Symbols
+ operator example, polymorphism, 193–194
* (asterisk), using with import statement, 61
@ (at) property, 174–177
== (equal to), magic method name, 196
> (greater than), magic method name, 196
>= (greater than or equal to), magic method name, 196
< (less than), magic method name, 196
<= (less than or equal to), magic method name, 196
!= (not equal to), magic method name, 196

A
abc (abstract base class) module, Python Standard Library, 232
AbortTransaction exception, 79, 81
absolute path, using with pygame, 101
abstract base class (abc) module,
Python Standard Library, 232
abstract classes and methods, 231–234.
See also classes; methods
abstraction, 179–181
Account class	error handling in, 59
with exceptions, 78–79
testing, 62
Account objects, 59, 71
dictionary of, 66–67
in lists, 64–66
accounts, creating, 62–64
Alert dialog, Dodger game, 343
angle brackets (<>), values in, 16

Animation class, 149, 304–306
animation classes
building, 296–304
merging, 304
in pygwidgets, 304–309
SimpleAnimation class, 296–300
SimpleSpriteSheetAnimation class, 300–304
animation program, pygwidgets package, 308–309
Answer dialogs, Dodger game, 345–347
anti-aliased line, drawing, 118
API (application programming interface), 137, 158
arc, drawing, 118
arguments
methods and parameters, 144
passing to methods, 40–41
rearranging in calls to methods, 53
asterisk (*), using with import statement, 61
at (@) property, 174–177

B
background music, playing in pygame, 115–116
Baddies and Goodies, 347
Ball class, 122–125
Ball objects, creating, 125–127
Balloon game
main code, 252–256
module of constants, 253
object diagram, 252
project folder, 252
screenshot, 251
source files, 252
balloon manager, 256–258
Balloon sample program, 251–261
Balloon class and objects, 258–261
bank account class, 58–60
bank account simulations. See also procedural implementations
operations and data, 7–8
table of data, 15–16
Bank class, 70, 79–81
Bank object, 70–71, 82
bank program, using exceptions in, 78–83
base class, inheritance, 212–214, 227–231
Blackjack deck, 278
blit() method, 102, 114, 124, 137
Boolean True, 105
Button class, building, 128–130
buttons, building, 131–132
C
callbacks, 137–141
CanastaDeck class, 279
Card class, 268–270
card games. See also Higher or Lower card game
Blackjack, 278
Card class, 268–270
Deck class, 270–272
Higher or Lower game, 272–276
testing with __name__, 276–278
with unusual decks, 279
Cartesian coordinate system, 91–95, 201
catching exceptions, 76
child class, inheritance, 212
circle, drawing, 119
Circle class, 187–190, 227, 230
class code, importing, 60–61
classes. See also abstract classes and methods
building, 33–35
creating instances from, 31–32
creating objects from, 28–30
form of, 26
implementing data types as, 32–33
and inheritance, 212–213
inheriting from same base class, 227–231
inside vs. outside, 164–165
making available, 29–30
and methods, 51
and objects, 23–25
objects and instantiation, 25–33
representing physical objects as, 35–44
in use, 45
writing, 26–27
class hierarchy, inheritance, 236–238
class scope, 27
class statement, inheritance, 216
class variables. See also variables
constants, 249–250
for counting, 250
creating, 248–249
client code
explained, 164
using direct access, 170
collidepoint(), 104
comparison operator magic methods, 195–196
composition, 71
composition and inheritance, 238
Controller object, 371–372
CountDownTimer class, 293–294
counting objects, 250
CountUpTimer class, pyghelpers package, 291–293
CPython, 242
customAnswerDialog dialog, Dodger game, 347
CustomButton class, 148, 235–236
CustomCheckBox class, pygwidgets package, 149
CustomRadioButton class, pygwidgets package, 149
customYesNoDialog dialog, Dodger game, 345
D
data, validating, 168–170
databases, accessing with objects and XTRAS, 178–179
data types, implementation as classes, 32–33
debugging approach, 203
Deck class, 270–272
decorators and @property, 174–177
__del__() method, 246–248, 260
design patterns, MVC (Model View Controller), 367–374
dice roll data, MVC (Model View Controller) design pattern, 369–370
__dict__ dictionary, 261–263
dictionary
of account objects, 66–67
using with instance variables, 261–263
DimmerSwitch class, 33, 48–50, 52–53
DimmerSwitch objects, creating, 50, 53
direct access, avoiding, 166–170
Director from Macromedia project, 178
DisplayMoney class, 222–227
DisplayText class, 149, 222
Dodger game
Alert dialog, 343
Answer dialogs, 345–347
customAnswerDialog dialog, 347
customYesNoDialog dialog, 345
extensions to, 366
implementation, 348
modal dialogs, 342–347
overview, 347
ScenePlay class, 351–355
textYesNoDialog dialog, 343
Yes/No dialog, 344
Don't Repeat Yourself (DRY), 253
Dragger class, pygwidgets package, 149
drawing shapes, 116–120
draw() method used with pygwidgets, 150, 155, 157, 193, 308
draw() method used with scenes, 325
draw.rect(), 187
DRY (Don't Repeat Yourself), 253
E
educational project, 178–179
ellipse, drawing, 119
Ellipse class, 192
Employee class, inheritance, 218
encapsulation
decorators and @property, 174–177
direct access, 166–172
with functions, 164
interpretations of, 165–172
making instance variables more private, 172–173
with objects, 164–165
in pygwidgets classes, 177
enter() method used with scenes, 325
__eq__() magic method name, 196, 198–199
equal to (=), magic method name, 196
error handling
in Account class, 59
with exceptions, 76–78
event-driven programs, 95–96
event loop, 99
except and try, 76–77
exceptions
in bank program, 78–83
error handling with, 76–78
handling, 81–83
F
file display example, 368
Fraction class, magic methods, 205–208
functions
encapsulation with, 164
len(), 164
vs. methods, 28
super(), 216
vars(), 52
G
Game class, 274
Game object, 272
games, transient objects in, 242
garbage collection, 248
__ge__() magic method name, 196
getrefcount() function, 244
getSceneKey() method used with scenes, 328
getters and setters, 170–171, 175–176
Ghostbusters, 138
GitHub repository, accessing, 157
global scope, 27
Goodies and Baddies, 347
goToScene() method used with scenes, 326
graphic file formats, using with pygame, 100–101
greater than (>), magic method name, 196
greater than or equal to (>=), magic method name, 196
__gt__() magic method name, 196, 198
GUI programs, event-driven model, 95–96

H
handleEvent() method used with pygame, 150, 192, 221, 307, 312
handleInputs() method used with scenes, 363
help() function, 152
Higher or Lower card game, 268.
   See also card games
   Game object, 274–276
   implementation, 4–7
   main program, 272–274
   representing data, 4
   reusable code, 7
HighScoresData class, 363

I
IDLE development environment, 90, 100–101
Image class, pygwidgets package, 149. See also subimages
ImageCollection class, pygwidgets package, 149, 157
implementation vs. interface, 84–85, 137
importing class code, 60–61
import statements, 98
inheritance. See also multiple inheritance
abstract classes and methods, 231–234
base class, 212
class hierarchy, 236–238
client’s view of subclass, 218–219
and composition, 238
difficulty of programming with, 238–239
DisplayMoney class, 222–227
employee and manager example, 214–218
example usage, 224–227
implementing, 213–214
InputNumber class, 219–222, 224–227
“is a” relationship, 213
Law of Demeter, 238
in object-oriented programming, 212–213
and pygwidgets, 234–236
real-world examples, 219–227
from same base class, 227–231
subclass, 212
test code, 217–218
use by pygwidgets, 234–236
initialization parameters, 43–44
__init__() method, 28, 37, 43, 216
   Account class, 59, 79
   Ball class, 123
   Bank class, 73
   inheritance examples, 228–229, 232–233
   InputNumber, 221
   pronouncing, 194
   property decorators, 175–176
   SceneMgr class, 335
   subclass in inheritance, 216
   using, 27
input() function, 133, 155
InputNumber class, 219–222, 224–227
InputText class, 149, 219, 222
installing
   pygame, 90–91
   pyghelpers, 287
   pygwidgets package, 149–150
instance and scope variables, 27–28
instances, 26, 31–32, 41–43. See also multiple instances
instance scope, 27
instance variables. See also slots
   changing into calculations, 167–168
   changing names of, 166–167
   using, 27, 165
instantiate, explained, 26
instantiation process, 25–33
interactive menu, building, 68–70
interface vs. implementation, 84–85, 137
Invent Your Own Computer Games with 
Python, 341
“is a” relationship, inheritance, 213
isInstanceOf() function, 196
items() method, 271

J
JSON format, 363–365

K
keyword parameters, pygame, 145–146

L
Law of Demeter, inheritance, 238
__le__() magic method name, 196
leave() method used with scenes, 325
len() function, 15, 164
less than (<), magic method name, 196
less than or equal to (<=), magic
method name, 196
LIFO (last in, first out) order, 179
LightSwitch class and test code, 30
light switch example, 22–23, 25–31
LightSwitch object, instantiating, 29
line, drawing, 119
Lingo language, 178
local scope, 27
__lt__() magic method name, 196, 198, 200

M
Macromedia project, 178
magic methods, 194–201. See also
methods
Manager class, inheritance, 219
memory management, slots, 261–263
memory used by objects. See also objects
Balloon sample program, 251–261
class variables, 248–250
managing with slots, 261–263
mental models, 49–52
menu, making interactive, 68–70
methods. See also abstract classes and
methods; magic methods
calling, 30, 41
calling for objects, 30–31
calling on lists of objects, 83–84

N
__name__, testing card games with, 276–278
naming convention, 26
__ne__() magic method name, 196
not equal to (!=), magic method
name, 196

O
object composition, 71
object lifetime
cascading deletion, 246–248
death notice, 246–248
garbage collection, 246–248
reference count, 242–246
transaction objects, 242
transient objects, 242
object manager object, creating, 70–76
object-oriented programming (OOP)
explained, 3
as solution, 45
tenets, 374
wrap-up, 374–375
See also pygame
Ball class, 122–125
Ball objects, 125–127
callbacks, 137–141
demo ball with SimpleText and SimpleButton, 135–137
interface vs. implementation, 137
program with buttons, 131–132
reusable object-oriented button, 127–132
reusable object-oriented text display, 133–135
SimpleButton, 130–131
SimpleText class, 133–135
steps to display text, 133
object-oriented solutions
classes, 19–20
objects. See also memory used by objects; physical objects
calling methods of, 30–31
calling methods on lists of, 83–84
and classes, 23–25
counting, 250
creating from classes, 28–30
definition of, 33
encapsulation with, 164
garbage collection, 248
inside vs. outside, 164–165
owning data, 165
reference count, 242–248
sending messages to, 184
string representations of values in, 203–205
transient type, 242
with unique identifiers, 66
variables referring to, 244
object scope, 27
OOP (object-oriented programming)
explained, 3
as solution, 45
tenets, 374
wrap-up, 374–375
operators
magic methods, 194–201
polymorphism for, 193–203
prefix, 26
parent class, inheritance, 212
path, using with pygame, 100
pathname, using with pygame, 100
patterns, extending with polymorphism, 192
physical objects. See also objects
building software models of, 22–23
classes and objects, 23–25, 45
classes, objects, and instantiation, 25–33
complicated classes, 33–44
OOP as solution, 45
representing as classes, 35–44
PIE (polymorphism, inheritance, encapsulation), 161
PinochleDeck class, 279
pixels
colors, 94–95
in window coordinate system, 91
playing sounds, 114–116
Play scene, Rock, Paper, Scissors game, 315
polygon, drawing, 119
polymorphism
classic example of, 184–185
extending patterns, 192
Fraction class with magic methods, 205–208
magic methods, 194–201
main program creating shapes, 190–192
for operators, 193–203
pygame shapes, 185–192
and pygwidgets, 192–193
sending messages to real-world objects, 184
string representations of values in objects, 203–205
vector example, 201–203
pop operation example, 201–203
positional parameters, pygame, 145
primitive shapes, drawing in pygame, 116–120
print() function, 133, 205
procedural implementations. See also bank account simulations
classes, 19–20
Higher or Lower card game, 3–4
problems with, 18–19, 45
properties
@ (at) and decorators, 174–177
and abstraction, 181
explained, 174
push operation, using with stacks, 179
PyCharm IDE, 100–101
*.py file extension, 61
pygame. See also object-oriented pygame
anti-aliased line, 118
arc, 119
bringing up blank window, 97–100
Cartesian coordinate system, 91–94
circle, 119
colors in, 94–95
detecting mouse click, 102–105
drawing images, 100–102
drawing shapes, 116–120
ellipse, 119
event-driven programs, 95–96
handling keyboard, 105–109
installing, 90–91
line, 119
location-based animation, 109–111
pixel colors, 94–95
playing sounds, 114–116
polygon, 119
primitive shapes, 118–120
recognizing key presses, 105–107
rect objects, 104, 107, 111–114, 119
repeating keys in continuous
mode, 107–109
Splash scene, 314
state machine example, 314–319
window coordinate system, 91–95
pygame.display.set_mode() function, 98
pygame GUI widgets. See also
pygwidgets package
arguments, functions, and
methods, 144–148
keywords and default values, 148
None as default value, 146–147
positional and keyword
parameters, 145–146
pygame.Rect(), 104
pygame shapes
Circle and Triangle shape classes,
187–190
Square shape class, 186–187
PygAnimation base class, 304, 307–308
pyghelpers package
classes for tracking time, 290
CountDownTimer class, 293–294
CountUpTimer class, 291–293
installing, 287
pygwidgets classes, encapsulation in, 177
pygwidgets package. See also pygame
GUI widgets
adding images, 151
Animation class, 304–309
animation program, 308–309
button object, 154
buttons, 152–154
checkboxes, 152–154
classes, 148–149, 157
class hierarchy, 237
consistency of API, 158
CustomButton class, 153–154
design approach, 150–151
DisplayText class, 155, 222
design example program, 157–158
goals and classes, 148–149
images, 151
and inheritance, 234–236
InputText class, 219
polymorphism, 192–193
PygAnimation base class, 307–308
radio buttons, 152–154
setting up, 149–150
sprite module, 151
SpriteSheetAnimation class, 306–307
TextButton class, 152–153
text output and input, 154–157
using buttons, 154
PyPI (Python Package Index), 149
Python, philosophy of, 242
Python Software Foundation, 242
Python Standard Library
abc (abstract base class) module, 232
calls to, 76–77
getrefcount() function, 244
Python Tutor website, 50, 243
R
raise statement and custom exceptions, 77–78
RAM memory space, 262
random package, 104
real-world objects, sending messages to, 184
receive() method used with scenes, 334
rectangle, drawing, 120
Rectangle class
inheritance example, 233–234
with magic methods, 196–198
reference count, 242–246
decrementing, 245–246
incrementing, 245
relative path, using with pygame, 100–101
respond() method used with scenes, 333
Results scene, Rock, Paper, Scissors game, 315–316
reusable object-oriented button, building, 127–132
reusable object-oriented text display, building, 133–135
RGB (red, green, blue), 94
Rock, Paper, Scissors game
Play scene, 315
Results scene, 315–316
Splash scene, 314
using scenes, 328–332
run() method of the scene manager, 336, 349

S
Sample class, 250
Scene base class, 322
scene manager
building scenes, 323–326
communication between scenes, 338–340
demo program, 320–328
example scene, 326–328
features, 319–320
implementation of, 334–340
main methods, 337–338
main program, 322–323
methods for implementing scenes, 324–325
navigating between scenes, 326
project folder, 321
quitting program, 326
Rock, Paper, Scissors, 328–332
run() method, 336–337
using, 319–320
SceneMgr class, 337
ScenePlay class, Dodger game, 351–355
scenes
communication between, 332–334
current and target, 332
state machine approach, 312–319
testing communications among, 334
scope and instance variables, 27–28
screensaver ball, building with object-oriented pygame, 121–127
“self,” meaning of, 52–55
self parameter, 41–42
self. prefix, 27
send() method used with scenes, 333
sendAll() method used with scenes, 334
setters and getters, 170–171, 175–176
Shape class
inheritance example, 232–233
using as base class, 228
shapes, drawing in pygame, 116–120
SimpleAnimation class, 296–300
SimpleButton class, 129, 131–132, 139–141
SimpleButton objects, 130–131, 137–138, 141
SimpleText class, 133–137
SimpleText object, 135
Slider class, 193
Slider Puzzle user interface, 290, 293
slots, using for memory management, 261–263. See also instance variables
software models, building for physical objects, 22–23
sound effects, playing in pygame, 114–116
SpaceShip class, 249
special methods, 194
Splash scene, 313–314, 347
SpriteSheetAnimation class, 149, 304, 306–307
sprite sheet image, 300
Square class, 195, 227
    inheritance example, 229
    pygame shapes, 186–187
    for reference counting, 243
Square object, 243
Stack class, 181
stack operations, 179–180
state diagram, 316
state machine, pygame example, 312–319
statistical display example, 368–373
__str__() method, 203–204
subclasses
    client’s view of, 218–219
    inheritance, 212–213, 215–217
    inheriting from base class, 227
subimages, creating, 300.
    See also Image class
superclass, inheritance, 212
super() function, 216
Sweigart, Al, 341

T
temporary variable, using, 66
test code
    accounts, 62–64
    creating, 61–62
    inheritance, 217–218
test programming, 177
textAnswerDialog dialog, Dodger game, 346
TextButton class, 148, 235
TextCheckBox class, pygwidgets package, 149
text display, building, 133–135
TextRadioButton class, pygwidgets package, 149
textYesNoDialog dialog, Dodger game, 343
throwing exceptions, 76
time, displaying, 290–294
Timer class, 287–290
timer demonstration program, 282
timer event, 284–285
Timer object, 288
timers
    building into main loop, 286
    calculating elapsed time, 285–287
counting frames, 283
demonstration program, 282
    implementing, 283–287
    installing pyghelpers, 287
    overview, 281–282
toggle, using, 38
transaction objects, 242
transient objects, 242
Triangle class, 187, 227, 230–231
try/except techniques, 76–81
tuple, setting x- and y-coordinates as, 151
TV class, creating, 35–40
TV objects, creating, 42
type() function, 32
U
update() method used with scenes, 325
V
variables. See also class variables
    referring to same object, 244
    using temporarily, 66
vars() function, 52
vector example, polymorphism, 201–203
View object, 371, 373
W
WidgetWithFrills class, 214
window coordinate system, pygame, 91–95
working directory, using with pygame, 100
X
x- and y-coordinates, setting as tuple, 151
XTRAs and objects, accessing databases with, 178–179
Y
Yes/No and Alert dialogs, Dodger game, 342–345