INDEX

Numbers and Symbols

256 objects and 257 objects, 154–155
`./`, using with Ubuntu, 42
`/?` command line argument, 25–26
`=` assignment operator, 113
`==` comparison operator, 113, 336
    chaining, 103, 105
    using to compare objects, 154
    using with `None`, 94–95
`!=` comparison operator, 336
* and ** syntax
    using to create variadic functions, 167–171
    using to create wrapper functions, 171–172
    using with arguments and functions, 166–167
* character, using as wildcard, 28–29
? character, using as wildcard, 28–29
`[:]` syntax, using, 104
`<` comparison operator, 337
`<=` comparison operator, 337
`>` comparison operator, 337
`>=` comparison operator, 337
`->` (arrow), using with type hints, 191
\ (backslash)
    purpose of, 18
    using with strings, 95
`:` (colon), using with lists, 97–98, 104
`,` (comma), including in single-item tuples, 150
`-` (dash), using with command line arguments, 25
$ (dollar sign), using in macOS, 23
`.` (dot), using with commands, 31
`--` (double dash), using with command line arguments, 25
__(double underscore), using in dunder methods, 322. *See also* underscore (_)
/ (forward slash)
    purpose of, 18
    using with command line arguments, 25
(hash mark)
    using with comments, 183
    using with docstrings, 188
`[]` (index operator), using, 117
`;` (semicolons), using with `timeit` module, 226–227
' (single quote), using, 46
~ (tilde), using in macOS, 23
`-` (unary operator), 155–157
`+` (unary operator), 156–157
_ (underscore)
    PEP 8's naming conventions, 60–61
    as prefix for methods and attributes, 291–292
    private prefix, 81
    using with `_spaces` attribute, 290
    using with dunder methods, 120
    using with private attributes and methods, 283
    using with `WizCoin` class, 279

A

abcTraceback.py program, saving, 4
`__abs__()` numeric dunder method, 328
absolute versus relative paths, 20–21
`__add__()` numeric dunder method, 327
`addToTotal()` function, creating, 172–173

algebra for big O, 236
algorithm analysis. *See* big O algorithm
aliases, defined, 121
`all()` function, 157–158
alphabetical sort, performing, 146–147
ALT key. *See* keyboard shortcuts
Amdahl's Law, 229–230
`and` operator, using, 103
`__and__()` numeric dunder method, 328
anonymous functions, 174–175
answer archive, building with Stack Overflow, 12
antigravity feature, enabling, 160
API (application programming interface), 130
API vs. library vs. framework vs. SDK vs. engine, 130
`append()` method, using, 115
applications, opening, 22
arguments. *See also* positional arguments
  vs. parameters, 128
  passing to functions, 166–167
  setting defaults for parameters, 142–143
arrow (`->`), using with type hints, 191
ASCII (American Standard Code for Information Interchange), 146
ASCII art, using in Four-in-a-Row game, 272
`AsciiArt` class, creating, 304–305
assignment and comparison operators, chaining, 103, 105
atoms and literals, 110
attributes
  as backing fields, 317–318
  defined, 278, 282–284
  vs. properties, 128–129
  turning into properties, 316–319
  vs. variable, 124

B

backing field, relationship to attributes, 317–318
backporting type hints, 196
backslash (\)
  purpose of, 18
  using with strings, 95
base class, relationship to inheritance, 296
Bash file, 22–23
BFG Repo-Cleaner tool, 220
big O algorithm
  algebra, 236
  algorithm analysis, 230
  analysis, 230
  analysis examples, 239–242
  analyzing, 243–244
  best practice, 244
  bookshelf metaphor for orders, 231–235
  determining order for code, 237–244
  doing analysis, 236
  function calls, 242–243
  lower and higher orders, 230–231
  lower orders and coefficients, 238–239
  math for analysis, 236
  measuring worst-case scenario, 235
  "n" is small, 244
  order of function calls, 242–243
  orders of notation, 245
big Omega notation, 235
big Theta notation, 235
binary code, explained, 129
Black code formatting tool
  adjusting line length setting, 55
  described, 45
  disabling double-quoted strings setting, 55–56
  disabling for parts of code, 57
  installing, 54
  previewing changes, 56–57
  running from command line, 54–57
  semantic decisions, 58
  syntactic decisions, 58
block vs. clause vs. body, 123–124
`__bool__()` numeric dunder method, 328
Boolean values, 158–159. *See also* values
Bourne Shells, 22–23
bugs, types of, 109, 150–151
bytecode vs. machine code, 129

C

C:\ part of path, 18
-c switch, using to run code from command line, 26
callables and first-class objects, 121–122
camelCase, 60
casing styles, 60
Catalina version, 23
cd (change directories) command, 29–30
__ceil__() numeric dunder method, 328
chaining operators, 103, 105, 159–160
child class, creating, 294
class attributes, 306
class methods, 304–306
class objects, 284
classes. *See also* inheritance
  creating, 77
  creating objects from, 278
  creating WizCoin, 279–284
  defined, 276
  designing for real world, 290–291
  as functions or modules, 77
  "is a" relationships, 299
clause vs. block vs. body, 123–124
CLI (command line interface), 22
close() and open() functions, 93–94
cls and clear (clear terminal) commands, 35
code. *See also* source code
  avoiding guesses, 90
  beauty of, 88
  commented out, 74–75
  flat vs. nested, 89
  formatting for readability, 12–13
  implementation, 90
  interrupting, 134
  namespaces, 91
  organizing, 77
  readability of, 89
  running from command line, 26
  silenced errors, 89–90
  simplicity and complexity of, 89
  sparse vs. dense, 89
  special cases, 89
  speed of, 90
  verbose and explicit, 89
code formatting, defined, 45
code point, getting for characters, 146–147
code smells. *See also* source code
  classes as functions or modules, 77
  commented-out code, 74–75
  dead code, 74–75
  defined, 69
  duplicate code, 70–71
  empty except blocks, 79–80
  error messages, 79–80
  list comprehensions, 77–79
  magic numbers, 71–73
  myths, 80–84
  print debugging, 75–76
  summary, 84–85
  variables with numeric suffixes, 76
codetags and TODO comments, 187
coercion, explained, 128
collections module, contents of, 120
collections.defaultdict, using for default values, 99–100
colon (:), using with lists, 97–98, 104
comma (,), including in single-item tuples, 150
command history, viewing, 28
command line
  arguments, 24–26
  options, 25
  overview, 22–23, 42
  running code with -c switch, 26
  running programs from, 23–24
  running *py.exe* program, 26–27
  running Python programs from, 26
  tab completion, 27–28
  terminal window, 23
Command Prompt shell, 23
commands
  canceling, 28
  cd (change directories), 29–30
  cls and clear (clear terminal), 35
  copy and cp (copy files), 31–32
  del (delete files and folders), 33–34
  dir (list folder contents), 30
  dir /s (list subfolder contents), 31
  find (list subfolder contents), 31
  ls (list folder contents), 30

commands *(continued)*
md and mkdir (make folders), 34
move and mv (move files), 32
mv (rename files and folders), 32–33
rd and rmdir (delete folders), 34–35
ren (rename files and folders), 32–33
rm (delete files and folders), 33–34
running from Python programs, 27
shortened names, 32
where (find programs), 35
which (find programs), 35
wildcard characters, 28–29
commented-out code, 74–75
comments
best practices, 197
myth about, 83–84
using, 182–188
using with type hints, 196
commit history, rewriting in Git, 220
commit log, viewing in Git, 216–217
commits, rolling back in Git, 218–220
comparing objects, 154–155
comparison operators. *See also* sequence comparisons
chaining with assignment operators, 103, 105
function form of, 333
comparisons, making, 94–95
__complex__() numeric dunder method, 328
composition vs. inheritance, 299–301
conditional expressions, ternary operator, 101–102
containers, defined, 119–120
Cookiecutter, using to create projects, 200–202
copy and cp commands, 31
copy.copy() and copy.deepcopy(), using with mutable values, 140–142
copying
files and folders, 31
mutable values, 140–142
cProfile profiler, 154, 228–230
CPU, instruction set of, 129
CPython implementation, 108
CTRL key. *See* keyboard shortcuts
cwd (current working directory), 19–20, 31

D

dash (-) using with command line arguments, 25
data, validating with setters, 319
data types
defined, 276
and return values, 177–178
dead code, 74–75
decimal.Decimal(), passing integers to, 148–149
decrement and increment operators, 156–157
default arguments, setting for parameters, 142–143, 165–166
default values, using collections.defaultdict for, 99–100
deleting
files and folders, 33–34
files from repo, 214–215
folders, 34–35
items from list, 134–140
and moving files in repo, 215–216
derived class, relationship to inheritance, 296
deterministic function, 173. *See also* functions
dictionaries
get() and setdefault() methods, 104
key-value pairs, 118
Python mailing list discussion, 131
setting type hints for, 195–196
using, 98–101
using as default arguments, 143–144
diff program, seeing changes with, 211–212
dir command, using, 30
dir /s command, 31
displays and literals, 110
__divmod__() numeric dunder method, 327
docs folder, contents of, 200

docstrings
defined, 182
summary, 197
using, 188–190
dollar sign ($), using in macOS, 23
doskey program, 28
dot (.), using with commands, 31
double dash (--), using with command line arguments, 25
double-free bugs, 109
double underscore (__), using in dunder methods, 322. *See also* underscore (_)
dunder methods. *See also* methods; OOP (object-oriented programming)
comparison, 332–337
defined, 120
numeric, 325–328
in-place augmented assignment, 330–332
reflected numeric, 328–330
string representation, 323–325
using, 322–323
duplicate code, 70–71
dynamic typing, 190

E

Easter egg, *The Zen of Python* as, 88
encapsulation, explained, 307–308
encoding definition, using with magic comment, 187–188
engine vs. library vs. framework vs. SDK vs. API, 130
`enumerate()` vs. `range()`, 92–93, 104
environment setup, process of, 17, 42
environment variables and PATH, 35–39
`__eq__()` comparison dunder method, 336
EQual operation, 336
equality (==) operator, using with None, 94–95
error codes, returning, 178–179
error messages
and `except` blocks, 79–80
getting help with, 11
parsing, 15
tracing, 178
understanding, 4–8
errors, preventing with linters, 8–9
exceptions
catching, 79–80
raising, 4, 90, 178–179
`RecursionError`, 318–319
explanatory comments, 184–185
expressions vs. statements, 122–123

F

`False` and `True` keywords, 158–159
FAQs (Frequently Asked Questions), 10
filenames, as command line arguments, 25
filenames and folders, matching with wildcards, 28–29
file paths, specifying, 20–21
files
copying, 31
deleting, 33–34
moving, 32
renaming, 32–33
filesystem, 18–21, 42
filtering with list comprehensions, 175–176
`find` command, 31
find feature, accessing, 64, 67
`find()` string method, error related to, 178
finding programs, 35
first-class objects and callables, 121–122. *See also* objects
flag arguments, myth about, 82
`__float__()` numeric dunder method, 328
floating-point numbers, accuracy of, 147–149, 151
`__floor__()` numeric dunder method, 328
`__floordiv__()` numeric dunder method, 327
folders
adding to `PATH` on macOS and Linux, 39
adding to `PATH` on Windows, 38–39
as command line arguments, 25
copying, 31
deleting, 33–34
in filesystem, 18

folders *(continued)*
home directory, 19
listing contents of, 30
making, 34
moving, 32
renaming, 32–33
folders and filenames, matching with wildcards, 28–29
`for` expressions, including in list comprehension, 79
`for` loops
in big O analysis, 240
getting index and value, 104
and lists, 134–140
versatility of, 125
form, filling out, 276–278
`format()` string method, 96–97
formatting for readability, 58
formatting strings, 96–97
forward slash (`/`)
purpose of, 18
using with command line arguments, 25
Four-in-a-Row tile-dropping game
output, 259–260
source code, 260–264
summary, 271–272
writing code, 264–271
frame object, explained, 5
frame summary, explained, 5
framework vs. library vs. SDK vs. engine vs. API, 130
f-strings, formatting strings with, 96–97
functional programming
higher-order functions, 174
lambda functions, 174–175
list comprehensions, 175–176
mapping and filtering, 175–176
side effects, 172–174
function calls, order in big O, 242–243
functions. *See also* deterministic function; higher-order functions; nondeterministic function; pure function; variadic functions; wrapper functions
default arguments, 165–166
vs. methods, 82, 124
names, 162
parameters and arguments, 165–172
passing arguments to, 166–167
and return statements, 80–81
size trade-offs, 162–165
and `try` statements, 81–82
using default arguments with, 142–143

G

garbage collection, 109, 226
`__ge__()` comparison dunder method, 337
`get()`, using with dictionaries, 98–101
`getPlayerMove()` function, 163–165
getters and setters, 315, 318
Git. *See also* repo
adding files to track, 208–209
command line tool, 207
commits and repos, 200, 206–207
committed state, 204
committing changes, 210–214
configuring username and email, 203
deleting files from repo, 214–215
as distributed version control system, 206
frequency of committing changes, 213–214
ignoring files in repo, 209–210
installing, 202–204
keeping track of file status, 204–206
`log` command, 216–217
modified state, 204
recovering old changes, 217–220
renaming and moving files in repo, 215–216
rewriting commit history, 220
running status with `watch` command, 207
staged state, 205–206
storing private information in, 220
viewing changes before committing, 211–212
viewing commit log, 216–217
workflow, 204–206

`git add` command, 223
`git clone` command, 223
`git commit` command, 223
`git diff`, using, 211–213
`git filter-branch` command, 220
`git init` command, 223
GitHub and `git push` command, 221–223
glob patterns, explained, 29
global variables, myth about, 82–83
glossary, accessing, 108, 131
`GrandchildClass`, creating, 294–295
Greater Than operation, 337
Greater than or Equal operation, 337
`__gt__()` comparison dunder method, 337
GUI (graphical user interface), 22
GUI Git tools, installing, 203–204

H

hash mark (#)
  using with comments, 183
  using with docstrings, 188
hashes, defined, 117–119
`--help` command line argument, 25–26
help with programming, asking for, 9–14
higher-order functions, 174. *See also* functions
home directory, 19
Homebrew, installing and configuring, 213
horizontal spacing, 47–51
Hungarian notation, 63

I

`id()` function, calling, 111, 154
identifiers, defined, 59
identities, defined, 111–114
IEEE 754 standard and floating-point numbers, 147–148
`if` statement as clause header, 124
immutable and mutable objects, 114–117, 144
`in` operator, using with values of variables, 105
increment and decrement operators, 156–157
indentation, using space characters for, 47–48. *See also* significant indentation
index operator ([]), using, 117–118
`index()` string method, exception related to, 179
indexes, defined, 117–119
inequality `!=` operators, avoiding chaining, 149–150
inheritance. *See also* classes; multiple inheritance; OOP (object-oriented programming)
  base classes, 296
  best practice, 308–309
  class attributes, 306–307
  class methods, 304–306
  vs. composition, 299–301
  creating child classes, 294
  derived classes, 296
  downside, 301–303
  explained, 293
  `isinstance()` and `issubclass()` functions, 303–304
  MRO (method resolution order), 310–312
  overriding methods, 296–297
  static methods, 306–307
  subclasses, 296
  super classes, 296
  `super()` function, 297–299
`__init__()`, and self, 280–282
__init__.py file and packages, 121
inline comments, 183–184
in-place augmented assignment dunder methods, 330–332
installing
  Black code formatting tool, 54
  Git, 202–204
  Homebrew, 213
  Meld for Linux, 213
  Mypy, 192–193
  Pyflakes, 9
  tkdiff, 213
instances, defined, 111–114, 276. *See also* `isinstance()`
instruction set, explained, 129
`int()` function, using, 158

__int__() numeric dunder method, 328
integers, passing to decimal.Decimal(), 148–149.
__invert__() numeric dunder method, 328
"is a" relationships for classes, 299
is operator, using, 113
isinstance(). *See also* instances
  and issubclass() functions, 303–304, 312
  using with Boolean values, 158
items
  best practices for dealing with, 134–140
  defined, 114
iterable unpacking, using to swap variables, 227
iterable vs. iterator, 125–126
iterating
  explained, 134
  forward and backward, 139

J

JDK (Java Development Kit), 130
join() operator, using with strings, 151
JVM (Java Virtual Machine), 129

K

keyboard shortcuts
  canceling commands, 28
  find feature, 64, 67
  interrupting code, 134
  interrupting infinite loops, 134
  opening applications, 22
  opening terminal window, 23, 41
  Task Manager, 22
  viewing running processes, 22
keys, defined, 117–119
keywords
  arguments, 167
  defined, 110–111
  True and False, 158–159
Kompare, using, 213

L

lambda functions, 174–175. *See also* functions
__le__() comparison dunder method, 337
legal comments, 186
len() function, using, 92
Less Than operation, 337
Less than or Equal operation, 337
"lessons learned" comments, 185–186
library vs. framework vs. SDK vs. engine vs. API, 130. *See also Python Standard Library*
LICENSE.txt file, 200
links to URLs, including in comments, 183
linters, preventing errors with, 8–9, 15
Linux
  installing Meld for, 213
  running Python programs on, 41
list comprehensions
  and all() function, 157
  mapping and filtering with, 175–176
  using, 77–79, 137
list concatenation, 115
lists
  adding or deleting items from, 134–140
  best practices for dealing with, 134–140
  contents of, 141–142
  making shallow copies of, 97–98
  setting type hints for, 195–196
  using as default arguments, 143–144
literals, defined, 109–110
logfiles, setting up, 75–76
logical error, defined, 127
looping, explained, 134
loops
  interrupting, 134
  moving duplicate code into, 71
ls command, using, 30
__lshift__() numeric dunder method, 328
__lt__() comparison dunder method, 337

M

machine code vs. bytecode, 129

macOS
installing tkdiff, 213
running Python programs on, 41
magic comments and source file encoding, 187–188
magic methods, defined, 120
magic numbers, 71–73
`main()` function, changing to override methods, 296–297
mapping. *See also* objects
defined, 119–120
and filtering with list comprehensions, 175–176
mapping data types, passing, 167
math dunder methods, 325–328
`__matmul__()` numeric dunder method, 327
`max()` and `min()` functions, 169
MCR (minimum, complete, reproducible) example, 11
`md` and `mkdir` commands, 34
Meld, installing for Linux, 213
memory leaks, 109
memory usage, 137–138
metasyntactic variables, 60. *See also* variables
methods. *See also* dunder methods; private attributes and private methods
vs. functions, 82, 124
`__init__()`, and self, 280–282
overriding, 296–297
`min()` and `max()` functions, 169
`__mod__()` numeric dunder method, 327
modules
defined, 120–121
finding, 14
and packages, 120–121
requests, 188–189
typing, 195–196
`move` and `mv` (move files) commands, 32
moving files and folders, 32
MRO (method resolution order), 310–312
`__mul__()` numeric dunder method, 327
multiple assignment trick, using to swap variables, 227
multiple inheritance, 309–310. *See also* inheritance
mutable and immutable objects, 114–117, 151
mutable values
best practices for dealing with, 142–144
copying, 140–144
and default arguments, 142–144
Mypy, using, 192–194

N

name length, considering, 61–64
nameless functions, 174–175
names.
advice about, 64–65
avoiding overwriting, 65–66
choosing, 67
making searchable, 64
prefixes in, 63–64
sequential numeric suffixes in, 64
namespaces, 90–91
`__ne__()` comparison dunder method, 336
`__neg__()` numeric dunder method, 328
nested conditional expressions, 102
nested loops, using in big O analysis, 241
`next()` function, calling, 126
no operation, explained, 74
nondeterministic function, 173. *See also* functions
`None`, using `==` (equality) operator with, 94–95
Not Equal operation, 336
`NotImplementedError`, raising, 75
numbers, magic, 71–73.
numeric dunder methods, 325–328

O

`O(1)`, Constant time, 231–232
objects. *See also* mapping
and classes, 276
comparing, 154–155, 334
creating from classes, 278
defined, 111–114
mutable and immutable, 114–117
sorting, 336

O(log n), Logarithmic, 232
O(n!), Factorial Time, 234–235
O(n), Linear Time, 232
O(n log n), N-Log-N Time, 232–233
O(n2), Polynomial Time, 233
O(n11), Exponential Time, 233–234
OOP (object-oriented programming). *See also* dunder methods; inheritance
  creating objects from classes, 278
  defined, 275
  designing classes, 290–291
  encapsulation, 307–308
  filling out form, 276–278
  and non-OOP examples, 285–290
  polymorphism, 308
  properties, 316–322
  summary, 292
  tic-tac-toe, 285–290
  type() function and __qualname__attribute, 284–285
  using class and static features, 307
  WizCoin class, 279–284
open()
  and close() functions, 93–94
  and readlines() functions, 126
operator module, 333, 336–337
operators, chaining, 103, 105, 151, 159–160
optimizations
  preallocated integers, 154
  string interning, 155
__or__() numeric dunder method, 328
ordinal, getting for characters, 146–147
os.chdir(), using, 20

P

packages
  defined, 120–121
  and modules, 120–121
parameters vs. arguments, 128
ParentClass, creating, 294–295
PascalCase, 60
pass statement
  relationship to stubs, 74–75
  using with except block, 79–80
PATH and environment variables, 35–39
pathlib module, importing, 18–19
paths, specifying, 18–21
p-code, explained, 129
PEP (Python Enhancement Proposal) 8
  documentation, 67
  naming conventions, 61
  and style guides, 46–47
Perl programming language, 90
pip list, running, 14
polymorphism, explained, 308
portable code, explained, 129
porting vs. backporting, 196
__pos__() numeric dunder method, 328
positional arguments, defined, 166–167. *See also* arguments
__pow__() numeric dunder method, 328
practice projects. *See also* projects
  Four-in-a-Row, 259–271
  The Tower of Hanoi, 248–259
preallocated integers, 154.
premature optimization, 226
print debugging, 75–76
print() function
  arguments for, 168
  passing list to, 166
  using with wrapper functions, 171
private attributes and private methods, 282–284. *See also* methods
processes and programs, 21–22
professional comments, 186
profiling, explained, 228
program vs. script, 129–130
programming help, asking for, 9–14
programming language vs. scripting language, 129–130
programs. *See also* Python programs
  finding, 35
  and processes, 21–22
  running from command line, 23–24, 26
  running without command line, 39–42
  vs. scripts, 129–130
project folder, contents of, 200
projects, creating with Cookiecutter, 200–202. *See also* practice projects

properties
vs. attributes, 128–129
best practices, 322
read-only, 320–321
turning attributes into, 316–319
using, 316
public access attributes and methods, 283
pure function, 173–174. *See also* functions
push command, using in Git, 221–223
.py source code files, locating, 200
.pyc files, bytecode in, 129
py.exe program, running, 26–27
Pyflakes, installing, 9
PyPy just-in-time compiler, 108
Python
documentation, 121
error messages, 4–8
glossary, 108, 131
language and interpreter, 108–109
programming language, 109
Python programs, running without command line, 39–42. *See also* programs; *The Zen of Python*
Python Standard Library, 120–121. See also library vs. framework vs. SDK vs. engine vs. API
pythonic code, core of, 104

Q

`__qualname__`attribute and `type()` function, 284–285
questions, asking, 10–11, 14–15

R

`__radd__()` reflected numeric dunder method, 330
raising exceptions, 90, 178–179
`__rand__()` reflected numeric dunder method, 330
`range()` vs. `enumerate()`, 92–93, 103–104
`rd` and `rmdir` commands, 34–35
`__rdivmod__()` reflected numeric dunder method, 330
`readlines()` and `open()` functions, using, 126
README files, 200, 211–212, 215–216, 218
read-only properties, 320–321
`RecursionError` exception, raising, 318–319
references, explained, 137–138
reflected numeric dunder methods, 328–330
relative vs. absolute paths, 20–21
renaming files and folders, 32–33
repo. *See also* Git
cloning for GitHub repo, 222–223
creating, 223
creating on computer, 206–207
deleting and moving files in, 215–216
deleting files from, 214–215
ignoring files in, 209–210
and version control systems, 200
`__repr__()` method, using, 325
`repr` string, sensitive information in, 325
requests module, *sessions.py* file, 188–189
return values and data types, 177–178. *See also* values
`__rfloordiv__()` reflected numeric dunder method, 330
`__rlshift__()` reflected numeric dunder method, 330
`rm` (removing files and folders) command, 33–34
`__rmatmul__()` reflected numeric dunder method, 330
`__rmod__()` reflected numeric dunder method, 330
`__rmul__()` reflected numeric dunder method, 330
roll back, performing in Git, 217–220
root folder, explained, 18
`__ror__()` reflected numeric dunder method, 330
`__round__()` numeric dunder method, 328
`__rpow__()` reflected numeric dunder method, 330
`__rrshift__()` reflected numeric dunder method, 330
`__rshift__()` numeric dunder method, 328

__rsub__() reflected numeric dunder method, 330
__rtruediv__() reflected numeric dunder method, 330
running processes, viewing, 22
runtime
  defined, 226
  quickening for functions, 173
  vs. syntax vs. semantic errors, 126–127
__rxor__() reflected numeric dunder method, 330

S

%s conversion specifiers, using, 96–97
script vs. program, 129–130
scripting language vs. programming language, 129–130
SDK vs. library vs. framework vs. engine vs. API, 130
self and __init__(), 280–282
semantic vs. syntax vs. runtime errors, 126–127
semicolons (;), using with timeit module, 227
sensitive information in repr strings, 325
sequence comparisons, 335–336. *See also* comparison operators
sequences
  defined, 119–120
  and iterables, 125
sessions.py file in requests module, 188–189
set types, defined, 119–120
setdefault(), using with dictionaries, 98–100
setters
  and getters, 315, 318
  using to validate data, 319–320
sh file, 22–23
shell programs, 22–23
side effects, 172–174
significant indentation, 91–92, 104. *See also* indentation
single quote ('), using, 46
slice syntax, explained, 97
snake_case, 60
snapshots, saving with Git, 200
software license, file for, 200
sort() function, behavior of, 146–147, 151, 332, 336
source code, avoiding dropping letters from, 62. *See also* code; code smells
source file encoding and magic comments, 187–188
space characters, rendering on screen, 47
spacing within lines, 48–51
Stack Overflow, building answer archive, 12
stack trace, 4–7
staged files
  committing, 211
  unstaging in Git, 218
statements vs. expressions, 122–123
static analysis, explained, 8, 192–194
static methods, 306–307
string concatenation, 144–146, 151
string interning, 155–156
strings
  formatting, 95–97
  as immutable objects, 144
  immutable quality of, 116
  interpolating, 104
stubs, relationship to code smells, 74
style guides and PEP (Python Enhancement Proposal) 8, 46–47
__sub__() numeric dunder method, 327
subclass, relationship to inheritance, 296. *See also* isinstance()
subfolders, listing contents of, 31
Sublime Text editor, 193
subprocess.run() function, 27
subtract() function, creating, 172
sum() function, 168
summary comments, 185
super class, relationship to i nheritance, 296
super() function, relationship to overriding method, 297–299
switch statement vs. dictionaries, 100–101
syntax
  catching errors, 6

misuse of, 92–95
vs. runtime vs. semantic errors, 58, 126–127
`sys.getsizeof()` function, 137–138
system environment variables, 38

T

tab completion, 27–28
Task Manager, opening, 22
terminal window
clearing, 35
opening, 23, 41
ternary operator, 101–102
tests folder, contents of, 200
text editor, Sublime Text, 193
tic-tac-toe program
creating, 285–290
MRO (method resolution order), 311–312
tilde (~), using in macOS, 23
`timeit` module, using to measure performance, 226–228. *See also* modules
`time.time()` function, 72, 227
tkdiff, installing on macOS, 213
TODO comments and codetags, 187
The Tower of Hanoi puzzle
`getPlayerMove()` function, 163, 165, 254–257, 268
output, 249–250
restrictions, 248
source code, 250–252
summary, 271–272
writing code, 252–259
tracebacks, examining, 4–7
`True` and `False` keywords, 158–159
`__truediv__()` numeric dunder method, 327
`__trunc__()` numeric dunder method, 328
tuples
identities, 119
immutable quality of, 116–117
using commas with, 150
values of, 116
type coercion vs. type casting, 128
`type()` function and `__qualname__`attribute, 284–285
type hints, 182, 190–196
types, defined, 276
typing, minimizing with tab completion, 27–28
typing module, 195–196

U

Ubuntu Linux, running Python programs on, 41–42
underscore (_). *See also* double underscore (__)
PEP 8's naming conventions, 60–61
as prefix for methods and attributes, 291–292
private prefix, 81
using with `_spaces` attribute, 290
using with dunder methods, 120
using with private attributes and methods, 283
using with `WizCoin` class, 279
undo features, 199, 217–220
Unicode resource, 188
unit tests, folder for, 200
Unix operating system, shell programs, 22–23
URL links, including in comments, 183
user environment variables, 38
UTF-8 encoding, 187–188

V

validating data using setters, 319–320
values. *See also* Boolean values; return values and data types
defined, 111–114
modifying in place, 115
variable names, 64, 66. *See also* names
variable values, 103–104
variables. *See also* metasyntactic variables
vs. attributes, 124
box vs. label metaphor, 112–113
checking values, 103–104
with numeric suffixes, 76
swapping, 227
variadic functions, creating, 167–171. *See also* functions
version control systems, 199–200
vertical spacing, 51–53
volumes, explained, 18

W

`watch` command, using with Git, 207
`webbrowser` module, 160
`where` command, 35
`which` command, 35
`while` keyword, 110
`while` loops
    in big O analysis, 241
    and lists, 134–140
wildcard characters, 28–29
Windows, running Python programs on, 40–41
WinMerge, downloading, 212–213
`with` statement, 93–94
`WizCoin` class, creating, 279–284
worst-case scenario, measuring with Big O, 235
wrapper functions, creating, 171–172. *See also* functions

X

XOR algorithm, using, 226–227
`__xor__()` numeric dunder method, 328

Z

Zakharenko, Nina, 131
The Zen of Python, 88–91. *See also* programs; Python programs
zero-based indexing, using, 117
Zsh and Z shells, 23