

# Index

## SYMBOLS AND NUMERICS

### \* (asterisk)

- \* (element grouping operator in DTD), 279
- \*\* (exponentiation operator), 406, 413
- \* (in format specifier), 410
- \* (in `import` command), 103
- \* (multiplication operator), 19, 412
- \* (regular expression wildcard), 186–187
- \* (wildcard in glob pattern), 122
- \* (XPath axis shortcut), 282

### @ (at sign), XPath axis shortcut, 282

### \ (backslash)

- \ (escape character), 7, 185
- \ (in Windows path names), 109, 178
- \n (newline character), 7–8

### ^ (caret), regular expression wildcard, 187

### : (colon)

- : (preceding code block), 48–49
- : (separating dictionary keys and values), 35

### { } (curly braces), enclosing dictionaries, 34

### \$ (dollar sign), regular expression wildcard, 187

### “...” (double quotes), enclosing strings, 6, 618

### = (equal sign)

- = (assignment operator), 28
- == (equality operator), 43–44

### ! (exclamation point)

- != (inequality operator), 45
- ! (wildcard in glob pattern), 122

### # (hash mark)

- # (preceding comments), 65–66
- #! (shebang comment line), 60, 545

### - (hyphen)

- (in format specifier), 10
- (subtraction operator), 19, 412

### < (left angle bracket)

- < (less than operator), 45–46
- <= (less than or equal operator), 47

### () (parentheses)

- enclosing list of DTD elements, 279
- enclosing tuples, 30

### . (period), regular expression wildcard, 185

### % (percent sign)

- % (escaping in strings), 18–19
- % (preceding format specifier), 9
- % (remainder operator), 20, 414
- % (string formatting operator), 408
- %d (format specifier), 18, 408
- %E (format specifier), 18
- %f (format specifier), 18, 22, 408
- %o (format specifier), 24, 409
- %#o (format specifier), 409
- %s (format specifier), 9
- %x (format specifier), 24, 409
- %X (format specifier), 24
- %#x (format specifier), 409

### + (plus sign)

- + (addition operator), 19, 412
- + (combining strings), 8
- + (element grouping operator in DTD), 279

### ? (question mark)

- ? (element grouping operator in DTD), 279
- ? (wildcard in glob pattern), 122

### > (right angle bracket)

- > (append in `print` statement), 111
- > (greater than operator), 45–46
- >= (greater than or equal operator), 47
- >> (prompt in Python shell), 5

### ' (single quote), element grouping operator in DTD, 279

### '...' (single quotes), enclosing strings, 6, 618

### / (slash)

- / (division operator), 19, 412
- // (floor division operator), 413
- / (in Unix path names), 178
- // (XPath axis shortcut), 282

## [ ] (square brackets)

- dereferencing dictionaries, 35
- dereferencing lists, 33
- dereferencing tuples, 31
- enclosing lists, 33
- regular expression wildcard, 187
- wildcard in glob pattern, 122
- “...” or “...” (triple quotes)**
  - enclosing documentation, 156–157
  - enclosing multi-line strings, 7–8, 618
- \_ (underscore), in variable names, 30**
- | (vertical bar)**
  - | (OR between DTD elements), 279
  - | (preceding optional arguments), 362
- 0 (False value), 36, 38**
- 0x, preceding hexadecimal literals, 406**
- 1 (True value), 36, 38**
- 4DOM DOM implementation, 285**
- 4Suite library, 294**
- 127.0.0.1, IP address for localhost, 613**
- 200 status code, 467**
- 403 status code, 467**
- 404 status code, 467**
- 500 status code, 467**

## A

- abs function, 414, 417**
- absolute path, converting relative path to, 116**
- absolute value, abs function for, 414, 417**
- abspath function, os.path module, 116**
- acos function, 415**
- action queue handler, 456–457**
- adapters, wftk, 436**
- addition operator (+), 19, 412**
- agents, workflow, 456**
- aggregator, RSS, 301–302**
- \_\_all\_\_ variable, 103–104, 155**
- Amazon wish lists, 497–500**
- Amazon.com web service, 495–497, 537**
- and operator, 48**
- anonymous functions, 127–128, 613**
- anonymous variables, 613**
- anydbm module, 250**
- apilevel global, DB API, 272**
- append method, list, 34**
- application layer, 309**

## arguments. See parameters

- argv list, 99–100**
- arithmetic**
  - operators, 19–20, 412–414
  - order of evaluation for, 21–22
  - precision of, 21–22
- array module, 420–421**
- array object**
  - from array module, 420–421
  - from numpy module, 422–424
- arrays**
  - definition of, 418–419
  - lists or tuples for, choosing, 419
  - mean of, 414–415
  - multidimensional, 418
  - standard deviation of, 419–420, 423–424
- asin function, 415**
- assert statement, 192–193**
- AssertionError, 192**
- assertions, 191–193**
- assignment operator (=), 28**
- asterisk (\*)**
  - \* (element grouping operator in DTD), 279
  - \*\* (exponentiation operator), 406, 413
  - \* (in format specifier), 410
  - \* (in import command), 103
  - \* (multiplication operator), 19, 412
  - \* (regular expression wildcard), 186–187
  - \* (wildcard in glob pattern), 122
  - \* (XPath axis shortcut), 282
- at sign (@), XPath axis shortcut, 282**
- atan function, 415**
- atan2 function, 415**
- attachments, e-mail, 315–316**
- attributes, XML, 275**
- auditing, 433–435**
- axis, XPath, 282**

## B

- backslash (\)**
  - \ (escape character), 7, 185
  - \ (in Windows path names), 109, 178
  - \n, newline character, 7–8
- base 16 (hexadecimal) notation**
  - definition of, 24, 615
  - formatting numbers as, 409
  - specifying literals in, 406

**base 10 notation, 24**

**base 8 (octal) notation**

definition of, 24, 617

formatting numbers as, 409

**Base64 encoding, 313–314, 613**

`base64` module, 141

**BaseRequestHandler class, 337–338**

*Beginning XML* (Hunter, et al.), 277

**binary copy protection module, 388**

**Binary Large Object (BLOB), 430**

**binary pickle protocol, 125**

**BitTorrent, 354, 613**

**BittyWiki example**

core library for, 481–484

creating wiki pages from Python session, 483–484

making executable, 493

markup used by, 486–492

resources for, 484–486

REST API document for, 529

REST API for, 500–503

SOAP API document for, 530

SOAP interface for, 525–527

web interface for, 484–493

WSDL proxy for, 533–534

XML-RPC API document for, 529–530

XML-RPC interface for, 514–517

**BLOB (Binary Large Object), 430**

**blocks**

definition of, 2

indentation indicating, 49

**Boa Constructor, 224**

**body**

e-mail, 311, 312

HTTP, 615

**braces ({}), enclosing dictionaries, 34**

**brackets, square ([])**

dereferencing dictionaries, 35

dereferencing lists, 33

dereferencing tuples, 31

enclosing lists, 33

regular expression wildcard, 187

wildcard in glob pattern, 122

`break` statement, 53–54

`build` command, 360

**built-in classes, listing, 144**

**built-in functions**

definition of, 10

listing, 144

**built-in variables, listing, 144**

**business processes, modeling of, 427**

## C

**C language**

extension modules written in

building and installing, 358–360

definition of, 356–358

passing parameters from Python to, 360–363

returning values to Python, 363–364

performance of, compared to Python, 355

using Python objects from, 380–382

**C Python, compared to Jython, 541, 569–570**

`cachedir` directory, Jython, 547

**call stack, 74–75, 613**

**CANVAS example**

background of, 385–386

GEOip library used by, 402

IP address, code obtaining, 399

penetration tests by, 389

pure-Python approach used by, 387

subscriptions used by, 394–395

**caret (^), regular expression wildcard, 187**

`ceil` function, 415

**CGI (Common Gateway Interface)**

definition of, 468–469, 613

environment variables for, 471–473

HTML forms and, 473–479

scripts, running, 469–470

web server's role in executing scripts, 470–471

`.cgi` files, 469

`cgi` module, 474–479

`cgitb` module, 471

**CGIXMLRPCRequestHandler class, 514, 530**

**channels**

IRC, 340

RSS, 297

**Character Large Object (CLOB), 430**

**characters. See also strings**

converting to numbers, 410–411

escape character, 6–7

**chat client, Python, 346–350**

**chat server, Python, 340–346**

`cl` **command, 358, 367, 368, 379**

`class` **keyword, 81–82**

**classes. See also objects; subclasses**

creating objects from, 82

defining, 81–82, 151–152

definition of, 613

documenting, 156–157

extending, 153

interface for, 83

methods for, 83–89

repurposing, 89

when to use, 95

`clicked` **signal, 214–215**

**client-server architecture**

definition of, 354, 614

REST as, 461

**CLOB (Character Large Object), 430**

`close` **method**

connection object, 266

cursor object, 266

`cmath` **module, 15, 418**

**code. See blocks; functions; programs**

**codeEditor**

compared to idle editor, 3

definition of, 3

line numbers in, 3–4

Python shell in, 4–5, 16–17

starting, 3

`codeEditor` **command, 3**

**colon (:)**

: (preceding code block), 48–49

: (separating dictionary keys and values), 35

**command-line parameters, accessing, 99–100, 134–137**

**comments. See also docstring**

creating, 65–66

definition of, 614

**commercial programs, using Python for**

development environment for, 395–396

distribution of, 400–401

how much Python to use for, 386–387

libraries for, 401–402

licensing issues, 387–389

piracy issues, 386–387

as platforms, 395

pricing strategies, 389–395

Python problems concerning, 397–400

Python programmers for, finding, 396–397

reasons for, 385

Web services and, 388–389

`commit` **method**

connection object, 262, 271

cursor object, 266

**Common Gateway Interface**

definition of, 468–469, 613

environment variables for, 471–473

HTML forms and, 473–479

scripts, running, 469–470

web server's role in executing scripts, 470–471

**comparison operators, 43–47**

**comparisons**

combining, 48

decisions and, 48–50

equality, 43–44

greater than, 45–46

greater than or equal, 47

inequality, 45

less than, 45–46

less than or equal, 47

reversing True or False, 47

**compiling, 104**

`complex` **method, 417**

**complex numbers, 15, 416–418**

`complex` **object, 417**

`conjugate` **method, complex object, 418**

`connection` **object**

DB API, 263–264

Gadfly, 262

`__contains__` **method, dictionary, 36**

**content types, MIME, 314–315, 614**

**context (environment) of program, 4**

`continue` **statement, 54**

`copy` **function, shutil module, 119**

**copy protection module, 388**

`cos` **function, 415**

`cosh` **function, 415**

`cPickle` **module, 125**

**C-Python, 541**

**Create, Read, Update, Delete (CRUD), 257**

`create table` **statement, 259–260**

**CRUD (Create, Read, Update, Delete), 257**

`ctime` function, `time` module, 118

**Cunningham, Ward (wiki inventor), 479**

**curly braces ({}), enclosing dictionaries, 34**

`cursor` method, `connection` object, 264

**Cursors, DB API, 264–271**

## D

**%d format specifier, 18, 408**

**data link layer, 308**

**data types**

dictionaries

accessing keys or values in, 35–36

creating, 34–35

definition of, 34, 614

equality of, 44

keys in, 35

string substitutions using, 133–134

values in, 35

lists

adding elements to, 34

appending, 40

for arrays, when to use, 419

changing elements of, 33

creating, 33

creating with ranges, 131–133

decisions within (list comprehension), 130

definition of, 33, 616

equality of, 44

last elements of, referencing, 38–39

popping elements from, 40–41

ranges of, referencing (slicing), 39–40

reducing (running function on all elements of),  
128–129

treating strings as, 36–37

visiting elements in without loops, 129–130

numbers

arithmetic operators for, 412–414

arithmetic, performing, 19–21

complex numbers, 416–418

converting characters to, 410–411

displaying, 17–19, 21, 22

formatting, 408–410

formatting in base 8 or base 16, 24

math functions for, 414–415

math module for, 415–416

negative numbers, 24, 609

type of, determining, 14

types of, 13–14, 405–408

strings

combining (concatenating), 8

converting to integers, 406

definition of, 5, 619

displaying, 10

displaying numbers as, 17–19

escaping characters in, 6–7

formatting, 9–10

immutable, problems with, 399–400

newline characters in, 7–8

quotes used to enter, 6–8

substitutions using templates, 610

substitutions with dictionaries, 133–134

treating as a list (slicing), 36–37

tuples

appending, 40

for arrays, when to use, 419

creating, 30–31

definition of, 30, 619

equality of, 44

last elements of, referencing, 38–39

layers of, 31–33

one-element tuples, 32

ranges of, referencing (slicing), 39–40

reducing (running function on all elements of),  
128–129

visiting elements in without loops, 129–130

**database manager (DBM), 614**

**DatabaseError, DB API, 273**

**databases**

accessing from Jython, 552–558

choosing, 255

DB API

connecting to database, 263–264

Cursors for, 264–271

definition of, 262–263, 614

deleting data, 270–271

downloading modules for, 263

errors, list of, 272–273

globals for, 272

inserting new data, 264–266

querying database, 266–269

transactions, 271–272

updating data, 269–270

### **databases (continued)**

#### **DBM persistent dictionaries**

- accessing, 252–255
- creating, 251–252
- definition of, 250
- modules for, 250–251
- when to use, 255

#### features for, 249

#### Gadfly database, 260–262

#### relational databases

- connecting to database, 260–261
- defining tables, 259–260
- definition of, 255–257, 618
- deleting data, 257–259
- for document management, 429–431, 441–448
- inserting new data, 257–259
- querying database, 257–259
- SQL (Structured Query Language), 257–261, 619
- updating data, 257–259
  - as `wftk` repository, 438–446
  - when to use, 255

### **DataError, DB API, 273**

#### **DB API**

- connecting to database, 263–264
- Cursors for, 264–271
- definition of, 262–263, 614
- deleting data, 270–271
- downloading modules for, 263
- errors, list of, 272–273
- globals for, 272
- inserting new data, 264–266
- querying database, 266–269
- transactions, 271–272
- updating data, 269–270

#### `dbhash` **module, 250**

### **DBM (database manager), 614**

#### `dbm` **module, 250**

#### **DBM persistent dictionaries**

- accessing, 252–255
- creating, 251–252
- definition of, 250
- modules for, 250–251
- when to use, 255

#### `__debug__` **variable, 192–193**

### **debugging**

- log level 6 for, 438
- threads, 216–217, 225, 399
- with `walk` function, 181
- web applications, 466

### **decisions, 48–50**

#### `def` **statement, 61–62**

#### Deferred **object, 351–353**

#### **definitions element, WSDL, 531–532**

#### `del` **statement, 111**

#### **DELETE command, 462, 473**

#### `delete` **statement, SQL, 257–259**

#### **del.icio.us web service, 537**

#### **dereference, 30–31**

#### `destroy` **signal, 214–215**

### **dictionaries**

- accessing keys or values in, 35–36
- creating, 34–35
- definition of, 34, 614
- equality of, 44
- keys in, 35
- string substitutions using, 133–134
- values in, 35

#### `dir` **function, 80, 144**

#### **directories. See also paths**

- creating, 121
- current, determining, 179
- determining whether path is, 118
- joining into path, 179
- listing, 116–118
- listing recursively with subdirectories, 118–119, 179
- navigation of, 176
- removing, 121–122
- running functions on all directories in tree, 180
- separator characters for, 178
- statistics about, 179
- for user and group management, 431–432

#### `disutils` **package, 171–173, 359–360**

#### **division operator (/), 19, 412**

#### `.dll` **files, 356**

### **DNS (Domain Name System), 309, 614**

#### `__doc__` **string, 63–64**

**docstring (documentation string).**

**See also comments**

- for classes, 82, 83, 92
- for functions, 63–64
- in test cases, 203

**document events, SAX, 287–288****document management**

- auditing and, 434–435
- file system for, 428–429
- relational databases for, 429–430, 441–448
- retention of documents, 430–431, 434–435, 446–448
- version control system for, 430

**document models**

- definition of, 614
- validating XML against, 285–287
- when to use, 278

**Document Object Model. See DOM****document objects, DOM, 288****document root, XML, 276****Document Type Definition. See DTD****documentation, for web service API, 529–534.**

**See also docstring**

**documentation string. See docstring****documents, REST resources as, 461–462.**

**See also files**

**dollar sign (\$), regular expression wildcard, 187****DOM (Document Object Model)**

- definition of, 288–289, 614
- implementations of, 285
- when to use, 289

**Domain Name System (DNS), 309, 614****double quotes (“...”), enclosing strings, 6, 618****DTD (Document Type Definition)**

- compared to XML, 280
- definition of, 277, 278, 614
- example of, 278–280
- limitations of, 280
- for RSS, 297

**dumbdbm module, 251****dump function, pickle module, 123–124****dynamic cursor, 558****E****e constant, 415****%E format specifier, 18****Eclipse IDE, 565****elif statement, 50****else statement**

- in if block, 50
- in repetitions, 54
- in try block, 57

**e-mail**

- attachments, 315–316
- compared to webmail, 331
- file format for, 311–312
- MIME messages, 313–321
- quote about, by Jamie Zawinski, 305
- retrieving, 323–331
- retrieving from IMAP server, 327–331
- retrieving from POP3 server, 325–327
- searching, 178
- sending, 305–307, 311–312, 321–323
- SMTP and, 321–323

**email module, 312, 321, 323, 326****email.Message class, 318****email.Mime\* classes, 318****email.MimeMultipart class, 318****enactment, in workflow, 433****encapsulation, 151, 156, 614****encodings, MIME, 313–314****encryption**

- definition of, 614
- of passwords, 140–141

**enterprise applications**

- auditing and, 433–435
- definition of, 427–428
- document management, 428–431, 434–435
- python-ldap module for, 448–453
- user and group management, 431–432
- wftk workflow toolkit
  - action queue handler, 456–457
  - definition of, 435–436
  - repository, in database, defining and accessing, 438–439

### **enterprise applications (continued)**

- repository, in database, for document management, 441–448
- repository, in database, storing records in, 439–446
- repository, in directory files, defining and accessing, 436–438
- repository, in directory files, storing records in, 438–439
- resources for, 607
- workflow trigger, 453–456
- workflow, 432–433

### **Enterprise license, 389**

### **environment (context) of program, 4**

### **ephemeral port, 614**

### **ephemeral port number, 334**

### **equal sign (=)**

- = (assignment operator), 28
- == (equality operator), 43–44

### **Error error, DB API, 273**

### **errors. See exceptions**

### **escape character, 6–7**

### **escape sequences, 615**

### **escaping, 6–7**

### **event-driven parsing, 288**

### **events, widgets, 214–215**

### **examples**

- BittyWiki example
  - core library for, 481–484
  - creating wiki pages from Python session, 483–484
  - making executable, 493
  - markup used by, 486–492
  - resources for, 484–486
  - REST API document for, 529
  - REST API for, 500–503
  - SOAP API document for, 530
  - SOAP interface for, 525–527
  - web interface for, 484–493
  - WSDL proxy for, 533–534
  - XML-RPC API document for, 529–530
  - XML-RPC interface for, 514–517
- CANVAS example
  - background of, 385–386
  - GEoip library used by, 402

- IP address, code obtaining, 399
- penetration tests by, 389
- pure-Python approach used by, 387
- subscriptions used by, 394–395
- of DTD (Document Type Definition), 278–280
- of Glade GUI construction kit, 231–238, 241–248
- Google API example, 520–522
- LAME project
  - definition of, 364
  - extension module for, 368–380
  - using, 364–367
- Meerkat API example, 509–511
- mirror client example, 336–337
- mirror server example, 335–336
- PyRAP example, 231–238, 241–248
- search utility examples, 200–207
- of Widget Tree, Glade, 241–248

### **except statement, 55–57**

### **exceptions**

- AssertionError, 192
- from within call stack, 74–75
- DB API, 272–273
- of DB API, list of, 272–273
- definition of, 55
- documenting, 156–157
- of `Fault` element of SOAP, 524–525
- of fault response of XML-RPC, 513–514
- flagging, 73–74
- handling, 55–57
- IndexError, 33, 34

### **exceptions (continued)**

- IOError, 113
- KeyError, 55, 56
- OSError, 114
- from Python shell, 23–24
- of Python shell, 23–24
- SOAP, 524–525
- TypeError, 32, 56, 68
- types of, 56
- XML-RPC, 513–514

### **exclamation point (!)**

- != (inequality operator), 45
- ! (wildcard in glob pattern), 122

### **exec function, os module, 137**

`execfile` **method**, `PythonInterpreter` **class**, 566

`execute` **method**, `cursor` **object**, 266

`exp` **function**, 415

Expat **XML parser**, 285

**exploit program**, 389

**exponential notation**, 408

**exponentiation operator (`**`)**, 406, 413

`extend` **method**, 34, 40

**eXtensible Markup Language (XML)**

compared to DTD, 280

created by Glade, 230–231

definition of, 275–277, 620

libraries for, 285

parsing

with DOM, 288–289, 290–292

with SAX, 287–288, 289, 290, 292–293

standards for, 277

transforming with XSLT, 293–296

validating, 285–287, 621

XPath expressions in, 282

**Extensible Style Language for Transformations (XSLT)**

definition of, 277, 293–294, 621

transforming XML with, 294–296

**Extensible Style Language Formatting Objects (XSL-FO)**, 277, 621

**extension module**

building, 358–360

compiling, 358, 367, 368, 379

definition of, 356–358

include files for, 356

initialization function for, 358

installing, 360

LAME project

definition of, 364

extension module for, 368–380

using, 364–367

method table for, 357, 361

names of C functions in, 357

passing parameters from Python to C, 360–363

returning values from C to Python, 363–364

signatures of C functions in, 356, 360

**Extreme Programming (XP)**, **test suites for**, 199–205

## F

**%f format specifier**, 18, 22, 408

False **value (0)**

definition of, 36, 38

as result of comparisons, 43

Fault **element**, **SOAP**, 524–525

**fault response**, **XML-RPC**, 513–514

**feeds**, **RSS**, 297–301

`fetchall` **method**, `cursor` **object**, 267

`fetchone` **method**, `cursor` **object**, 267

**Fielding, Roy (REST definition)**, 460

**FieldStorage class**, 474–475

**file object**

definition of, 109

deleting, 111, 112

line lengths in, printing, 112

reading, 111–112

writing (creating), 110–111

**files**

closing, 111, 112

copying, 119

deleting, 119–120

determining whether path is, 118

for document management, 428–429

errors accessing, 113

globbing, 122–123

line lengths in, printing, 112

moving, 119

packages

creating, 101–103

definition of, 95, 101–102, 617

importing all elements of, 103–104

re-importing, 104–106

testing, 106

permissions, 120

pickle files, writing, 123–125

reading, 111–112

renaming, 119

rotating old versions of, 120–121

saving program in, 59–61

searching for

examples of, 200–207

by file name, 176–177

by file type, 181–184, 187–188

splitting extension from end of, 115–116

### files (continued)

- statistics about, 179
- writing (creating), 110–111

**filter function, using with lambda statement, 127–128, 186**

**findgtk.py module, 211–212**

**firewall**

- effect on binding to external hostname, 335
- penetration tests and, 389

**500 status code, 467**

**Flickr web service, 537**

**floats (floating-point numbers)**

- converting strings to, 408
- converting to integers, 406
- definition of, 14, 407, 615
- exponential notation for, 408
- high-precision, 15
- precision of, 407, 408
- scientific notation for, 17

**floor division operator (//), 413**

**floor function, 415**

**for statement, 51–54**

**foreign key, in relational database, 256**

**fork function, os module, 137**

**ForkingMixIn class, 339–340**

**<FORM> tag, 473**

**format specifiers**

- asterisk (\*) in, 410
- %d format specifier, 18, 408
- %E format specifier, 18
- %f format specifier, 18, 22, 408
- hyphen (-) in, 10
- %o format specifier, 24, 409
- %#o format specifier, 409
- %s format specifier, 9
- using, 9
- %x format specifier, 24, 409
- %X format specifier, 24
- %#x format specifier, 409

**forward slash**

- / (division operator), 19, 412
- // (floor division operator), 413
- / (in Unix path names), 178
- // (XPath axis shortcut), 282

**403 status code, 467**

**404 status code, 467**

**4DOM DOM implementation, 285**

**4Suite library, 294**

**from modifier, import command, 101, 145**

**functions. See also specific functions**

- anonymous functions, 127–128, 613
- calling from within other functions, 71–72
- creating, 61–62
- creating module with, 150
- creating within other functions (nesting), 72–73
- definition of, 10, 59, 615
- documenting, 63–64, 156–157
- layers of invocations of, 74–75
- naming, 62
- parameters for, 66–71
- recursive functions, 115
- scope of variables and, 64–65
- as signal handlers, 214–215

## G

**Gadfly database, 260–262**

**gcc command, 358, 367, 368, 379**

**gdbm module, 251**

**GEOip library, 402**

**GET command, 462, 473**

**getcwd function, os module, 179, 180**

**getmtime function, os.path module, 118–119**

**getopt function, getopt module, 135–136**

**getopt module, 134–137**

**getsize function, os.path module, 118–119**

**Glade GUI construction kit**

- definition of, 223–224
- example using, 231–238, 241–248
- installing, 225
- project, creating, 227
- starting, 226
- Widget Tree, 238–241
- widgets, creating in window, 228–230
- window, creating, 227–228
- XML created by, 230–231

**glob function, glob module, 122–123**

**glob module, 122–123**

**globbing, 122–123**

**glue, 356. See also extension module**

**Gmail, 331****GNU DBM library, 251**`gnu_getopt` function, `getopt` module, 136–137**Google API example, 520–522****Google web service, 537****googol, representing, 406****greater than operator (>), 45–46****greater than or equal operator (>=), 47****groups, managing, 431–432****GUI builders, 223–225****GUIs. See also pyGTK toolkit**

GUI builders for, 223–225

toolkits for writing with Python, 209–210

## H

**hash mark (#)**

# (preceding comments), 65–66

#! (shebang comment line), 60, 545

**headers**

definition of, 615

e-mail, 311–312

HTTP

CGI script outputting, 470

definition of, 467–468, 615

`help` function, 157–162**hexadecimal (base 16) notation**

definition of, 24, 615

formatting numbers as, 409

specifying literals in, 406

**high-precision floating-point numbers, 15****hostname**

binding to, 332

definition of, 615

DNS managing, 309

external, binding to, 334–335

localhost, 309, 613, 616

using instead of IP addresses, 310

**HSqIDB database, 554****HTML forms**

definition of, 473–474

parsing, 474–479

**HTML (HyperText Markup Language)**

parsing, 283–285

relationship to XML, 282

`htmllib` module, 284–285`HTMLParser` class, 283–284**HTTP (HyperText Transfer Protocol)**

body, 615

commands for, 462–463, 473

definition of, 615

design guidelines for, 468

headers

CGI script outputting, 470

definition of, 467–468, 615

request

definition of, 466–467

viewing with web server, 464–466

request, definition of, 615

response

definition of, 467–468

viewing with web server, 464–466

response, definition of, 615

session length of, 461

status code, 467–468, 615

verb, 462–463, 473, 615

visible web server using, 464–466

web server using, 463

`HTTPServer` object, 463**HttpServlet class, 561–564****HTTP\_USER\_AGENT variable, 473****Hunter, David (*Beginning XML*), 277****hyperbolic functions, 415****HyperCard, GUI builder for, 225****HyperText Markup Language (HTML)**

parsing, 283–285

relationship to XML, 282

**HyperText Transfer Protocol. See HTTP****hyphen (-)**

- (in format specifier), 10

- (subtraction operator), 19, 412

`hypot` function, 415

## I

**IANA (Internet Assigned Numbers Authority), 310–311****IDE (Integrated Development Environment), 395–396, 565****idempotent action, 462, 615****idle editor, compared to codeEditor, 3****if statements, 48–50**

`imag` **attribute**, **complex object**, 417

## **imaginary number**

- creating complex numbers from, 416
- definition of, 14–15, 615

## **IMAP (Internet Message Access Protocol)**

- definition of, 615
- unique message IDs used by, 330–331

## **IMAP server**

- retrieving e-mail from, 327–331
- secure, 331

## **imaplib module, 327–331**

## **immutable**

- definition of, 32
- strings as, problems with, 399–400

## **import command**

- `from` modifier, 101, 145
- using, 96–97, 145
- version 2.4 enhancements to, 611

## **indentation, 49**

## **index, database, 441–442**

## **IndexError, 33, 34**

## **inequality operator (!=), 45**

## **inf (infinity), 19**

## **infinite loop, 52–53, 616**

## **infrastructure platforms, 427**

## **inheritance, 151, 616**

## **`__init__` method, 152**

## **`__init__.py` file, 101–103**

## **input, 10. *See also* Input/Output (I/O)**

## **Input/Output (I/O). *See also* files; pickles**

- definition of, 10, 109, 616
- file exceptions, 113
- manipulating files, 119–121
- reading files, 111–112
- writing files, 110–111

## **insert statement, SQL, 257–259**

## **install command, 360**

## **int method, 406**

## **integers. *See also* longs**

- automatically converted to long if too large, 609
- converting to, from other types, 406
- definition of, 14, 406, 616
- internal representation of, 407

## **Integrated Development Environment (IDE), 395–396, 565**

## **IntegrityError, DB API, 273**

## **interface for a class, 83**

## **interface methods, 83–84, 85–87**

## **InterfaceError, DB API, 273**

## **internal methods, 84–85**

## **InternalError, DB API, 273**

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

## **Internet Assigned Numbers Authority (IANA), 310–311**

## **Internet Message Access Protocol (IMAP)**

- definition of, 615
- unique message IDs used by, 330–331

## **Internet Protocol. *See* IP**

## **Internet protocol stack, 308–309, 618**

## **Internet Relay Chat (IRC), 340, 616**

## **interpreted languages, 2**

## **introspection API, XML-RPC, 530–531**

## **I/O (Input/Output). *See also* files; pickles**

- definition of, 10, 109, 616
- file exceptions, 113
- manipulating files, 119–121
- reading files, 111–112
- writing files, 110–111

## **IOError, 113**

## **IP address**

- definition of, 309–310, 616
- for localhost, 309, 613

## **IP (Internet Protocol)**

- addresses, 309–310
- definition of, 616
- ports, 310–311

## **IRC (Internet Relay Chat), 340, 616**

## **`isdir` function, `os.path` module, 118**

## **`isfile` function, `os.path` module, 118**

## **ISO 9000, 434**

## **ISO 9001, 434**

## **iterator, 239, 616**

## **J**

## **j (imaginary number indicator), 15, 416**

## **Java language. *See also* Jython**

- classes, calling from Jython, 548–550
- classes, embedding Jython interpreter in, 566–568
- compared to Python, 539
- compiling Jython code into, 568–569
- integrating Jython into, 547–552

**Java Virtual Machine (JVM), 539, 616**`javadom` **Java DOM implementation, 285****JDBC APIs, 552–553****jEdit text editor, 564****J2EE (Java 2 Enterprise Edition), 616****J2EE servlets, writing, 558–564**`join` **function, `os.path` module, 114, 179****joins, SQL, 267–269****JVM (Java Virtual Machine), 539, 616****Jython**

advantages of, 540–541

compared to C Python, 569–570

compiler, 542

compiling into Java, 568–569

database tables, creating, 555–558

databases, accessing, 552–558

definition of, 539–540, 616

executable commands using, 545–546

IDE for, 565

installing, 541–542

interpreter

calling, 546–547

embedding into Java classes, 566–568

integrating into Java applications, 547–552

running, 542

Java classes, calling, 548–550

jEdit text editor for, 564

J2EE servlets, writing, 558–564

packaging applications based on, 547

platforms supported by, 541

resources for, 607

running interactively, 542–543

running Python scripts using, 543–544

Swing API, using with, 550–552

testing from, 565–566

user interface, creating, 550–552

**jython script**

passing parameters to, 544–545

running, 542

**jythonc script**

definition of, 542

running, 568–569

`jython.jar` **file, 547****K****KDE builder, 224****KeyError, 55, 56****keys, in dictionaries, 35****L**`lambda` **statement, 127–128****LAME project**

definition of, 364

extension module for, 368–380

using, 364–367

**languages, programming. See also C language;****Java language; Python**

compared to protocols, 307–308

interpreted, 2

**Large Object (LOB), 430****layers (levels of abstraction), 308–309****LDAP (Lightweight Directory Access Protocol),****432, 453. See also python-ldap module**`ldapadd` **utility, 450****left angle bracket (<)**

&lt; (less than operator), 45–46

&lt;= (less than or equal operator), 47

`len` **function**

for arrays, 34

for strings, 112

for tuples, 31

`__len__` **method, 80–81****less than operator (<), 45–46****less than or equal operator (<=), 47****levels of abstraction (layers), 308–309**`Lib` **directory, Jython, 547**`libglade` **library, 223, 225, 231**`libgmail` **project, 331****licensing, 387–389****Lightweight Directory Access Protocol (LDAP),****432, 453. See also python-ldap module****line numbers, displayed in codeEditor, 3–4****list comprehension, 130**`listdir` **function, `os` module, 118–119, 179, 181**

## lists

- adding elements to, 34
- appending, 40
- for arrays, when to use, 419
- changing elements of, 33
- creating, 33
- creating with ranges, 131–133
- decisions within (list comprehension), 130
- definition of, 33, 616
- equality of, 44
- last elements of, referencing, 38–39
- popping elements from, 40–41
- ranges of, referencing (slicing), 39–40
- reducing (running function on all elements of), 128–129
- treating strings as, 36–37
- visiting elements in without loops, 129–130

## literal numbers, 406, 407

`LiveHTTPHeader` extension, 466

`load` function, `pickle` module, 124

## LOB (Large Object), 430

## local mail spool, 323–325

## localhost

- definition of, 616
- IP address for, 309, 613

## log files

- clipping (filtering), 177
- rotating old versions of, 120–121

`log` function, 415

`log10` function, 415

## logarithm functions, 415

## longs (long integers)

- definition of, 14, 406–407
- integers automatically converted to if too large, 609
- internal representation of, 407

## loops (repetitions)

- breaking out of, 52–54
- continuing, 54
- creating, 51–52
- definition of, 617
- `else` statement in, 54
- infinite loops, 52–53, 616

# M

**mail.** See **e-mail**

**mail spool, local, 323–325**

**Mail To The Future web service, 537**

**mailbox module, 323–325**

`__main__` name, 106

`makedirs` function, `os` module, 121

`map` function, 129–130

`marshal` library, 285

**marshalling, 399**

`match` function, `re` module, 186

**math.** See **arithmetic**

**math functions, 414–415**

`max` function, 414

**mbox format, 324**

**Meerkat API example, 509–511**

**Meerkat web service, 537**

**metadata, 430**

## methods

- definition of, 79–80, 617
- documenting, 156–157
- interface methods, 83–84, 85–87
- internal methods, 84–85
- private methods, 84
- using, 87–89

## MIME (Multipurpose Internet Mail Encoding)

- content types, 314–315
- definition of, 617
- encodings, 313–314
- multipart messages, 316–321
- standards for, 313

`min` function, 414

**mirror client example, 336–337**

**mirror server example, 335–336**

`mkdir` function, `os` module, 121

**model, view, controller design, 238**

**modeling of business processes, 427**

## modules

- accessing elements of, 97–98, 99
- creating, 97–99, 150, 165–168
- definition of, 95–96, 143–144, 617
- documentation for, viewing, 157–162
- documenting, 156–157

editing, 98–99  
 example of, 146–149  
 exceptions for, defining, 154  
 importing  
   all elements of, 103, 145, 168  
   into program with module scope, 96–97  
   specific elements of, 145, 611  
   into top-level scope of program, 101, 103–104,  
   168–169  
 installing, 170–173  
 list of elements in, 144  
 list of modules that have been called, 105  
 location of, 97–98, 145–146  
 public items in, defining, 155–156  
 re-importing, 104–106  
 reloading, 145  
 running as a program, 164  
 scope of, named for module, 99  
 testing, 106, 162–163, 164  
 viewing, 146–149

**modulus (remainder) operator (%), 20, 414**

**move function, `shutil` module, 119**

**MP3 example. See LAME project**

**multidimensional arrays, 418**

**multipart message, 316–321, 617**

**multiplication operator (\*), 19, 412**

**Multipurpose Internet Mail Encoding (MIME)**

content types, 314–315  
 definition of, 617  
 encodings, 313–314  
 multipart messages, 316–321  
 standards for, 313

**multithreaded servers, 339–340**

**multithreading. See threads**

**mutex, 218**

**MySQL**

document retention using, 446–448  
 modules for, 263  
 resources for, 607  
 storing data in, 439–446

## N

**\n, newline character, 7–8**

**\_\_name\_\_ name, 106**

**names. See variables**

**namespaces, XML, 276–277**

**NDA, 389**

**negative numbers, 24, 609**

**network layer, 309**

**network programming**

e-mail

attachments, 315–316  
 MIME messages, 313–321  
 retrieving, 323–331  
 sending, 305–307, 311–312  
 SMTP and, 321–323

peer-to-peer architecture, 354

protocol design, 350

protocols

compared to programming languages, 307–308  
 definition of, 307  
 Internet Protocol (IP), 309–311  
 Internet protocol stack, 308–309, 618

socket programming

binding to external hostname, 334–335

binding to hostname, 332

definition of, 331–332

example of, 332–334

firewalls and, 335

mirror client example, 336–337

mirror server example, 335–336

multithreaded servers, 339–340

Python chat client, 346–350

Python chat server, 340–346

SocketServer module, 337–339

Twisted framework, 351–353

**newline character (\n)**

definition of, 7

entering in strings, 7–8

**nickname, IRC, 340**

**node test, XPath, 282**

**None value, 38, 611**

**normcase function, `os.path` module, 180**

**normpath function, `os.path` module, 116**

**not operator, 47**

**NotSupportedError, DB API, 273**

**numarray package, 422–424**

**numbers**

arithmetic operators for, 412–414

arithmetic, performing, 19–21

### numbers (continued)

- complex numbers, 416–418
- converting characters to, 410–411
- displaying, 17–19, 21, 22
- formatting, 408–410
- formatting in base 8 or base 16, 24
- math functions for, 414–415
- math module for, 415–416
- negative numbers, 24, 609
- type of, determining, 14
- types of, 13–14, 405–408

### numerical analysis, 405

### numerical programming. *See also* numbers

- arrays for, 418–424
- definition of, 405

## O

### **%o** format specifier, 24, 409

### **%#o** format specifier, 409

### object-oriented programming (OOP)

- definition of, 2, 151
- uses of, 92

### objects. *See also* classes

- creating from classes, 82
- definition of, 79–81, 617
- listing all elements of, 80
- pickling, 123–125
- Python, using from C code, 380–382
- scope of, 89–92
- uses of, 81

### octal (base 8) notation

- definition of, 24, 617
- formatting numbers as, 409

### **1** (True value), 36, 38

### **127.0.0.1**, IP address for localhost, 613

### OOP (object-oriented programming)

- definition of, 2, 151
- uses of, 92

### open function, 112

### OpenLDAP project, 432, 448–449

### operating systems

- directory separator characters specific to, 178
- porting to other operating systems, 398–399

### OperationalError, DB API, 273

### operators

- arithmetic, 19–20, 412–414
- assignment operator, 28
- for combining comparisons, 48
- comparison operators, 43–47

### or operator, 48

### ord function, 411

### order of evaluation, for arithmetic, 21–22

### os module

- definition of, 113
- errors raised by, 114
- text processing functions, 178–184

### OSError, 114

### os.listdir module, 116–117

### os.path module, 114–116

### output, 10. *See also* Input/Output (I/O)

### overloading, 20

## P

### packages

- creating, 101–103
- definition of, 95, 101–102, 617
- importing all elements of, 103–104
- re-importing, 104–106
- testing, 106

### parameters

- command-line parameters, 99–100, 134–137
- for functions, 66–71
- passing from Python to C, 360–363
- passing to `python` script, 544–545

### paramstyle global, DB API, 272

### parentheses (())

- enclosing list of DTD elements, 279
- enclosing tuples, 30

### pass statement, 56–57

### passwords, encrypting, 140–141, 331, 392

### path variable, sys module, 97–98, 145–146, 170–171

### PATH\_INFO variable, 472–473

### paths. *See also* directories

- assembling directory names into, 114
- assigning to strings, 109–110
- converting relative to absolute, 116
- determining whether file or directory, 118

- normalizing, 116
- normalizing case of, 180
- splitting extension from end of, 115–116
- splitting into directories, 114–115, 179
- pattern matching. See globbing**
- peer-to-peer architecture, 354, 617**
- penetration tests, 389**
- percent sign (%)**
  - % (escaping in strings), 18–19
  - % (preceding format specifier), 9
  - % (remainder operator), 20, 414
  - % (string formatting operator), 408
  - %d (format specifier), 18, 408
  - %E (format specifier), 18
  - %f (format specifier), 18, 22, 408
  - %o (format specifier), 24, 409
  - %#o (format specifier), 409
  - %s (format specifier), 9
  - %x (format specifier), 24, 409
  - %X (format specifier), 24
  - %#x (format specifier), 409
- per-company licensing, 389**
- performance**
  - assertions and, 193
  - of C code compared to Python code, 355
  - improving, 355
  - of loops, 400
  - of Twisted framework, 353
- period (.), regular expression wildcard, 185**
- physical layer, 308**
- pi constant, 415**
- pickle module, 123, 399**
- pickles, 123–125**
- piracy, 386–387**
- plus sign (+)**
  - + (addition operator), 19, 412
  - + (combining strings), 8
  - + (element grouping operator in DTD), 279
- polymorphism, 151, 617**
- pop method, list, 40–41**
- POP (Post Office Protocol), 618**
- POP3 server**
  - retrieving e-mail from, 325–327
  - secure, 331
- Popen class, 610**
- poplib module, 325–327**
- ports**
  - definition of, 310–311, 618
  - ephemeral port number, 334
  - well-known port, 311, 620
- POST command, 462, 473, 508, 511–513**
- Post Office Protocol (POP), 618**
- pound sign (#)**
  - # (preceding comments), 65–66
  - #! (shebang comment line), 60, 545
- predicates, XPath, 282**
- pricing strategies, 389–395**
- primary key, in relational database, 256**
- print function, 10, 21**
- print statement, 111**
- print\_line\_lengths method, file object, 112**
- private methods, 84**
- processes. See also subprocess**
  - multiple, using, 137–139
  - threading, 139–140
- programmers, Python, finding, 396–397**
- programming**
  - changing requirements of, 2–3
  - consistency in, 2
  - control in, 2
  - object-oriented programming, 2
  - principles of, 1–3
- programming languages. See also C language; Java language; Python**
  - compared to protocols, 307–308
  - interpreted, 2
- ProgrammingError, DB API, 273**
- programs**
  - blocks of, 2, 49
  - context (environment) of, 4
  - current scope of, determining, 106
  - running, 61
  - saving in a file, 59–61
  - as source code, 3
  - top-level scope of, name for, 106
  - writing in codeEditor, 3–5
- protocol stack, 308–309, 618**

## protocols

- compared to programming languages, 307–308
- definition of, 307, 618
- design considerations, 350
- Internet Protocol (IP), 309–311
- Internet protocol stack, 308–309, 618
- terse, 350

`pullDOM` **DOM implementation, 285**

**PUT command, 462, 473**

`.py` **files, 60, 146**

`Py_BuildValue` **function, 363–364**

`.pyc` **files, 104**

`pydoc` **command, 173**

`pyglade` **module, 606**

`pygtk` **module**

- definition of, 402
- importing, 211–212
- resources for, 606

## pyGTK toolkit

- creating widgets, 213–214
- definition of, 210
- FAQ for, 213
- Glade GUI construction kit, 223–224
- installing, 211
- mailing list for, 213
- resources for, 213
- running programs, 213–214
- testing installation of, 211–212
- tutorials for, 213
- widgets
  - creating, 213–214
  - definition of, 213
  - handling signals, 214–215, 216–221
  - multiple, in one window, 222–223
  - `show` method for, 214
  - signals generated by, 214

`PyMethodDef` **structure, 357**

`PyMODINIT_FUNC` **function, 358**

**PyPy project, 308**

**pyQT toolkit, 209–210, 224**

**PyRAP example, 231–238, 241–248**

**Py\_RETURN\_NONE macro, 356**

`PySAX` **libraries, 285**

**PyServlet class, 560–561**

## Python. See also Jython

- compared to Java, 539
- porting to other operating systems, 398–399
- porting to other versions of, 397–398
- problems with, 397–400
- reasons to use
  - for commercial and shareware programs, 385
  - for enterprise applications, 427
  - for extension programming, 355
  - for GUIs, 209–210
  - for network programming, 305
  - for text processing, 175–176
  - for web applications, 459–460
- resources for, 605–606
- version 2.4, new features in, 609–611

**Python chat client, 346–350**

**Python chat server, 340–346**

`python` **command**

- `h` option, 135
- `i` option, 61, 100
- running, 4

## Python shell

- definition of, 4
- errors, receiving, 23–24
- `>>` prompt in, 5
- starting, 4–5
- using with `codeEditor`, 16–17

## PythonCard

- `codeEditor` in, 3
- definition of, 3, 225
- downloading, 3
- resources for, 606

`Python.h` **file, 356**

`PythonInterpreter` **class, 566**

`python-ldap` **module**

- defining and accessing data, 449–450
- definition of, 448–449
- searching data, 451–453

`PYTHONPATH` **variable, 360**

`PyUnit` **facility, 191, 607. See also unittest module**

`pywftk` **module, 607**

**PyXML package, 285, 290**

**Q****Query, Update, Insert, Delete (QUID), 257****QUERY\_STRING variable, 472–473****question mark (?)**

? (element grouping operator in DTD), 279

? (wildcard in glob pattern), 122

**queue of threads, 216–221****QUID (Query, Update, Insert, Delete), 257****quoted-printable encoding, 313–314, 618****quotes**

double quotes (“...”), enclosing strings, 6, 618

single quote (‘...’), element grouping operator in DTD, 279

single quotes (‘...’), enclosing strings, 6, 618

triple quotes (“...” or “...”...”)

enclosing documentation, 156–157

enclosing multi-line strings, 7–8, 618

**R****“r”, preceding regular expressions, 185****raise statement, 73–74****range, 618****range function, 131–132****raw strings, in regular expressions, 185****RDBMS (Relational Database Management System), 618. See also relational databases****RDF Site Summary. See RSS****re module, 186–188****read method, file object, 111****readline method, file object, 111****readlines method, file object, 112****real attribute, complex object, 417****Really Simple Syndication. See RSS (Rich or RDF Site Summary)****recursive functions, 115****reduce function, 128–129****regular expressions**

definition of, 176, 184–185

examples of, 186–187

searching for files using, 187–188

uses of, 176

**Relational Database Management System****(RDBMS), 618. See also relational databases****relational databases**

connecting to database, 260–261

defining tables, 259–260

definition of, 255–257, 618

deleting data, 257–259

for document management, 429–431, 441–448

inserting new data, 257–259

querying database, 257–259

SQL (Structured Query Language), 257–261, 619

updating data, 257–259

as wftk repository, 438–446

when to use, 255

**relative path, converting to absolute, 116****reload function, 105–106, 145****remainder (modulus) operator (%), 20, 414****Remote Procedure Call, XML. See XML-RPC****remove function, os module, 119****repetitions (loops)**

breaking out of, 52–54

continuing, 54

creating, 51–52

definition of, 617

else statement in, 54

infinite loops, 52–53, 616

**repository, wftk**

in database

defining and accessing, 439–441

for document management, 441–448

storing records in, 439–446

definition of, 435

in directory files

defining and accessing, 436–438

storing records in, 438–439

**repr function, 407****representation, REST, 462, 618****REpresentational State Transfer. See REST****request**

HTTP

definition of, 466–467, 615

viewing with web server, 464–466

SOAP, 522–524

XML-RPC, 511–513

**REQUEST\_METHOD variable, 472****reserved words, 29**

## resource identifier

definition of, 618

HTTP, 466–467, 468

REST, 461, 484, 486, 495, 500–501

## resources (information). *See also* web sites

*Beginning XML* (Hunter, et al.), 277

pyGTK toolkit, 213

software, 605–607

web services, publicly available, 536–537

## resources, REST, 460, 461–462, 618

## response

HTTP

definition of, 467–468, 615

viewing, 464–466

SOAP, 524

XML-RPC, 513

## REST (REpresentational State Transfer)

definition of, 460–462, 618

design guidelines for, 468

documentation for, 529

operations of, 462–463

resources for, 460, 461–462

web services using

Amazon Web Services, 495–497

Amazon wish lists, 497–500

BittyWiki example of, 500–503

definition of, 494–495

when to use, 534–535

wiki search and replace using, 503–508

## RESTfulness, 618

`reverse` function, 610

## RFC 1459, 340

## RFC 1521, 313

## RFC 1939, 325

## RFC 2616, 467

## RFC 2822, 311–312, 618

## RFCs, 311

## Rich Site Summary. *See* RSS

## right angle bracket (>)

> (append in `print` statement), 111

> (greater than operator), 45–46

>= (greater than or equal operator), 47

>> (prompt in Python shell), 5

`rmdir` function, `os` module, 121

`rmtree` function, `shutil` module, 122

## robot

definition of, 618

web services as, 493–495

## roles, 432, 433

`rollback` method, `connection` object, 271–272

## rooms, IRC, 340

## rot13 cipher, 411–412

`round` function, 414

## RSS (Rich or RDF Site Summary)

creating RSS aggregator, 301–302

creating RSS feed, 297–301

definition of, 296, 619

DTD for, 297

versions of, 296–297

`runTest` method, `TestCase` class, 193–196

# S

`%s` format specifier, 9

`SafeTemplate` class, 610

Sarbanes-Oxley Act (SOX), 434

## SAX (Simple API for XML)

definition of, 285, 287–288, 619

when to use, 289

## schemas

advantages of, 281

definition of, 277, 278, 280, 620

disadvantages of, 281–282

example of, 280–281

## scientific notation, 17

## scope

definition of, 619

of modules

importing modules into, 96–97

name of, 99

of objects, 89–92

of programs, determining, 106

top-level

importing modules into, 101, 103–104

name for, 106

of variables, 64–65

## scripts

CGI, 469–471

running Python scripts using Jython, 543–544

`search` function, `re` module, 186

search utility examples, 200–207

secure IMAP, 331

secure POP3, 331

Secure Socket Layer (SSL), 331, 619

`select` function, 349–350

`select` module, 349–350

`select` statement, SQL, 257–259

## sequences

accessing in reverse, 610

appending, 40

definition of, 619

equality of, 44

last elements of, referencing, 38–39

lists

adding elements to, 34

appending, 40

for arrays, when to use, 419

changing elements of, 33

creating, 33

creating with ranges, 131–133

decisions within (list comprehension), 130

definition of, 33, 616

equality of, 44

last elements of, referencing, 38–39

popping elements from, 40–41

ranges of, referencing (slicing), 39–40

reducing (running function on all elements of),  
128–129

treating strings as, 36–37

visiting elements in without loops, 129–130

ranges of, referencing (slicing), 39–40

strings as, 38

tuples

appending, 40

for arrays, when to use, 419

creating, 30–31

definition of, 30, 619

equality of, 44

last elements of, referencing, 38–39

layers of, 31–33

one-element tuples, 32

ranges of, referencing (slicing), 39–40

reducing (running function on all elements of),  
128–129

visiting elements in without loops, 129–130

servers, trusted, 350

servlet container, 559

servlets, J2EE, 558–564

setter methods, 152

`setUp` method, `TestCase` class, 196–199

`setup.py` file, 171–172

`sgmlop` C helper module, 285

`sha` module, 140–141

shareware, using Python for, 385

development environment for, 395–396

distribution of, 400–401

how much Python to use for, 386–387

libraries for, 401–402

licensing issues, 387–389

piracy issues, 386–387

as platforms, 395

pricing strategies, 389–395

Python problems concerning, 397–400

Python programmers for, finding, 396–397

reasons for, 385

Web services and, 388–389

sharp sign (#)

# (preceding comments), 65–66

#! (shebang comment line), 60, 545

shebang comment line (#!), 60, 545

shell. See Python shell

`show` method, for widgets, 214

`shutil` module, 119

signals generated by widgets, 214–215

Simple API for XML. See SAX

Simple Mail Transfer Protocol. See SMTP

Simple Object Access Protocol. See SOAP

`SimpleHTTPRequestHandler` object, 464

`SimpleXMLRPCServer` class, 514, 530

`sin` function, 415

single quote ('), element grouping operator in  
DTD, 279

single quotes ('...'), enclosing strings, 6, 618

single-threaded multitasking, 348–350

`sinh` function, 415

site-packages directory, 171

slash (/)

/ (division operator), 19, 412

// (floor division operator), 413

/ (in Unix path names), 178

// (XPath axis shortcut), 282

## **SMTP (Simple Mail Transfer Protocol)**

definition of, 619  
port number for, 310  
sending e-mail with, 321–323  
`smtp`lib module, **306, 321–323**

`.so` files, **356**

## **SOAP (Simple Object Access Protocol)**

BittyWiki example using, 525–527  
definition of, 520, 619  
documentation for, 530  
`Fault` element (errors flagged by), 524–525  
Google API example, 520–522  
request for, 522–524  
response for, 524  
when to use, 534  
wiki search and replace using, 527–529

## **SOAPpy library, 520**

## **socket library, 332**

### **socket programming**

binding to external hostname, 334–335  
binding to hostname, 332  
definition of, 331–332  
example of, 332–334  
firewalls and, 335  
mirror client example, 336–337  
mirror server example, 335–336  
multithreaded servers, 339–340  
Python chat client, 346–350  
Python chat server, 340–346  
`SocketServer` module, 337–339  
`TimeoutSocket` module, 401–402

### **sockets**

binding to hostname, 332  
definition of, 332, 619  
example of, 332–334

`SocketServer` module, **337–339**

software, resources for, **605–607**

`sorted` function, **117**

source code, **3**

## **SOX (Sarbanes-Oxley Act), 434**

### **spaces (whitespace)**

for code blocks, 49  
definition of, 620

`spawn` function, `os` module, **138**

specialization (inheritance), **151, 616**

Speller web service, **537**

## **spider**

definition of, 619  
REST example of, 503–508  
SOAP example of, 527–529  
XML-RPC example of, 518–519

`split` function, `os.path` module, **114–115, 179**

`splittext` function, `os.path` module, **115–116**

## **SQL (Structured Query Language)**

connecting to database, 260–261  
`create table` statement, 259–260  
definition of, 619  
`delete` statement, SQL, 257–259  
`insert` statement, SQL, 257–259  
joins, 267–269  
`select` statement, SQL, 257–259  
`update` statement, SQL, 257–259

`sqrt` function, **415**

### **square brackets ([])**

dereferencing dictionaries, 35  
dereferencing lists, 33  
dereferencing tuples, 31  
enclosing lists, 33  
regular expression wildcard, 187  
wildcard in glob pattern, 122

## **SSL (Secure Socket Layer), 331, 619**

### **stack trace, 75**

`startup` method, `connection` object, **262**

`stat` function, `os` module, **179, 181**

state machine, **455**

stateless servers, REST, **461**

static database cursor, **558**

status code, HTTP, **467–468, 615**

steps, XPath, **282**

`str` function, **18, 407, 408**

stream-based parsing, **288**

`StreamRequestHandler` class, **338**

string formatting operator (%), **408**

string module, **134**

strings. *See also* `docstring`

combining (concatenating), 8  
converting to integers, 406  
definition of, 5, 619  
displaying, 10  
displaying numbers as, 17–19  
escaping characters in, 6–7  
formatting, 9–10

immutable, problems with, 399–400  
 newline characters in, 7–8  
 quotes used to enter, 6–8  
 substitutions using templates, 610  
 substitutions with dictionaries, 133–134  
 treating as a list (slicing), 36–37

**Structured Query Language. See SQL**

**subclass Error, DB API, 273**

**subclasses**

creating, 139, 153  
 inheritance and, 151  
 polymorphism and, 151

**subprocess**

creating, 610  
 multithreaded servers and, 339  
 when to use, 140

`subprocess.call` **function, 393, 610**

**subscription ID, Amazon Web Services, 495**

**subtraction operator (-), 19, 412**

`sum` **function, 414**

**Swing API, using Jython with, 550–552**

`sys` **module, path variable, 97–98, 145–146, 170–171**

`sys.modules` **dictionary, 105**

`sys.modules.keys` **list, 105**

`system` **function, os module, 138–139**

## T

**tables, in relational databases**

creating in Jython, 555–558  
 defining, 259–260  
 definition of, 256

**tags, XML, 275–276**

`tan` **function, 415**

`tanh` **function, 415**

**tarball, 400–401**

**task, in workflow, 432–433**

**TCP (Transport Control Protocol), 309, 619**

**TCP/IP, 309, 619**

`tearDown` **method, TestCase class, 196–199**

**Technorati web service, 537**

**Template class, 610**

**templates**

string substitutions using, 610  
 XSLT, 293–294

**terse protocols, 350**

**test cases, 193–196**

**test fixtures, 196–199**

`test` **function, 163, 164**

**test suites**

creating, 194–196  
 definition of, 193  
 in Extreme Programming, 199–205

`TestCase` **class**

`runTest` **method, 193–196**

`setUp` **method, 196–199**

`tearDown` **method, 196–199**

**testing**

assertions, 191–193  
 in Extreme Programming, 199–205  
 from Jython, 565–566  
 in software life cycle, 207–208  
 test cases, 193–196  
 test fixtures, 196–199

**tests. See comparisons; decisions**

**text processing**

definition of, 175  
 directory navigation for, 176  
 files, searching for, 176–177  
 log files, clipping (filtering), 177  
 mail, searching, 178  
 os module functions for, 178–184  
 regular expressions for, 176, 184–188  
 uses of, 175–178

**ThreadingMixIn class, 339–340**

**threads**

debugging, 399  
 definition of, 139–140  
 handling signals with, 216–221

`Timeoutsocket` **module, 401–402**

**TK GUI toolkit, 209**

**Tomcat server, 559–560**

**transactions, DB API, 271–272**

**Transport Control Protocol (TCP), 309, 619**

**transport layer, 309**

**tree view widget**

creating, 239–241  
 definition of, 238  
 example using, 241–248

## **trigonometric functions, 415**

### **triple quotes (“...” or “...”“”“”)**

- enclosing documentation, 156–157
- enclosing multi-line strings, 7–8, 618

### **True value (1)**

- definition of, 36, 38
- as result of comparisons, 43

### **truncating numbers, 406**

### **trusted servers, 350**

### **try statement, 55–57**

### **tuples**

- appending, 40
- for arrays, when to use, 419
- creating, 30–31
- definition of, 30, 619
- equality of, 44
- last elements of, referencing, 38–39
- layers of, 31–33
- one-element tuples, 32
- ranges of, referencing (slicing), 39–40
- reducing (running function on all elements of), 128–129
- visiting elements in without loops, 129–130

### **Twisted framework, 351–353, 620**

### **200 status code, 467**

### **type function**

- checking parameters using, 68–69
- definition of, 14

### **TypeError, 32, 56, 68**

### **types. See data types**

## **U**

### **UID (Unique ID)**

- definition of, 620
- for IMAP, 330–331

### **underscore (\_), in variable names, 30**

### **Unicode, 620**

### **unittest module, 193**

### **unlink function, os module, 120**

### **unsubscribe, 68**

### **update statement, SQL, 257–259**

### **URL, 461**

### **urllib module, 496**

### **user agent, 470, 473, 500, 620**

### **user interface, creating from Jython, 550–552.**

*See also* GUIs

### **users, managing, 431–432**

## **V**

### **validation of XML, 285–287**

### **variables**

- anonymous, 613
- assigning values to, 28
- changing values of, 29
- characters that can't be used in, 30
- conflicting with names in other contexts, 169–170
- copying, 29
- definition of, 27–28, 620
- scope of, 64–65
- words that can't be used as, 29

### **verb, HTTP, 462–463, 473, 615**

### **version control system, 430**

### **versions of Python**

- porting to other versions, 397–398
- version 2.4, new features in, 609–611

### **vertical bar (|)**

- | (OR between DTD elements), 279
- | (preceding optional arguments), 362

### **visible web server, 464–466**

## **W**

### **wait function, os module, 137**

### **walk function, os.path module, 180, 181, 182–183**

### **Warning error, DB API, 273**

### **watermarking, 390–394**

### **W3C (World Wide Web Consortium), 278**

### **web applications. See also web services**

- advantages of, 459
- CGI (Common Gateway Interface)
  - definition of, 468–469, 613
  - environment variables for, 471–473
  - HTML forms and, 473–479
  - scripts, running, 469–470
  - web server's role in executing scripts, 470–471
- definition of, 620
- frameworks for, 460

HTTP and, 462–468  
 modules for, 460  
 REST architecture and, 460–462  
 as web services, 536

#### wiki

definition of, 479–481, 620  
 markup used by, 480–481  
 search and replace for, using REST, 503–508  
 search and replace for, using SOAP, 527–529  
 search and replace for, using XML-RPC,  
 518–519  
 storage used by, 481

#### web server

role in executing CGI script, 470–471  
 visible (viewing request and response), 464–466  
 writing, 463

#### web services

definition of, 388–389, 493–494, 620  
 designing, 535–536  
 documenting web service API, 529–534  
 etiquette for, 535–536  
 publicly available, list of, 536–537  
 REST (REpresentational State Transfer)  
 Amazon Web Services, 495–497  
 Amazon wish lists, 497–500  
 BittyWiki example of, 500–503  
 definition of, 494–495  
 when to use, 534–535  
 wiki search and replace using, 503–508  
 SOAP (Simple Object Access Protocol)  
 BittyWiki example using, 525–527  
 definition of, 520, 619  
 documentation for, 530  
 Fault element (errors flagged by), 524–525  
 Google API example, 520–522  
 request for, 522–524  
 response for, 524  
 when to use, 534  
 wiki search and replace using, 527–529  
 standards for, 494  
 standards for, choosing, 534–535  
 web applications as, 536  
 WSDL file defining, 531–534  
 XML-RPC (Remote Procedure Call)  
 BittyWiki example using, 514–517  
 data representation, 512–513

definition of, 508–509, 620  
 documentation for, 529–530  
 fault response (errors flagged by), 513–514  
 introspection API, 530–531  
 Meerkat API example, 509–511  
 POST request for, 508, 511–513  
 response, 513  
 when to use, 534–535  
 wiki search and replace using, 518–519

#### web sites

Amazon.com web service, 537  
 CGI (Common Gateway Interface), 469  
 DB API, 273  
 del.icio.us web service, 537  
 Eclipse IDE, 565  
 Flickr web service, 537  
 floating point numbers, 17, 21  
 4Suite library, 294  
 Gadfly database, 260  
 Glade GUI construction kit, 225  
 Google web service, 520, 522, 537  
 Grinder, 541  
 HSqlDB database, 554  
 jEdit text editor, 564  
 J2EE, 561  
 Jython, 541, 607  
 LAME project, 364  
 libgmail project, 331  
 LiveHTTPHeader extension, 466  
 Mac Python, 606  
 Mail To The Future web service, 537  
 Meerkat web service, 537  
 MySQL, 607  
 open source licensing, 387  
 PyDev, 565  
 pyglade module, 606  
 pygtk module, 606  
 Python, 171, 605–606  
 Python API from C, 356  
 PythonCard, 225, 606  
 PyUnit facility, 607  
 pywftk module, 607  
 PyXML package, 285  
 RedRobin Jython, 565  
 REST, 460  
 RSS (Rich or RDF Site Summary), 297

### web sites (continued)

- SOAP library, 520, 522
- Speller web service, 537
- Technorati web service, 537
- Tomcat server, 559
- Twisted framework, 351, 402
- WebserviceX web service, 537
- Web-Sniffer, 466
- wFtk workflow toolkit, 607
- wiki, 481
- wxPython toolkit, 210
- Xmethods web service, 537
- XML, 277
- XPath, 282
- XSLT namespace, 293
- Yahoo! web service, 537

**webmail, compared to e-mail, 331**

**WebserviceX web service, 537**

**well-formedness of XML, 286, 621**

**well-known port, 311, 620**

wFtk **workflow toolkit**

- action queue handler, 456–457
- definition of, 435–436
- repository, in database
  - defining and accessing, 438–439
  - for document management, 441–448
  - storing records in, 439–446
- repository, in directory files
  - defining and accessing, 436–438
  - storing records in, 438–439
- resources for, 607
- workflow trigger, 453–456

whichdb **module, 251**

while **statement, 51–54**

**whitespace**

- for code blocks, 49
- definition of, 620

**widget packing, 222–223**

**Widget Tree, Glade**

- creating, 239–241
- definition of, 238
- example using, 241–248

**widgets**

- creating, 213–214, 228–230
- definition of, 213
- handling signals, 214–215, 216–221

- initialization required by, 238
- multiple, in one window, 222–223
- show method for, 214
- signals generated by, 214
- tree view widget, 238–241

### wiki

BittyWiki example

- core library for, 481–484
- creating wiki pages from Python session, 483–484
- making executable, 493
- markup used by, 486–492
- resources for, 484–486
- REST API document for, 529
- REST API for, 500–503
- SOAP API document for, 530
- SOAP interface for, 525–527
- web interface for, 484–493
- WSDL proxy for, 533–534
- XML-RPC API document for, 529–530
- XML-RPC interface for, 514–517

definition of, 479–481, 620

markup used by, 480–481

search and replace for

- using REST, 503–508
- using SOAP, 527–529
- using XML-RPC, 518–519

storage used by, 481

### Wikipedia encyclopedia, 479

### WikiWords, 480, 481

### wildcards in glob patterns, 122.

*See also regular expressions*

### WingIDE, 396

### workflow

- action queue handler, 456–457
- definition of, 432–433
- workflow trigger, 453–456

### World Wide Web Consortium (W3C), 278

### write method, file object, 110

### WSDL (Web Services Description Language)

- definition of, 620
- web service interface defined by, 531–534

### wxDesigner, 224

### wxGlade, 224

### wxPython toolkit, 210, 224

**X****%x format specifier, 24, 409****%X format specifier, 24****##x format specifier, 409****XHTML**

definition of, 282

parsing, 283–284

**Xmethods web service, 537****XML (eXtensible Markup Language)**

compared to DTD, 280

created by Glade, 230–231

definition of, 275–277, 620

libraries for, 285

parsing

with DOM, 288–289, 290–292

with SAX, 287–288, 289, 290, 292–293

standards for, 277

transforming with XSLT, 293–296

validating, 285–287, 621

XPath expressions in, 282

**XML schema. See schemas****<?xml ...?> tag, 276, 279****XML validation, 285–287, 621****XML well-formedness, 286, 621**`xml.dom` **DOM processor, 285**`xml.dom.minidom` **DOM implementation, 285, 290–292**`xmlproc` **XML parser, 285, 286–287****XML-RPC (Remote Procedure Call)**

BittyWiki example using, 514–517

data representation, 512–513

definition of, 508–509, 620

documentation for, 529–530

fault response (errors flagged by), 513–514

introspection API, 530–531

Meerkat API example, 509–511

POST request for, 508, 511–513

response, 513

when to use, 534–535

wiki search and replace using, 518–519

`xml.sax` **package, 290, 292–293****XP (Extreme Programming), test suites for, 199–205****XPath, 277, 282****Xrange, 621**`xrange` **object**

accessing in reverse, 610

definition of, 132–133

**XSL-FO (Extensible Style Language Formatting Objects), 277, 621****XSLT (Extensible Style Language for Transformations)**

definition of, 277, 293–294, 621

transforming XML with, 294–296

**Y****Yahoo! web service, 537****Z****Zawinski, Jamie (quote about e-mail), 305****0 (False value), 36, 38****Ox, preceding hexadecimal literals, 406****zxJDBC package, 553**

















