INDEX

Symbols
+ (addition), 26
+= (addition in place), 122
* (arbitrary arguments), 146
** (arbitrary keyword arguments), 148
{} (braces)
    dictionaries, 92
    sets, 104
@ (decorator), 221, 424
/ (division), 26
== (equality), 72, 74
** (exponent), 26
> (greater than), 75
>= (greater than or equal to), 75
# (hash mark), for comments, 29
!= (inequality), 74
< (less than), 75
<= (less than or equal to), 75
[] (list), 34
% (modulo), 116
+= (multiline strings), 115
* (multiplication), 26
\n (newline), 22
>>> (Python prompt), 4
- (subtraction), 26
\t (tab), 22
_ (underscore)
    in file and folder names, 10
    in numbers, 28
    in variable names, 17

class
    Aliens, 151–152, 178–179
    alien.py, 195–197
    Alien Invasion. See also Pygame
        aliens, 256–274
            building fleet, 259–262
            checking edges, 265
    collisions, with bullets, 267
    collisions, with ship, 270–273
    controlling fleet direction, 264–266
    creating an alien, 256–258
    dropping fleet, 265–266
    reaching bottom of screen, 273–274
    rebuilding fleet, 268–269
    bullets, 247–253, 266–270
    collisions, with aliens, 267
    deleting old, 250–251
    firing, 249–250
    larger, 268
    limiting number of, 251–252
    settings, 247
    speeding up, 269

levels
    modifying speed settings, 283–285
    resetting the speed, 285
    displaying, 294–296
    moving fleet, 254–296
    planning, 228
    Play button, 278–283
        Button class, 278–279
        deactivating, 282
Index

Alien Invasion (continued)

Play button (continued)

drawing, 279–280
hidinm mouse cursor, 282–283
resetting game, 281–282
starting game, 281
reviewing the project, 256
scoring, 286–298
all hits, 290
high score, 292–294
increasing point values, 290–291
level, 294–296
number of ships, 296–299
resetting, 289–290
rounding and formatting, 291–292
score attribute, 286
updating, 289
settings, storing, 232–233
ship, 233–244
adjusting speed, 242–243
continuous movement, 239–242
finding an image, 233–234
limiting range, 243–244
amusement_park.py, 80–82
and keyword, 75
antialiasing, 279
API. See application programming interface
apostrophe.py, 24–25
append() method, 37–38
application programming interface (API), 355
API call, 355–357
GitHub API, 368
Hacker News API, 368–371
processing an API response, 357–362
rate limits, 362
requesting data, 356–357
visualizing results, 362–368
arguments, 131. See also under functions
as keyword, 151–152
assertions, 213, 217–218
attributes, 159. See also under classes

B

banned_users.py, 76–77
bicycles.py, 34–35
Boolean values, 77
Bootstrap, 433. See also under Django
braces ({}), 92
sets, 104
break statement, 121
built-in functions, 467

C
calls (functions), 130, 132–135
car.py, 162–178
cars.py, 43–45, 72
cities.py, 121
classes
attributes, 159
accessing, 160
default values, 163–164
modifying, 164–166
creating, 158–161
importing, 173–179
multiple classes, 175–176
single classes, 174–175
inheritance, 167–172
attributes and methods, 169
child classes, 167–170
composition, 170
__init__() method, 167–169
instances as attributes, 170–172
overriding methods, 170
parent classes, 170
subclasses, 168
super() function, 168
superclasses, 168
instances, 157
methods, 159
calling, 160
chaining, 185
__init__() method, 159
modeling real-world objects, 172–173
multiple instances, 161
naming conventions, 158
objects, 157
style guidelines, 181
comma-separated value files. See CSV files

comment.py, 29
comments, 29–30
conditional tests, 72–77. See also

if statements

confirmed_users.py, 124–125
constants, 28
continue statement, 122
counting.py, 117–118, 122–123
CSV files, 330–341
csv.reader() function, 330–333
error checking, 338–341
file headers, 330–332

D
data analysis, 301
databases. See under Django
data visualization, 301. See also

Matplotlib; Plotly
datetime module, 333–335
deat\textunderscore valley\_highs\_lows.py, 339–341
decorators, 221–223, 423–425
default values
class attributes, 163–164
function parameters, 134–135
definition (functions), 130
def keyword, 130
del statement

with dictionaries, 96
with lists, 38–40
dice\_visual\_d6d10.py, 326–327
dice\_visual.py, 324–326
dictionaries

defining, 92
empty, 94
formatting larger, 96–97
KeyError, 98
key-value pairs, 92
adding, 93–94
removing, 96
looping through
keys, 101–102
keys in order, 102–103
key-value pairs, 99–101
values, 103–104
methods
get(), 97–98
items(), 99–101
keys(), 101–103
values(), 103–104
nesting
dictionaries in dictionaries, 110–111
dictionaries in lists, 105–108
lists in dictionaries, 108–109
ordering in, 94, 102–103
sorting a list of, 370
values
accessing, 92–93, 97–98
modifying, 94–96
die.py, 320
die\_visual.py, 320–321
dimensions.py, 66–67
div (HTML), 437
division\_calculator.py, 192–195
Django. See also Git; Learning Log project
accounts app, 415–423
creating app, 415–416
logging out, 419–420
login page, 416–419
registration page, 420–423
admin site, 381–386
associating data with a user, 425–430
Bootstrap, 434–445
card, 443
collapsible navigation, 437
container, 440
django-booststrap5 app, 434
documentation, 444
HTML headers, 435–436
jumbotron, 440–441
list groups, 443
navigation bar, 436–439
styling forms, 441–442
commands
createsuperuser, 382
flush, 427
makemigrations, 381, 385, 426
migrate, 377
runserver, 377–378, 383, 392
shell, 386
startapp, 379, 415
startproject, 376
creating new projects, 376
Django (continued)

databases
  cascading delete, 384
  creating, 376
  foreign keys, 384, 425
  many-to-one relationships, 384
  migrating, 377, 381, 385, 426
  non-nullable field, 427
  Postgres, 447
  queries, 398, 428
  querysets, 386–387, 395, 398, 426–428
  resetting, 427
  SQLite, 377

deployment, 445–461, 493–501
  committing the project, 453
  configuration files, 447–450
  creating Platform.sh project, 453–455
  creating superuser, 456–457
  custom error pages, 459–460
  deleting projects, 461
  free trial limits, 446
  gunicorn, 447
  ignoring files, 452–453
  installing Platform.sh CLI, 446, 497–500
  installing platform.shconfig, 446
  other deployment approaches, 500
  Platform.sh, 445
  Postgres database, 447, 450–451
  psycopg2, 447
  pushing a project, 455
  pushing changes, 458, 460
  requirements.txt, 446
  securing project, 457–460
  settings, 451
  SSH sessions, 456–457
  troubleshooting, 494–501
  using Git, 451
  viewing project, 456
  development server, 377–378, 383, 392

documentation
  model fields, 380
  queries, 388
  templates, 400
  forms, 404–423, 429–430
  csrf_token, 407
  GET and POST requests, 406
  ModelForm, 404, 408
  save() method, 405–406, 409–410, 430
  templates, 407, 410–411, 413, 417, 419, 422
  validation, 404–406
  widgets, 408

HTML
  anchor tag (<a>), 393
  <body> element, 437
  comments, 437
  <div> elements, 437
  <main> element, 440
  margins, 440
  padding, 440
  <p> elements, 391
  <span> elements, 438

HTTP 404 error, 428–429, 459–460
INSTALLED_APPS, 380
installing, 375–376
localhost, 378
logging out, 419–420
@login_required decorator, 423–424
login template, 417
mapping URLs, 388–390, 397–398
migrating the database, 426–427
models, 379
  activating, 380–381
  defining, 379, 384
  foreign keys, 384, 425
  registering with admin, 382–383, 385–386
  __str__() method, 380, 384
projects (vs. apps), 379
redirect() function, 405–406
release cycle, 376
restricting access to data, 427–430
settings
  ALLOWED_HOSTS, 451
  DEBUG, 457–458
  INSTALLED_APPS, 380–381, 415–416, 434
  LOGIN_REDIRECT_URL, 417–418
  LOGIN_URL, 424
  LOGOUT_REDIRECT_URL, 420
  SECRET_KEY, 451
  shell, 386–387, 426–427
  starting an app, 379
  styling. See Django: Bootstrap
  superusers, 382, 456–457
templates
  block tags, 393
  child template, 393–394
  context dictionary, 395
  filters, 399
  forms in, 407
  indentation in, 393
  inheritance, 392–394
  linebreaks, 399
  links in, 392–393, 399
  loops in, 395–397
  parent template, 392–393
  template tags, 393
  timestamps in, 398–399
  user object, 418
  writing, 390–392
URLs. See Django: mapping URLs
  UserCreationForm, 421–422
  user ID values, 426
  versions, 376
  view functions, 388, 390
  virtual environments, 374–375
docstrings, 130, 153, 181
dog.py, 158–162
dot notation, 150, 160
E
earthquakes. See mapping earthquakes
electric_car.py, 167–173
  module, 177–179
encoding argument, 195–196
enumerate() function, 331
eq_explore_data.py, 343–347
equality operator (==), 72, 74
eq_world_map.py, 347–352
even_numbers.py, 58
even_or_odd.py, 117
exceptions, 183, 192–199
  deciding which errors to report, 199
  else block, 194–195
  failing silently, 198–199
  FileNotFound error, 195–196
  handling exceptions, 192–196
  preventing crashes, 193–195
  try-except blocks, 193
  ZeroDivisionError, 192–195
exponents (**), 26
F
favorite_languages.py, 96–97, 100–104, 109
file_reader.py, 184–187
files
  encoding argument, 195–196
  FileNotFound error, 195–196
  file paths, 186
    absolute, 186
    exists() method, 203–204
    pathlib module, 184
    Path objects, 184–186, 330
    relative, 186
    from strings, 198
    on Windows, 186
  read_text() method, 185, 195–196
  splitlines() method, 186–187
  write_text() method, 190–191
first_numbers.py, 57
fixtures, 221–223
flags, 120–121
floats, 26–28
foods.py, 63–64
for loops, 49–56, 99–104. See also
dictionaries; lists
formatted_name.py, 137–139
f-strings
  format specifiers, 291–292
  using variables in, 20–21
full_name.py, 21
functions, 129–155
  arguments
    arbitrary, 146–149
    default values, 134–135
    errors, 136
    keyword, 133–134
functions (continued)
  arguments (continued)
    lists as, 142–145
    optional, 138–139
    positional, 131–133
  body, 130
  built-in, 467
  calling functions, 130, 132–135
  defining, 130
  importing, 149–153
    aliases, 151–152
    entire modules, 150–151
    specific functions, 151
  modifying a list in a function, 142–145
  modules, 149–153
  parameters, 131
  return values, 137–141
  style guidelines, 153

G
  GeoJSON files, 342–347, 350–351
  GET requests, 406. See Django: forms
  getting help
    Discord, 480
    official Python documentation, 479–480
    online resources, xxxv, 478
    r/learnpython, 480
    rubber duck debugging, 478
    searching online, 479
    Slack, 481
    Stack Overflow, 479
  three main questions, 477–478
  Git, 356, 451–453, 483–492. See also
    Django: deployment
    abandoning changes, 488–489
    adding files, 486
    branches, 486
    checking out previous commits, 489–491
    commits, 486–488
    configuring, 452, 484
    deleting a repository, 491–492
    .gitignore, 484
    HEAD, 490
    ignoring files, 484
    initializing a repository, 485
  installing, 484
  log, 487
  repositories, 356
  status, 485–486
  GitHub, 356
    greeter.py, 114–115, 130–131
    greet_users.py, 142

H
  Hacker News API, 368–371
  hash mark (#), for comments, 29
    hello_git.py, 484–491
    hello_world.py, 10–12, 15–19
  hidden files, 448, 485
    hn_article.py, 368–369
    hn_submissions.py, 369–371

I
  IDE (integrated development environment), 469–470
  if statements
    and keyword, 75
    Boolean expressions, 77
    checking for
      equality (==), 72
      inequality (!=), 74
      item in list, 76
      item not in list, 76
      list not empty, 86–87
    elif statement, 80–83
    else statement, 79–80
    if statements and lists, 85–88
    ignoring case, 73–74
    numerical comparisons, 74–76
    or keyword, 76
    simple, 78
    style guidelines, 89
    testing multiple conditions, 82–83
  immutable, 65
  import *, 152, 177
  import this, 30–31
  indentation errors, 53–56
  index errors, 46–47
  inheritance, 167–173. See also
    under classes
  input() function, 114–116
    numerical input, 115–116
    writing prompts, 114–115
insert() method, 38
itemgetter() function, 370
items() method, 99–101

J
JSON files
GeoJSON files, 342–347, 350–351
JSON data format, 201
json.dumps() function, 201–204, 343–344, 368
json.loads() function, 201–204, 343–344

K
keys() method, 101–103
key-value pairs, 92. See also dictionaries
keyword arguments, 133–134
keywords, 466

L
language_survey.py, 219
Learning Log project, 373
files, 392

404.html, 459
500.html, 459
accounts/urls.py, 416, 420
accounts/views.py, 421–422
admin.py, 382–383
base.html, 392–393, 396, 418–419, 422, 435–440
edit_entry.html, 413
forms.py, 404, 408–409
.gitignore, 452–453
index.html, 390–394, 440–441
ll_project/urls.py, 388–389, 416
login.html, 417, 441–442
models.py, 379–380, 384
new_entry.html, 410
new_topic.html, 407
.platform.app.yaml, 448–450
register.html, 422
requirements.txt, 446–447
routes.yaml, 450
services.yaml, 450
topic.html, 398–399, 443–444
topics.html, 395–396, 442–443
ongoing development, 460
pages, 391
edit entry, 412–414
home page, 388–394
login page, 416–419
new entry, 408–411
new topic, 404–408
registration, 420–423
topic, 397–400
topics, 394–397
writing a specification (spec), 374

len() function, 44–45
library, 184
Linux
Python
checking installed version, 8
setting up, 8–12, 465–466
terminals
running programs from, 12
starting Python session, 9
troubleshooting installation
issues, 10
VS Code, installing, 9
lists, 33
as arguments, 142–145
comprehensions, 59–60
copying, 63–64
elements
accessing, 34
accessing last, 35
adding with append(), 37–38
adding with insert(), 38
identifying unique, 104
modifying, 36–37
removing with del, 38–39
removing with pop(), 39–40
removing with remove(), 40–41
empty, 37–38
enumerate() function, 331

index 509
errors
- indentation, 53–56
- index, 46
- for loops, 49–56
  - nested, 108–109, 261–262
- indexes, 34–35
  - negative index, 35
  - zero index, 34–35
- `len()` function, 44–45
- naming, 33–34
- nesting
  - dictionaries in lists, 105–108
  - lists in dictionaries, 108–109
- numerical lists, 56–60
  - `max()` function, 59
  - `min()` function, 59
  - `range()` function, 58–59
  - `sum()` function, 59
- removing all occurrences of a value, 125
- slices, 61–62
- sorting
  - `reverse()` method, 44
  - `sorted()` function, 43–44
  - `sort()` method, 43
- square brackets, 34
- logical errors, 54
- `lstrip()` method, 22–23

M

macOS
- `.DS_Store` files, ignoring, 453
- Homebrew package manager, 499

Python
- checking installed version, 7
- setting up, 7–12, 464–465
- terminals
  - running programs from, 12
  - starting Python session, 7
- troubleshooting installation issues, 10
- VS Code, installing, 8
  - `magicians.py`, 49–56
  - `magic_number.py`, 74
  - `making_pizzas.py`, 150–152
- mapping earthquakes, 342–352. See also Plotly
  - downloading data, 343, 352
  - GeoJSON files, 342–347, 350–351
  - latitude-longitude ordering, 345
  - location data, 346–347
  - magnitudes, 346
  - world map, 347–348

Matplotlib
- axes
  - `set_aspect()` method, 313–314
  - removing, 317
- ax objects, 303
- colormaps, 310–311
- fig objects, 305
- `figsize` argument, 318
- formatting plots
  - `alpha` argument, 337–338
  - built-in styles, 306
  - custom colors, 310
  - labels, 303–304
  - line thickness, 303–304
  - plot size, 318
  - shading, 337–338
  - tick labels, 309–310
- gallery, 302
- installing, 302
- `plot()` method, 303–306
- `pyplot` module, 302–303
- `savefig()` method, 311
- saving plots, 311
- `scatter()` method, 306–311
- simple line graph, 302–306
- `subplots()` function, 303
- methods, 20
  - helper methods, 237
- modules, 149–152, 173–179. See also classes: importing; functions: importing
  - modulo operator (%), 116–117
- `motorcycles.py`, 36–41
- `mountain_poll.py`, 125–126
- `mpl_squares.py`, 302–306
- `my_car.py`, 174–175
- `my_cars.py`, 176–179
- `my_electric_car.py`, 176
N
name errors, 17–18
name_function.py, 211–217
name.py, 20
names.py, 211–212
nesting. See dictionaries: nesting; lists:
  for loops
newline (\n), 21–22
next() function, 330–331
None, 98, 140
number_reader.py, 202
numbers, 26–28
  arithmetic, 26
  constants, 28
  exponents, 26
  floats, 26–27
  formatting, 291–292
  integers, 26
  mixing integers and floats, 27–28
  order of operations, 26
  round() function, 291–292
  underscores in, 28
number_writer.py, 201
O
object-oriented programming
  (OOP), 157. See also classes
  or keyword, 76. See also if statements
P
pandas, 320
parameters, 131
parrot.py, 114, 118–121
pass statement, 198–199
paths. See files: file paths
PEP 8, 68–69
person.py, 139–140
pets.py, 125, 132–136
pip, 210–211
  installing Django, 374–376
  installing Matplotlib, 302
  installing Plotly, 320
  installing Pygame, 228
  installing pytest, 211
  installing Requests, 357
Linux, installing pip, 465–466
  updating, 210
pi_string.py, 187–189
pizza.py, 146–148
Platform.sh. See Django: deployment
players.py, 61–62
Plotly, 302, 319. See also mapping
  earthquakes; rolling dice
  chart types, 322
  customizing plots, 323, 325–326, 364
  documentation, 368
  fig.show() method, 322
  fig.write_html() method, 327
formatting plots
  axis labels, 323
  color scales, 349–350
  hover text, 350–351, 365–366
  links in charts, 366–367
  marker colors, 349–350, 367
  tick marks, 325–326
  titles, 323
  tooltips, 365–366
  update_layout() method,
    325–326, 364
  update_traces() method, 367
gallery, 320
histograms, 322
installing, 320
plotly.express module, 322,
  347, 368
px alias, 322
px.bar() function, 322–323, 363–367
saving figures, 327
scatter_geo() function, 347–352
pop() method, 39–40
positional arguments, 131–133. See also
  functions: arguments
POST requests, 406. See also
  Django: forms
printing_models.py, 143–145
Project Gutenberg, 196–197
prompts, 114–115
.py file extension, 15–16
Pygame. See also Alien Invasion
  background colors, 231–232
  clock.tick() method, 230–231
  collisions, 266–267, 270–271,
    289–290
  creating an empty window, 229–230
  cursor, hiding, 282–283
Pygame (continued)
displaying text, 278–280
ending games, 274–275
event loops, 229–230
frame rates, 230–231
fullscreen mode, 245
groups
  adding elements, 249–250
defining, 248–249
drawing all elements in,
  249–250, 257–258
emptying, 268–269
looping through, 249–251
removing elements from,
  250–251
updating all elements in,
  248–249
images, 234–236
installing, 228
levels, 283–285
Play button, 278–283
print() calls in, 251
quitting, 244–245
rect objects, 234–235
  creating from scratch,
    247–248
  get_rect() method, 234–235
  positioning, 234–235, 238–243,
    247–248, 256–262, 278,
    286–298
  size attribute, 261
responding to input, 230
events, 230
keypresses, 238–242
mouse clicks, 281–283
screen coordinates, 235
surfaces, 230
testing games, 268
pytest. See testing code
Python
  >>> prompt, 4
  built-in functions, 467
  checking installed version, 466
  installing
    on Linux, 465–466
    on macOS, 7–11, 464–465
    on Windows, 5–6, 463–464
  interpreter, 15–16
  keywords, 466
Python Enhancement Proposal
  (PEP), 68
  standard library, 179–180
terminal sessions, 4
  on Linux, 9
  on macOS, 7–8
  on Windows, 6
versions, 4
why use Python, xxxvi
python_repos.py, 357–362
python_repos_visual.py, 362–367
Q
qui value, 118
R
random_walk.py, 312–313
random walks, 312–318
  choice() function, 313
  coloring points, 315–316
  fill_walk() method, 312–313
  generating multiple walks, 314–315
  plotting, 313–314
  RandomWalk class, 312–313
  starting and ending points, 316–317
range() function, 58–59
read_text() method, 185, 195–196
refactoring, 204–206, 237–238, 260,
  269–270
remember_me.py, 202–206
removeprefix() method, 24
removesuffix() method, 25
Requests package, installing, 357
return values, 137–141
rollercoaster.py, 116
rolling dice, 319–327. See also Plotly
  analyzing results, 321–322
  Die class, 320
  different-size dice, 326–327
  randint() function, 320
  rolling two dice, 324–326
rubber duck debugging, 478
rstrip() method, 22–23
rw_visual.py, 313–318
scatter_squares.py, 306–311
sets, 103–104
sitka_highs_lows.py, 336–338
sitka_highs.py, 330–336
sleep() function, 272
slices, 61–64
sorted() function, 43–44, 102–103
sort() method, 43
splitlines() method, 186–187
split() method, 196–197
SQLite database, 376–377
square_numbers.py, 58–59
squares.py, 59–60
Stack Overflow, 479
storing data, 201–204. See also JSON files
  saving and reading data, 202–204
strings, 19–25
  changing case, 20
  f-strings, 20–21, 291–292
methods
  lower(), 20
  lstrip(), 22–23
  removeprefix(), 23–24
  removesuffix(), 25
  rstrip(), 22–23
  split(), 196–197
  splitlines(), 186–187
  strip(), 22–23
  title(), 20
  upper(), 20
multiline, 115
newlines in, 21–22
single and double quotes, 19, 24–25
tabs in, 21–22
variables in, 20–21
whitespace in, 21–23
strip() method, 22–23
strptime() method, 333–335
style guidelines, 68–69
  blank lines, 69
  CamelCase, 181
classes, 181
dictionaries, 96–97
functions, 153
if statements, 89
indentation, 68
line length, 69
PEP 8, 68
survey.py, 218
syntax errors, 24
  avoiding with strings, 24–25
  syntax highlighting, 16

T
  tab (\t), 21–22
templates. See under Django
testing code, 209–223
  assertions, 213, 217–218
  failing tests, 214–216
  full coverage, 212
  naming tests, 213
  passing tests, 212–214
  pytest, 209–223
    fixtures, 221–223
    installing, 210–211
    running tests, 213–214
test cases, 212
testing classes, 217–223
testing functions, 211–217
unit tests, 212
test_name_function.py, 212–217
test_survey.py, 220–223
text editors and IDEs. See also VS Code
  Emacs and VIM, 475
  Geany, 474
  IDLE, 474
  Jupyter Notebooks, 475
  PyCharm, 475
  Sublime Text, 474
third-party package, 210
toppings.py, 74, 82–83
tracebacks, 10, 17–18, 192, 195–196
try-except blocks. See exceptions
tuples, 65–67
  defining, 65
  for loop, 66–67
  writing over, 67
type errors, 66

U
  underscore (_)
    in file and folder names, 10
    in numbers, 28
    in variable names, 17
unit tests, 212
user_profile.py, 148–149

V
values() method, 103–104
variables, 16–19, 28
  constants, 28
  as labels, 18–19
  multiple assignment, 28
  name errors, 17–18
  naming conventions, 17
  values, 16
venv module, 374–375
version control. See Git
virtual environments, 374–375
voting.py, 78–80
VS Code, 4–5
  configuring, 470–473
  features, 469–470
  installing
    on Linux, 9
    on macOS, 8
    on Windows, 6
  Python extension, 9–10
  opening files with Python, 185
  Python extension, 9
  running files, 10
  shortcuts, 473–474
  tabs and spaces, 471

W
weather data, 330–341. See also CSV
  files; Matplotlib
while loops, 117–126
  active flag, 120–121
  break statement, 121
  continue statement, 122
  infinite loops, 122–123
  moving items between lists, 124
  quit values, 118
  removing all items from list, 125
whitespace, 21–23. See also strings
Windows
  file paths, 186
  Python
    setting up, 5–6, 9–12, 463–464
    troubleshooting installation, 10
  terminals
    running programs from, 12
    starting Python session, 6
VS Code, installing, 6
word_count.py, 197–199
write_message.py, 190–191
write_text() method, 190–191

Z
Zen of Python, 30–31
ZeroDivisionError, 192–195