**document models, 268**

**document root (XML), 266**

**documentation**  
 in context of functions, 75  
 of modules, 168–176  
 Web services, 441–442

**documenting APIs. *See* Web service APIs, documenting**

**doGet method, 503**

**DOM**  
 basics of, 275–276  
 parsers, 276–278

**doPost method, 503**

**double equals (==), equality comparison, 52**

**double quotes (“), 9–11**

**drivers**  
 databases, 496–497  
 JDBC, 494

**DTDs (Document Type Definitions)**  
 basics of, 268–270  
 HTML, 273

**dynamic cursors, defined, 500**

## E

**Eclipse Integrated Development Environment (IDE), 506**

**element classes (XML), 281–283**

**else: condition, 64**

**e-mail. *See also* mailboxes; MailServer**  
 mail spools, parsing with, 305–306  
 retrieving, 305–313  
     from IMAP servers, 309–313  
     parsing with mailbox, 305–306  
     from POP3 servers, 307–309  
     printing mailbox summaries, 309, 311

    security: POP3 and IMAP 313  
     webmail vs. e-mail, 313

**sending, 293–305**  
     e-mail file format, 294–295d  
     example of, 288–289  
     with MailServer, 305  
     MIME messages. *See* MIME (Multi-purpose Internet Mail Extension)  
     with SMTP and smtplib, 303–304  
     sifting through, 192  
     vs. webmail, 313

**employees**  
 managers of, updating, 258–259  
 removing, 259–260

**encapsulation**  
 basics of, 163  
 defined, 168

**encode function, 350–351**

**encodings (MIME), 295–297**

**equality**  
 equality comparison, 51–53  
 more/less than, 55

**equals (=) sign, names and values, 32**

**error codes, standard, 416**

**error handling**  
 database modules, 261–262  
 deeper, reading, 88–89  
 flagging errors, 87  
 module-specific, defining, 166–167  
 preparing Python for, 65–67  
 SOAP Web services, 468  
 XML-RPC Web services, 459–460

**etiquette for Web services, 479–480**

**events, SAX, 275**

**except: statements, 65**

**exceptions. *See also* os exceptions**  
 classes, 97  
 creating, 66–67  
 DB API, 261–262  
 error handling, 65–67  
 file exceptions, 131  
 format specifiers, 26  
 IOError exception, 131  
 Python 3.1 changes, 554

**execute method, 256, 258**

**exercises (answers by chapter), 515–548**

**exponential notation, 370**

**exponentiation operator (\*\*), 375**

**extend method, for growing lists, 45**

**extensibility, XML, 267**

**Extensible Markup Language. See XML (Extensible Markup Language)**

**eXtensible Stylesheet Language Transformations (XSLT), 280**

**extension modules, C and**

- building and installing, 340–342
- outline, 338–340

**Extreme Programming. See XP testing methodology**

## F

**fall-through statements, 60**

**false values. See values, True and False**

**Fielding, Roy, 408, 409**

**file systems, navigating with os module, 192–198**

- basics of, 192–194
- files, listing, 194–195
- files, searching for, 195–198
- paths, 194–195

**files**

- directories, 127–142
  - exceptions in os. *See os exceptions*
  - exercises, 142
  - file exceptions, 131
  - file objects, 127–131
  - paths and directories, 131–132
  - summary, 142
  - text, appending to files, 129
  - text files, reading, 130–131
  - text files, writing, 128–129
- file information, obtaining, 136–137
- file permissions, 138
- listing, 194–195
- parsing (XML), 284
- renaming/moving/copying/removing, 137–138
- rotating, 138–139
- searching for, 190–191, 195–198, 220–224
- WSDL, 475, 477

**filter functions, 143–144**

**find\_file function, 221, 222**

**finding files. See searching for files**

**flags**

- creating persistent dictionaries, 242
- defined, 116

**float objects, 369**

**float type, 368**

**floating-point numbers**

- %f format specifier, 24
- basics of, 16, 19

in Python, 369–370

using care with, 23

**floor division, 375**

**folders, saving program files in, 71**

**fonts (widgets), customizing, 234**

**for . . . in . . . : statements, 63, 64**

**for loop, iterating, 60, 61–62, 64**

**for operations, 60, 61–62**

**foreign keys, defined, 246**

**ForkingMixIn class (SocketServer module), 321**

**format specifiers, 12**

**formats**

- e-mail, 294–295
- of numbers, 25–26

**formatting numbers, 370–372**

**forms (HTML), limited vocabulary, 422–423**

**forward slash (/), 21, 131, 192, 272**

**fromstring() function, 283, 284**

**functions**

- anonymous, 143–144
- built-in math, 375–378
- C implementation of, 338–339
- calling, 88
- creating modules with, 162
- defined, 8
- defining, 74
- documenting, 168–169
- inside of, 86–87
- invoking when complete, 85–86
- invoking with parameters, 80
- lambda and filter, 143–144
- layers of, 88–89
- methods, 95
- named. *See named functions*
- os and os.path, 193–195
- in os.path, 137
- program files, 71–73
- recursive, 133

## G

**\_\_getitem\_\_ method, 147**

**getPage(string pageName) method, 473**

**global scope**

- importing into, 120
- names in, 77

**globals (DB API), 261**

**globbing, 140–141**

**glue (C), defined, 338**

**gnu\_getopt**, 150

**graphical user interface**. See **GUI (graphical user interface)**; **GUIs**, writing

**greater than and less than**, 54–55

**the Grinder**, 483

**GUI (graphical user interface)**

GUI widgets. See **Tkinter**, creating GUI widgets with

IDLE GUI, 5

writing, 227–238

  exercises, 238

  overview, 227

  summary, 238

**Tkinter**. See **Tkinter**, creating GUI widgets with toolkits for, 228–229

## H

**handle\_data** function (**HTML**), 274

**handle\_starttag/endtag** methods (**HTML**), 273–274

**handleRequest** method, 503–504

**handling errors**. See **error handling**

**has** and **has\_various** methods, 101

**headers**

  email, 294–295

**HTTP**, 416–417

**help** function, **documenting modules**, 169–176

**hexadecimal literals**, 368, 371

**hexadecimal numbers**, **formatting**, 27

**hierarchy**

  schemas, 271

  of widgets, 232

**Hoare, C. A. R.**, 337

**Holovaty, Adrian**, 389

**hostnames**

  external, binding to, 316–317

  vs. IP addresses, 292

**HSqIDB**, 496–497, 498–499

**HTML**

  basics of, 422

  forms, safety when accessing, 423–427

  forms basics, 422

  forms limited vocabulary, 422–423

  as subset of **XML**, 272–274

**HTMLParser** class, 273

**HTTP: real-world REST**, 411–417

**HTTP** requests, 414–416

**HTTP** responses, 414–417

  Visible Web Server, 412–415

  Web server, 411–412

**HTTP\_USER\_AGENT** string, 422

**HttpServlet** (**Jython**), 503–504

**Hunter, David**, 267

## I

**-i**, **running programs with**, 73, 486

**\_\_init\_\_** method, 98

**\_\_init\_\_** implementation, 355–356

**\_\_init\_\_.py** file, 119, 391

**IANA (Internet Assigned Numbers Authority)**, 293

**IDE (Eclipse Integrated Development Environment)**, 506

**IDLE GUI**, 5

**if**. . . reserved word, 57–60

**Im** mathematical operation, 379

**imaginary numbers**, 17–18, 378

**IMAP**

  e-mail from servers with **imaplib**, 309–313

  security, 313

**immutable frozensets**, 46

**import** keyword, 112–113, 115, 118

**importing**

  extension modules, 342–343

  modules, 112–113, 159

  modules and packages, 121–123

**IndexErrors**, 37

**infinite loops**, **defined**, 62–63

**inheritance**, 163

**initialization function**, 340

**installing**

**Django**, 389–390

**Jython**, 483–484

  modules, 183–186

**Python**, 5–6

**Tomcat**, 501

**int** constructor, 368

**int** type, 368

**integers**

  basics of, 16, 19, 368

  long integers, 369

**Python 3.1** changes, 554

**interfaces**

  basics of, 97–98

**BittyWiki** Web interface. See **BittyWiki** Web interface

**Python**, 363

robots, 441

user interfaces from Jython, 492–494

### **Internet. See also network programming**

addresses, 292

ports, 293

### **Internet Assigned Numbers**

**Authority (IANA), 293**

### **Internet Protocol (IP)**

basics of, 292–293

stack, 290–291

### **interpreted languages, defined, 4**

#### **interpreters**

Jython interpreter, embedding, 507–510

JythonInterpreter plugin, 506

Python interpreter and C, 290–291

### **IOError exception, 131**

### **IP addresses, 292**

#### **iteration**

iterators, generating for loops, 146–148

for loop, 60, 61–62, 64

returning iterators (Python 3.1), 553

while operations, 60–62

## **J**

### **Java, integrating with Jython, 489–506**

databases. *See* databases, accessing from Jython

Java classes, using in Jython, 489–494

Java EE servlets, writing in Jython. *See* Jython, writing Java EE servlets in

user interfaces from Jython, 492–494

### **Java, integrating with Python, 481–512**

C-Python and Jython, handling differences, 510–511

C-Python vs. Jython, 483

exercises, 512

Java, integrating with Jython. *See* databases, accessing from Jython; Java, integrating with Jython

Java basics, 481

Java vs. Python, 481

JDBC, 494

Jython, executable commands, 186–187

Jython, installing, 483–484

Jython, running interactively, 484–485

Jython, running on your own, 488

Jython, testing from, 506–507

Jython basics, 481–482

Jython interpreter, embedding, 507–510

Jython scripts, calling from Java, 508–510

Jython scripts, controlling, 486–487

Jython scripts, running, 485–486

Jython-based applications, packaging, 488–489

scripting with Java applications, 482–483

summary, 511

### **Java Database Connectivity (JDBC), 494**

### **Java EE (Java Platform Enterprise Edition), 500**

### **Java methods, calling, 510–511**

### **Java Naming and Directory Interface (JNDI)**

classes, 507

### **Java virtual machine (JVM), 481**

### **javax.servlet.http.HttpServlet class, 503**

### **JButton widget, 493**

### **JDBC (Java Database Connectivity), 494**

### **jEdit text editor, 506**

### **JFrame widget, 494**

### **JLabel widget, 493**

### **JNDI (Java Naming and Directory Interface)**

classes, 507

#### **joins**

complex, writing, 257–258

defined, 248

performing, 248–249

simple query for performing, 256–257

### **Josephson, Michael, 444**

### **JVM (Java virtual machine), 481**

### **JyScriptRunner.java file, 508–510**

#### **Jython**

applications, packaging, 488–489

basics, 481–482

vs. C-Python, 483

C-Python, handling differences, 510–511

databases. *See* databases, accessing from Jython

executable commands, 486–487

installing, 483–484

Java classes, using in, 489–494

Jython command, 486

Jython interpreter, embedding, 507–510

resources, 550

running interactively, 484–485

running on your own, 488

scripts, calling from Java, 508–510

scripts, controlling, 486–487

scripts, running, 485–486

tools for, 506

user interfaces, 492–494

when to use, 483

**Jython, writing Java EE servlets in, 500–505**

- application server setup, 501
- basics of, 500
- HttpServlet, extending, 503–504
- PyServlet, adding to application servers, 501–503
- Python servlets, writing, 504–505
- tools for Jython, 506

**JythonInterpreter plugin, 506****K****Kaplan-Moss, Jacob, 389****KeyError, 66****keys**

- dictionaries, 39, 40–41
- foreign keys, defined, 246
- primary keys, defined, 246

**keys method (persistent dictionaries), 243****keyword parameter mechanism, 222****L****\_\_len\_\_ methods, 95****lambda, 143–144****LAME**

- extension module, 350–363
- project, 346–350

**languages**

- comparing protocols of, 289–290
- Internet protocol stack, 290–291
- interpreted, defined, 4
- large systems programming languages, 482
- scripting, 482
- semantic markup languages, 266
- XML, 265–267
- XSLT, 280

**layers**

- defined, 291
- of functions, 88–89

**layouts (GUI widgets), creating, 232–233****less than and greater than, 54–55****libgmail project, 313****libraries**

- BittyWiki core library, 429–432
- dbm library, 241
- LAME library, 347
- libxslt C library, 280
- SOAP library for Python, 466

- socket library, 314
- wxPython library, 229
- XML document to describe, 266
- XML libraries, 274

**Linux**

- compiling extension modules on, 340–342
- installing Python on, 6
- os.path on, 132–133
- packages for, 347

**lists**

- appending sequences to, 45
- arrays, 381
- basics of, 37–39
- list comprehension, 145
- recursive directory listings, 136–137
- slicing, 44
- for storing data, 45–46
- treating strings like, 41–42
- vs. tuples, 38–39

**literal numbers, 368****LiveHTTPHeader extension, 415****local scopes, 77****localhost, defined, 292****logs, clipping, 191****long integers, 369****loops**

- generating iterators for, 146–148
- infinite loops, defined, 62–63
- for loop, 60, 61–62, 64
- while loops, 60–62

**lxml, 280, 283–284****M****\_\_models.py\_\_ file, 404****Mac, installing Python on, 6****mail. See e-mail****mail spools, parsing with, 305–306****mailboxes**

- parsing with, 305–306
- POP3 and IMAP, 309, 311
- summary of, printing, 306, 309, 311

**MailServer, 305****make\_text\_file() function, 128****manage.py — file, 391****map function, 144–145****markup**

- BittyWiki Web interface, 435–441
- wikis, 429

**math module, 374, 376–377**

**mathematics. See also arithmetic**

in Jython, 485

in Python, 374–378

**max built-in function, 376**

**maxOccurs, 271**

**membership in classes, defined, 163**

**memory, DOM, 275, 276**

**methods. See also specific methods**

BaseRequestHandler subclass, 320

BittyWiki API server, 473

BittyWiki SOAP server, 473

defining classes, 99–103

documenting, 168–169

functions, 95

interface, writing, 100–101

internal, writing, 99

Java, calling, 510–511

Python 3.1 changes in, 555

strings, 94

XML-RPC introspection API, 474

**MIME (Multi-purpose Internet Mail Extension), 295–303**

attachments, 298

basics of, 295

content types, 297

encodings, 295–297

multipart messages, 298–303

SmartMessage, 302–303

**min built-in function, 376**

**mirror clients, 318–320**

**mirror servers, 317–318**

**mistakes, numbers, 26–27**

**modal dialog boxes, 236**

**the mode, 242**

**models (Django)**

basics of, 401

configuring database settings, 401–403

creating, 403–405

installing, 404–405

**models, MVC architecture, 391**

**Model-Template-View (MTV) architecture, 401**

**Model-View-Controller (MVC) architecture, 390–391**

**modules**

array, 382–383

basics of, 111–112

CGI, 423

cmath, 380

creating from pre-existing code, 113–115

current scope, 120–121

defined, 157, 162

exercises, 125

exporting from packages, 121

extension modules, building and

installing, 340–342

extension modules outline, 338–340

import keyword, 112–113, 115, 118

importing, 112–113

for interacting with web clients and servers, 408

for Internet protocols, 288

math, 374, 376–377

packages, using, 120–124

packages basics, 118–120

quopri module, 296

re-importing modules and packages, 121–123

removed/renamed in Python 3.1, 556–557

select module, 331–332

summary, 124–125

sys.modules, examining, 122–123

testing, 124

urllib module, 444

using, command line, 115–117

**modules, building, 157–188**

basics of, 157–159

classes, creating, 163–164

classes, extending, 165–166

completing, 166

creating whole modules, 179–183

DB API and module capabilities, 261

DB API modules, downloading, 252–253

documenting, 168–176

exercises, 188

exploring, 160–161

exporting, 167–168

finding, 159–160

functions, 162

getopt module, 149–152

importing, 112–113, 159, 167

installing, 183–186

modules and packages, creating, 162

module-specific errors, defining, 166–167

OOP, defining, 163

os module. *See* file systems, navigating

with os module; os exceptions

re module. *See* regular expressions and

the re module

running as programs, 178

selecting (DBM), 240–241

summary, 187

testing, 176–177

**modulus operation, 22****money, displaying, 25****moving**

files, 137–138

os exceptions, 137–138

**MP3s**

creating (LAME), 347–350

encoding, 346, 350–361

**MTV (Model-Template-View) architecture, 401****multidimensional sequences, 36****multipart messages (MIME), 298–303****Multi-purpose Internet Mail Extension (MIME).****See MIME (Multi-purpose Internet Mail Extension)****multithreaded servers, 321–322****mutable sets, 46****MVC architecture, 390–391****MySQL resources, 550**

## N

**named functions, 73–87**

calling from within other functions, 84–86

comments, 78–79

defining, 73–74

describing, 75–76

duplicate names, 76–78

errors, flagging, 87

functions inside of, 86–87

name selection, 75

parameters, checking, 81–83

parameters, setting default values, 83–84

values, providing, 79–80

**named variables, 93****names**

assigning values to, 32

for data, 31–34

grouping code under, 73–74

nicknames, defined (chats), 323

reserved, 34

**/names command, 323****namespaces, 266–267****network layer, 291****network programming, 287–335**e-mail, sending. *See* e-mail, sending; MIME (Multi-purpose Internet Mail Extension)

exercises, 335

peer-to-peer architecture, 333–334

protocol design considerations, 333

protocols. *See* protocolsretrieving e-mail. *See* e-mail, retrievingsocket programming. *See* socket programming

summary, 334

terse protocols, 333

trusted servers, 333

**/nick [nickname] command, 323****nicknames, defined (chats), 323****node test (XPath), 272****nodes**

DOM, 278

XPath, 272

**none value, 42****nonmodal dialog boxes, 236****Notepad, creating program files in, 18–19****NULL characters, LAME extension module, 358****numbers, 15–29, 368–374**

characters as, 373–374

complex, 378–380

exercises, 29

floating-point. *See* floating-point numbers

formats, 25–26

formatting, 370–372

integers. *See* integers

kinds of, 15–18

literal, 368

long integers, 369

mistakes, 26–27

octal and hexadecimal, 27

order of evaluation, 24–25

program files. *See* program files and numbers summary, 28

type function, 16

**numerical analysis, defined, 367****numerical programming, 367–385.****See also numbers**

arithmetic, 374–375

arrays, 380–383

built-in math functions, 375–378

complex numbers, 378–380

exercises, 384–385

mathematics, 374–378

summary, 384

## O

**- O, 209****object-oriented programming (OOP)**

defined, 4

defining, 163

usefulness of, 106

## **object-relational databases (ORM), 245**

### **objects**

- basics of, 93–95
- from C code, 363–365
- creating from classes, 96–99
- defined, 111
- file objects, 127–131
- LAME extension module, 351, 352, 353
- making code into, 96–103
- methods and strings, 94
- prevalence in Python, 339
- using, 95

### **octal, defined, 27**

### **octal numbers, formatting, 27**

### **OnDemandAmazonList class, 447–448**

### **open function, 241, 242**

### **operators**

- arithmetic, 374
- in DTDs, 269
- Python 3.1 changes in, 555

### **ord function, 373**

### **ORM (object-relational databases), 245**

### **os and os.path functions, 193–195**

### **os exceptions, 132–141**

- directories, creating and removing, 140
- directory contents, 135–136
- file information, obtaining, 136–137
- files, renaming/moving/copying/removing, 137–138
- globbing, 140–141
- paths, 132–134
- rotating files, 138–139

### **os module, 112. See also file systems, navigating with os module**

### **os.fork calls, 152, 153, 154**

### **os.spawn family of functions, 153**

### **os.wait function, 153**

### **os.walk function, 194, 196, 221**

### **overloading, defined, 22**

## **P**

### **P2P forums, xxxiii–xxxiv**

### **packages**

- basics of, 111, 118–120
- creating installable, 184–186
- defined, 111
- exporting modules from, 121

### importing, 121–123

### Jython-based applications, 488–489

### modules, 120–124

### Python 3.1 changes in, 557

### testing, 124

### **packing order (widgets), 233**

### **parameters**

- arguments, 116
- checking type, 81–83
- database connections, 253
- defined, 26, 79
- invoking functions with, 80
- passing to C, 342–345
- PyObject\_CallMethod, 365
- setting default values, 83–84
- XML-RPC, 458

### **params list (SOAP), 468**

### **Paramstyle global, 261**

### **parent-child relationships**

- widget hierarchy, 232
- XML, 283

### **parents**

- element classes (XML), 281–282
- parent directories, 140
- parent paths, 132–133
- processes, 152–153
- widgets, 232

### **parsing**

- DOM parsers, 276–278
- HTML forms, 423
- HTMLParser class, 273
- with lxml, 283–284
- with mailbox, 305–306
- SAX parsers, 276, 279
- xml.dom.minidom parser, 277–278
- xml.sax parser, 276

### **PATH\_INFO variable, 422**

### **paths**

- directories, 131–132
- os exceptions, 132–134
- os.path module, 194–195

### **peer-to-peer architecture, 333–334**

### **permissions**

- administrative panel, 388
- file permissions, 138

### **Pilgrim, Mark, 444**

### **plugins (Jython), 506**

### **polymorphism, 163**

**POP3**

- e-mail, retrieving, 307–309
- vs. IMAP, 309
- security, 313
- UIDs, 313

**poplib, retrieving e-mail with, 307–309**

**pop-ups (print() function), 8–9**

**ports (Internet), 293**

**predicates, XPath, 272**

**primary keys, defined, 246**

**print() function**

- basics of, 8–9
- displaying numbers with, 24
- joining strings with, 12
- Python 3.1, 553

**print\_dir function, 137**

**print\_dir\_by\_ext function, 136**

**print\_line\_lengths function, 131**

**printing**

- HTML form submissions, 426–427
- lengths of lines, 131
- math, 23
- sys.argv, 117

**private methods, 98**

**processes**

- multiple tasks with one process, 154–155
- using more than one, 152

**program files**

- first line of, 72
- .py extension, 72
- saving code in, 71–72

**program files and numbers, 18–24**

- %f format specifier, 24
- basic math, 21–23
- creating in Notepad, 18–19
- different number types, 19–21
- printing math, 23

**programming, 3–14**

- basics of, 3–5
- exercises, 14
- extension. *See* C, extension programming with Extreme Programming. *See* XP testing methodology
- foundation of, 93
- installing Python, 5–6
- network. *See* network programming
- numerical. *See* numerical programming
- OOP, 4. *See also* object-oriented programming (OOP)
- the shell, 6

- socket. *See* socket programming

- strings. *See* strings

- summary of basics, 13–14

**programming languages. *See* languages**

**programs, running modules as, 178**

**projects, vs. applications (Django), 403**

**properties**

- databases, connecting to, 496
- JDBC drivers, 494
- passing named to constructors, 491

**protocols, 289–293**

- defined, 289
- design considerations, 333
- Internet addresses, 292
- Internet ports, 293
- Internet protocol stack, 290–291
- mirror servers, 324
- programming languages, 289–290
- Python Chat Server, 323–329
- terse protocols, 333

**Purcell, Steve, 211**

**.py extension, 18, 72, 113**

**Py\_BuildValue function, 346**

**PyAmazon, 444**

**PyArg\_ParseTuple function, 343–344, 345, 364**

**PyArg\_ParseTupleAndKeywords function, 344–345**

**.pyc Files, compiling, 122**

**PyDev plugin, 506**

**PyMethodDef structures, 339**

**PyObject\_CallMethod, 365**

**PyQT, 228**

**PyServlet class, 501–503, 505, 507**

**Python**

- vs. C, 337
- C-Python and Jython, handling differences, 510–511
- C-Python vs. Jython, 483
- installing, 5–6
- vs. Java, 481
- Jython, 482–483
- Python DB API, Jython, 495
- when to use, 191

**Python 3.1 changes, 553–558**

- 2to3 tool added, 558
- in APIs, 553
- in built-ins, 557
- in classes, 555
- in comparisons, operators and methods, 555
- exceptions, 554

- integers, 554
- modules removed/renamed, 556–557
- in packages, 557
- print function, 553
- syntactical, 555–556
- in Unicode and 8 bit strings, 554

### **Python Chat Client, 329–331**

### **Python Chat Server, 322–329**

- basics of, 322–323
- design of, 323
- protocol, 323–329

### **Python Code Editor, 71**

### **Python IDLE GUI, 5**

### **Python interpreter, C, 290–291**

### **PythonChatClient.py client, 329**

### **PyUnit module. See also testing**

- basics of, 207, 209
- resources, 550

### **Pywftk resources, 550**

## Q

**queries, database, 256–257**

**QUERY\_STRING variable, 422**

**QUID (Query, Update, Insert, Delete), 247**

**/quit [farewell message] command, 323**

**quopri module, 296**

**quoted-printable encoding, 295–297**

**quotes in strings, 6, 7–11**

## R

**radio buttons, creating, 235–236**

**radix 10, defined, 27**

**raise . . . command, 87**

**range function, 146–148**

**range iterators, 147**

**raw strings, defined, 199**

**Re mathematical operation, 379**

**re module. See regular expressions  
and the re module**

**read method, 130**

**reading text files, 130–131**

**readline method, 130**

**records, inserting, 254–255**

**recursive functions, 133**

**regular expressions, 199**

- text processing, 190
- vs. wildcards, 199

**regular expressions and the re module, 199–203**

- basics of, 199–202

- exercises, 204

- summary, 203

- tests, adding, 202–203

**Reinhardt, Django, 389**

**relational databases, 245–251**

- vs. DBM persistent dictionaries, 245

- setting up, 250–251

- SQL statements, writing, 247–249

- tables, defining, 249–250

- working with, 245–247

**relative paths, defined, 129, 134**

**reload function (modules), 176**

**remainder operator (%), 375**

**reminder operation, 22**

**removing. See also deleting**

- directories, 140

- employees (from databases), 259–260

- files, 137–138

**renaming**

- files, 137–138

- modules, 556–557

**render\_to\_response method, 398–399**

**repetition**

- repetitive tasks, 60–62

- stopping, 62–65

**representation of resources (REST), 410**

**request structure (BittyWiki Web interface), 433**

**REQUEST\_METHOD verb, 422**

**requests (HTTP)**

- basics of, 414–416

- BittyWiki Web interface, 433

- SOAP Web services, 466–467

- XML-RPC Web services, 457–458

**resizing GUI widgets, 230–231**

**the resource (REST), 410**

**resources. See also Websites for downloading;**

**Websites for further information**

- BittyWiki Web interface, 433–435

- REST, 409

**resources in REST, 410–411**

**responses (HTTP)**

- Amazon.com Web service, 444–445

- basics of, 414–417

- SOAP Web services, 467–468

- XML-RPC Web services, 459

**REST (Representational State Transfer). See also**

**HTTP: real-world REST; REST Web services**

- Amazon Web services, 443

- basics of, 408–409

further information, 409  
 operations, 410–411  
 pros and cons of, 478  
 representations of resources, 410  
 resources, 410  
 vs. SOAP, 478  
 vs. XML-RPC, 478  
 vs. XML-RPC Web services, 456

### **REST Web services, 442–455**

Amazon.com, 443–445  
 basics of, 442–443  
 BittyWiki REST API, 448–451  
 quick start, 443–445  
 wiki search-and-replace, 451–455  
 wish lists, 445–448

### **RFC 2822, 294–295**

#### **rfile, defined, 320**

#### **robots and Web services, 441–442, 449**

#### **rot13 cipher, 373**

#### **rotating files, 138–139**

#### **round function (numbers), 376**

#### **rows**

deleting, 249  
 inserting, 248

## **S**

#### **\_\_settings.py\_\_ file, 402**

#### **save(string pageName, string text) method, 473**

#### **SAX**

basics of, 274–276  
 vs. DOM, 275–276  
 parsers, 276, 279

#### **schemas (XML), 268, 270–271**

#### **scope**

defined, 104  
 naming functions, 77  
 of objects, 96–99, 104–107  
 packages, 123

#### **scripting**

with Java applications, 482–483  
 for Python, 483  
 subscribing, defined, 81

#### **scripts**

CGI, Web servers, 419–420  
 executable (Jython), 487–488  
 Jython, calling from Java, 508–510  
 Jython, controlling, 486–487

Jython, running, 485–486  
 text processing scripts, 189, 190, 191  
 turning into Web applications (CGI). See CGI  
 (Common Gateway Interface)

#### **sdist argument, 184**

#### **search utility, implementing, 216–217**

#### **search-and-replace**

using REST, 451–455  
 using SOAP, 470–472  
 using XML-RPC, 463–465

#### **searching for files**

navigating with os module, 195–198  
 search frameworks, 220–224  
 text processing scripts, 190–191

#### **Secure Socket Layer (SSL), POP3 and IMAP, 313**

#### **security (POP3 and IMAP), 313**

#### **select command, joins, 248–249**

#### **select module, 331–332**

#### **select module, single-threaded multitasking with, 329–331**

#### **self, defined, 104**

#### **sequences**

appending to lists, 45  
 multidimensional, 36  
 properties, 43–47  
 ranges of, 44  
 referencing final elements, 43–44

#### **server commands, 325**

#### **servers**

adding PyServelet to (Jython), 501–503  
 application server setup (Jython), 501  
 Chat Server. See Python Chat Server  
 IMAP servers, 309–313  
 mail servers, keeping on, 309  
 mirror servers, 317–318  
 multithreaded servers, 321–322  
 POP3 servers, 307–309  
 session state, 409–410  
 trusted servers, 333

#### **servlet containers, defined, 501**

#### **servlets**

defined, 500  
 HttpServlet (Jython), 503–504  
 PyServlet (Jython), 501–503  
 writing, 504–505

#### **session state, servers, 409–410**

#### **sets, 46–47**

#### **settings.py — file, 391**

#### **setUp method, 213**

- setup.py script, 184–185**
- shebang comment, 487, 488**
- the shell, 6**
- shortcuts, XPath, 272**
- signed numbers, 27**
- signed type numbers, 27**
- SimpleHTTPRequestHandler class, 412, 414**
- single quotes ('), 9–11**
- size (widgets), customizing, 234**
- slicing**
  - sequences, 44
  - strings, 41–42
- SmartMessages, 302–303**
- SMTP (Simple Mail Transport Protocol), 303–304**
- smtplib module, 288, 333–334**
- SOAP Web services**
  - basics of, 465
  - BittyWiki, exposing interface to, 468–470
  - BittyWiki, manipulating through, 470
  - errors, 468
  - pros and cons of, 478
  - quick start, 466
  - requests, 466–467
  - responses, 467–468
  - wiki search-and-replace using, 470–472
  - vs. XML-RPC and REST, 478
- SOAPpy package, 465, 466**
- socket programming, 314–332**
  - Chat Server. *See* Python Chat Server
  - external hostnames, binding to, 316–317
  - mirror clients, 318–320
  - mirror servers, 317–318
  - multithreaded servers, 321–322
  - Python Chat Client, 329–331
  - single-threaded multitasking, 331–332
  - socket library, 314
  - sockets, defined, 314
  - sockets basics, 314–316
  - SocketServer, 320–321
- software life cycles, testing, 224–225**
- source code**
  - defined, 5
  - used in book, xxxiii
- specialization, defined, 163**
- speed, DOM vs. SAX, 276**
- spools (mail) parsing with, 305–306**
- SQL (Structured Query Language)**
  - QUID, 247
  - writing statements, 247–249
- SQLite**
  - basics of, 250
  - connecting to, 254, 255
  - databases, creating with, 250–251
- sqlite3**
  - databases, creating, 250–251
  - Django models, creating, 401–403
  - resources, 550
- square braces ([ ])**
  - glob patterns, 141
  - lists, 37
  - types, 34
- SSL (Secure Socket Layer), POP3 and IMAP, 313**
- stability of interfaces, defined, 103**
- stack, Internet protocol, 290–291**
- stack trace, 88–89**
- standards**
  - for Web services, 442, 478–479
  - XML, 267
- Startproject command, 391**
- static database cursors, defined, 500**
- static keyword, 339**
- steps (XPath), 272**
- storage, back-end (BittyWiki), 429**
- storing data**
  - relational databases, 245
  - using lists, 45–46
- str constructor, numbers, 370**
- stream-based data (SAX), 275, 276**
- strings, 7–13**
  - % sign in, 21
  - basics of, 7
  - combining, 11–13
  - defined, 6, 7, 94
  - including different numbers in, 20–21
  - methods, 94
  - print() function, 8–9
  - quotes, 7–11
  - regular expressions, 199–202
  - REST support of, 456
  - slicing, 41–42, 44
  - string interpolation, 436–437
  - substitution using dictionaries, 148–149
  - treating like lists, 41–42
  - using to compare types, 82–83
- Structured Query Language (SQL)**
  - QUID, 247
  - writing statements, 247–249
- subclasses, creating, 165–166**
- subclassing, threads, 154**

**SubjectLister class**  
 IMAP, 309, 311, 312  
 POP3, 307–309  
**subscripting, defined, 81**  
**subscription IDs, 443**  
**sum function (numbers), 376**  
**SuperSimpleSocketServer, 315–316**  
**swing APIs, Jython, 492–493**  
**syntax, Python 3.1 changes in, 555–556**  
**sys.argv, 117**  
**sys.modules, 113–114, 122–123, 160**  
**sys.path variable (modules), 183–184**  
**System.listMethods() method, 474**  
**System.methodHelp(string funcName)**  
 method, 474  
**System.methodSignature(string funcName)**  
 method, 474

## T

### tables

creating (Jython), 497–500  
 defining, 249–250

**tags, XML, 266**

**TCP (Transmission Control Protocol), 291**

**TCP/IP, 291, 314**

**tearDown method, 213**

**telnet program, connecting with, 315–316**

### templates

basics of, 280, 396–398  
 web frameworks, 389

**templates and views (Django), 398–403**

models, configuring database settings, 401–403  
 models, 401

templates basics, 396–398

using, 398–401

views, creating, 394–396

**terse protocols, 333**

**test cases, 209–212**

**test fixtures, 213–216**

### test suites

basics of, 210–212

writing, 217–220

**TestCase classes, 209, 210, 213**

**testing, 207–226. See also XP testing**

#### methodology

assertions, 208–209

basics of, 207

files, 202–203

full systems, 483

from Jython, 506–507

modules, 124, 176–177

packages, 124

PyUnit basics, 207

software life cycles, 224–225

summary, 225–226

### tests

reversing outcome of, 56

test cases, 209–212

test fixtures, 213–216

test suites, 210–212

tests within tests, 58–60

### text

adding to elements (XML), 282–283

appending to files, 129

chat text, 325

mirroring with MirrorServer, 318

Python 3.1 changes, 554

### text files

reading, 130–131

writing, 128–129

### text processing, 189–204

clipping logs, 191

defined, 189

file systems, navigating. *See* file systems,  
 navigating with os module

files, searching for, 190–191

mail, sifting through, 192

regular expressions. *See* regular expressions  
 and the re module

usefulness of, 189–190

**TextTestRunner class, 211**

**ThreadingMixIn class (SocketServer module),  
 321–322**

**threads, multiple tasks, 153**

**Tkinter, creating GUI widgets with, 229–237**

appearance, 233–234

configuring options, 231

dialog boxes, 236–237

exercises, 238

layouts, creating, 232–233

packing order, 233

radio buttons and checkboxes, 235–236

resizing, 230–231

simple program, 229–230

summary, 238

Tkinter basics, 229

widgets, applying actions to, 231–232

widgets, types of, 237

**Tkinter resources, 550**

**Tomcat, servlets containers, 501**

**tools**

2to3 tool, 558

Jython, 506

toolkits for writing GUIs, 228–229

Workflow toolkit resources, 550

**top command (POP3), 308, 309**

**toprettyxml utility, 278**

**transactions (databases)**

committing, 255

connections, 260

defined, 245

support for, 245

working with and committing results, 260–261

**Transmission Control Protocol (TCP), 291**

**transport layer, 291**

**triple quotes (“ “ “), 7, 10**

**true values. See values, True and False**

**truncating numbers, defined, 368**

**trusted servers, 333**

**try statements, 65–67**

**tuples**

arrays, 381

basics of, 34–37

vs. lists, 38–39

slicing, 44

**type function, 16, 82**

**type objects, 353, 354**

**TypeError, 66, 81**

**types**

content types (MIME), 297

determining with type function, 82

immutable, 37, 42

lists as, 37–39

number types, 16, 19–21

special types, 42–43

strings, to compare, 82–83

tuples as, 34–37

## U

**\_\_urls.py\_\_ file, 395**

**UID (unique ID), 312–313**

**Unicode in Python 3.1, 554**

**unit tests, 209–212, 483**

**unittest, 209**

**unsigned numbers, 27**

**unsubscribeable, defined, 81**

**URLconf, Django, 395–396**

**urllib module, 444**

**URLs**

URL mapping and web frameworks, 388

URLconf, 395, 398

**\_\_urls.py\_\_ file, 394, 395**

**user authentication, 388**

**user interfaces. See also GUI (graphical user interface); GUIs, writing**

creating (Jython), 492–494

## V

**values**

dictionaries, 39–40

difference comparison, 53

equality comparison, 52–53

form values, accessing (HTML), 423–427

for functions, 79–80

greater/less than, 54–55

more than one comparison, 56–60

providing for named functions, 79–80

returning from C, 345–346

setting for parameters, 83–84

**values, True and False**

equality comparison, 51–53

reversing, 56

as special types, 42–43

**variable \_\_debug\_\_, 209**

**variables, 31–48**

defined, 32, 93

dictionaries, 39–41

environment variables (CGI), 420–422

exercises, 48

lists, 37–39

lists, for storing data, 45–46

lists, joining, 45

named, 93

names for data, 31–34

ranges of sequences, 44

referencing final elements, 43–44

sequence properties, 43–47

sets, 46–47

special types, 42–43

strings, treating like lists, 41–42

summary, 47–48

tuples, 34–37

**vectors, defined, 116**

**verbs (HTTP), 411**

**views. See also templates and views (Django)**

- creating (Django), 394–396
- MVC architecture, 391
- returning (Python 3.1), 553

**W****the Web**

- background of, 407
- REST, 408–411
- virtues of, 408

**web application frameworks, 387–389****Web applications, 407–441. See also Web services**

- accessing form values, 423–427
- benefits for developers, 407
- exercises, 480
- HTML forms, 422–423
- HTTP: real-world REST. *See* HTTP: real-world REST
- REST, 408–411
- summary, 480
- turning CGI scripts into. *See* CGI (Common Gateway Interface)
- as Web services, 480
- wikis. *See* wikis, building

**Web servers**

- CGI scripts, 419–420
- simple, 411–412
- Visible Web Server, 412–415

**Web service APIs, documenting, 472–478**

- human-readable API documentation, 473
- WSDL, 475–478
- XML-RPC introspection API, 474

**Web services, 441–480**

- basics of, 441–442
- defined, 441
- documenting APIs. *See* Web service APIs, documenting
- etiquette, 479–480
- REST Web services. *See* Rest Web services
- robots, 441–442
- standards, choosing, 478–479
- standards for, 442
- Web applications as, 480
- XML-RPC. *See* XML-RPC Web services

**Websites for downloading**

- database modules, 252
- Django, 390
- Eclipse, 506

- Jython, 483
- LAME package, 347
- LiveHTTPHeaders extension, 415
- lxml, 280
- PyAmazon, 444
- Python, 6, 389
- SOAP libraries, 466
- software required for book examples, 549–550
- source code used in book, xxxiii
- Tomcat, 501
- toolkits for writing GUIs, 228
- Web-Sniffer, 415

**Websites for further information**

- 2to3 tool, 558
- DB API, 262
- Django, 389
- error codes, 416
- floating-point numbers, 19
- the Grinder, 483
- Java EE, 503
- jEdit text editor and plugins, 506
- LAME Project, 347
- libgmail project, 313
- lxml, 284
- namespaces, 267
- ORM, 245
- Python, 551
- Python 3.1 changes, 553
- Python API documentation from C, 338
- Python documentation, 184
- REST, 409
- SQL, 247
- Tkinter, 229
- wiki design principles, 429
- wxPython, 229
- XPath shortcuts, 272

**webmail, vs. e-mail, 313****Web-Sniffer, 415****wfile, defined, 320****wftk resources, 550****while . . . : statements, 62–63****while operations, 60–62****widgets. See also Tkinter, creating GUI widgets with**

- defined, 229

**Wikipedia, 428****wikis**

- search-and-replace using REST, 451–455
- search-and-replace using SOAP, 470–472
- search-and-replace using XML-RPC, 463–465

## **wikis, building, 428–441**

- BittyWiki core library, 429–432
- BittyWiki Web interface. *See* BittyWiki Web interface
- pages, creating, 432
- wiki basics, 428–429
- WikiWords, 430

## **WikiSpiderREST.py command, 455, 463**

## **WikiSpiderSOAP.py client, 470–473**

## **WikiSpiderXMLRPC.py script, 463–465**

## **WikiWords, 428, 430**

## **wildcards. *See also* globbing**

- vs. regular expressions, 199

## **Willison, Simon, 389**

## **Windows, installing Python on, 5–6**

## **wish lists (Amazon.com), 445–448**

## **Wrox P2P, xxxiii–xxxiv**

## **WSDL, documenting Web service APIs, 475–478**

## **wxPython, 229**

## **wxWidgets, 229**

# X

## **XML (Extensible Markup Language), 265–285**

- basics of, 265, 267
- DOM, 275–276
- DOM parsers, 276–278
- DTDs, 268–270
- element classes, 281–283
- exercises, 285
- extensibility and standards, 267
- as hierarchical language, 265–267
- HTML as subset of, 272–274
- libraries, 274
- lxml, 280, 283–284

- SAX, 274–276
- SAX parsers, 276, 279
- schemas, 268, 270–271
- summary, 285
- XPath, 272
- XSLT, 280

## **xml.dom.minidom parser, 277–278**

## **XML-RPC Web services, 456–465**

- basics of, 456–457
- BittyWiki API through, 460–463
- errors, 459–460
- introspection API, 474
- pros and cons of, 478
- requests, 457–458
- responses, 459
- vs. REST, 456, 478
- SOAP. *See* SOAP Web services
- vs. SOAP, 478
- wiki search-and-replace using, 463–465

## **xmlrpc.server function, 474**

## **xml.sax parser, 276**

## **XP testing methodology, 216–224**

- search frameworks, 220–221
- search parameters, adding, 222–224
- search utility, implementing, 216–217
- test suites, writing, 217–220

## **XPath, 272**

## **XSLT (eXtensible Stylesheet Language Transformations), 280**

# Z

## **Zawinski, Jamie, 288–289**

## **zxJDBC module (Jython), 494, 495**