INDEX

Symbols
= (assignment) operator, 177
\ (backslash escape character), 190–191
() (call operator), 284
** (combining dictionaries)
  operator, 252
/ (division) operator, 176
// (floor division) operator, 176
# (hash) symbol, 378
\ (line continuation character), 189
% (modulo) operator, 176
+ (plus sign)
  addition operator, 176
  string concatenation operator, 192
** (power) operator, 177
* (splat) operator, 226, 228
* (string replication) operator, 192
- (subtraction) operator, 176
@ (syntactic sugar) symbol, 363–364

A
absolute paths, 319–320
abstraction, 283
accessing
  array attributes, 504–506
  data types, 185
access modes, 325–326
  for binary files, 335
  for shelves, 337
accuracy of prediction in k-NN, 615–616
accuracy_score() method, 615
add_axes() method, 567
adding
  consoles in JupyterLab, 152–153
  images in notebooks, 107–109
  key-value pairs, 251
  list items, 231–232
  text in notebooks, 102–104
addition operator (+), 176
aggregate functions in NumPy, 529, 531–533
aggregation methods in pandas, 650
allocating memory in NumPy, 496
Altair, 429–456
Anaconda
  components of, 7–9
  ecosystem, 397–398
  installing
    on Linux, 12–13
    on macOS, 11–12
    space requirements, 9
    on Windows, 9–11
Anaconda, Inc., 9
Anaconda Navigator
  Community tab, 17–18
  conda environments, 24–34
    backing up, 33
    creating, 25–26
    duplicating, 33
    package management in, 27–33
    removing, 34
  Environments tab, 15–17
  File menu, 18–19
  Home tab, 13–14
  JupyterLab, installing and launching, 140–142
  Jupyter Notebook, installing and launching, 95
  Jupyter Qt console, installing and launching, 51–52
  launching, 13, 25
  Learning tab, 17
  purpose of, 13
  Spyder, installing and launching, 62–63
Anaconda.org, 9
Anaconda Prompt. See CLI
animating plots, 569–573
ANNs (artificial neural networks), 406
anonymous functions. See lambda functions
Apache Spark, 418
append() method, 231–232
apply_along_axis() function, 532
applying style sheets, 578–580
arange() function, 499–501
arguments
in exceptions, 276
of functions, 285–286
positional and keyword, 286–287
Armstrong, Joe, 348
array() function, 497
arrays, 491–492
accessing attributes, 504–506
broadcasting, 526–527
creating, 494–504
arange() function, 499–501
array() function, 497
functions for, 494
linspace() function, 501–502
prefilled, 502–504
describing with dimension and shape, 492–494
flattening, 519–520
incrementing and decrementing, 528
indexing and slicing
Boolean indexing, 515–517
multidimensional arrays, 511–514
1D arrays, 507–509
2D arrays, 509–511
joining, 521–522
matrix dot product, 527–528
printing, 497
purpose of, 492
reading and writing data, 533–535
shaping, 518–519
splitting, 522–524
transposing, 520–521
vectorization, 524–526
artificial intelligence, branches of, 405
artificial neural networks, 406
Artist objects (Matplotlib), 542
asfreq() method, 644
assigning variables, 177, 203–206
assignment operator (=), 177
astimezone() method, 634
attributes, 347
of arrays, accessing, 504–506
defining classes, 350–352
augmented assignment operators, 178
autocompleting text in Editor pane (Spyder), 84
aware objects in datetime module, 633–636
Axes object (Matplotlib), 538–539, 560–561
B
backing up conda environments in Anaconda Navigator, 33
backslash escape character (\), 190–191
base classes, 355
base condition, 297
base environment, 23
Beautiful Soup, 414
Bednar, James, 93, 422
best practices, naming variables, 209–211
bfill() method, 649
binary files, access modes, 335
binary universal functions in NumPy, 530–531
Binder, 129–130, 448
binding. See assigning variables
block comments, 380
blocks of code, 259–260
Bokeh, 430–456
dashboards, 446
geospatial data, 484
Boolean data type, 215
Boolean indexing, 515–517
Boolean operators, 263–264
Bowtie, 446
boxplot() method, 603–604
branching, 258
break statement, 270
broadcasting, 508, 526–527
built-in data types, 184–185
built-in functions, 290–292
built-in modules, 311–313
calling
  functions, 284
  instance methods, 353–355
call operator (), 284
cartograms, 465
Cartopy, 464–465
case sensitivity of variables, 209
catching exceptions when opening files, 342–343
categorical data, converting to numerical data, 610–612
catenary, 542
cells. See specific types of cells (code, Markdown, output, and so on)
chained assignment, 205–206
changing
  runtime configuration parameters, 574–576
  start date, 656
channels, defined, 16
checkpoints in notebooks, 109
choosing
  deep learning frameworks, 408
  geospatial libraries, 484–487
  image manipulation libraries, 410
  natural language processing libraries, 412
  plotting libraries, 450–456
  syntax styles in Jupyter Qt console, 53–54
circular dependencies, 309
class attributes, 350
classes, 347–348
dataclasses, 361–362
  decorators, 362–364
  defining, 365–368
  optimizing, 372–373
  plotting with, 368–370
  post-initialization processing, 370–372
  defining, 349–352, 364–365
docstrings for, 386–387
inheritance, 355–359
instance methods
calling, 353–355
defining, 352–353
instantiating objects, 353–355, 357–358
object control, 359–361
classification problems, 587. See also Palmer Penguins project
class modules, creating, 373–375
cleaning package cache, 48
clearing
  namespace in IPython console (Spyder), 76–77
  workspaces in JupyterLab, 157
CLI
conda environments, 34–48
cleaning package cache, 48
duplicating and sharing, 44–46
list of commands, 35
package management with, 39–44
removing, 47
restoring, 46–47
storage locations for, 37–39
extensions, installing, 169–170
JupyterLab, installing and launching, 142
Jupyter Notebook, installing and launching, 96
Jupyter Qt console, installing and launching, 52–53
launching, 34
Spyder, installing and launching, 65–64
cloning. See duplicating conda environments
close() method, 326
closing
  files, 326, 329–330
  notebooks, 109
  shelves, 338
  workspaces, in JupyterLab, 157
Code Analysis pane (Spyder), 85–86, 391–395
code blocks, 259–260
code cells, 101
  in Editor pane (Spyder), 81–83
  in JupyterLab, 150–152
  in Jupyter Notebook, 104–106
code repositories, sharing notebooks, 125–128
Colab, 131
collection-controlled loops, 264
color blindness, 578
columns, renaming in pandas, 590–591
column_stack() functions, 522
combining
dictionaries, 252
sequences into dictionaries, 248–249
sets, 244
command line interface. See CLI
command mode keyboard shortcuts in Jupyter Notebook, 110–111
command palette in Jupyter Notebook, 112
commands (conda)
environment management, 35
package management, 39
command shell, defined, 2
comments, 210, 377, 378–382
commenting-out code, 381
inline, 380–381
multiline, 380
single-line, 379
Community tab (Anaconda Navigator), 17–18
comparison operators, 214–217
comprehensions, 271–274
computer vision libraries, 409
concrete paths, 320
conda, 8–9
   pip and, 24
   prompt, changing, 38–39
conda configuration file, 38
conda environment manager, purpose of, 22
conda environments
   in Anaconda Navigator, 15–17, 24–34
   backing up, 33
   creating, 25–26
   duplicating, 33
   package management in, 27–33
   removing, 34
   base environment, 23
with CLI, 34–48
   cleaning package cache, 48
   creating, 36–37
   duplicating and sharing, 44–46
   list of commands, 35
   package management with, 39–44
   removing, 47
   restoring, 46–47
   storage locations for, 37–39
   defined, 9
JupyterLab, installing, 140–143
Jupyter Notebook, installing, 94–97
organization of, 22–23
Palmer Penguins project setup, 587–588
purpose of, 21
Spyder, installing, 66–68
conda info command, 23
conda package manager, purpose of, 23
condition-controlled loops, 264
conditions in flow control statements, 259
configuring Spyder interface, 64–66
consoles, adding in JupyterLab, 152–153
constants, 209, 295
container data types
dictionaries, 246–256
list of, 220
lists, 229–238
sets, 239–246
tuples, 220–229
Contextily, 462–463
continue statement, 270
control statements. See flow control statements
converting strings to dates and times, 630–631
coordinated universal time, 633
copying
cells between notebooks in JupyterLab, 158–159
lists, 235–237
copy() method, 236
copy module, 237
correlations, quantifying, 608–609
corr() method, 608–609
counting in DataFrames, 600–601
count() method, 196
countplot() method, 597–601
cross_validate() method, 618
cross-validation, 616–620
cufflinks, 435
current date and time, retrieving, 626–627
current figure in Matplotlib, 546
current working directory, 317–319
custom extensions, creating in JupyterLab, 171
customizing widgets, 121–122

D
Dash, 433, 446–447
dashboards, 445–450
  Dash, 446–447
  Panel, 449–450
  Streamlit, 447–448
  Voilà, 448–449
Dask, 404, 416–418, 441
data
  loading in JSON format, 341
  requesting, 413–418
  saving in JSON format, 340
databases, 336
dataclasses, 361–362
  decorators, 362–364
  defining, 365–368
  optimizing, 372–373
  plotting with, 368–370
  post-initialization processing, 370–372
dataclass module, 361–362
DataFrames, 403, 585–586
  converting categorical data to numerical data, 610–612
  counting in, 600–601
  describing, 596–597
  displaying, 590–591
  duplicates in, 592
  missing values in, 592–596
  quantifying correlations, 608–609
  reindexing, 595–596
data normalization with scikit-learn, 613–615
data serialization
  with json module, 339–342
  with pickle module, 334–336
  pickling vs. JSON, 334
datasets. See also DataFrames; Palmer Penguins project
  exploring, 596–609
  loading, 589–590
  in seaborn, 589
  size of, 451
  for training algorithms, 612–613
Datashader, 441–443
data structures. See arrays
data types
  accessing, 185
  Boolean, 215
  built-in, 184–185
  containers, list of, 219–220
  in datetime module, 627
  dictionaries, 246–256
  floats, 186–189
  integers, 185–186
  lists, 229–238
  in NumPy, 495
  sets, 239–246
  strings, 189–200
  tuples, 220–229
  type casting, 186–187
data visualization
dashboards, 445–450
  Dash, 446–456
  Panel, 449–450
  Streamlit, 447–448
  Voilà, 448–449
geospatial libraries
  Bokeh, 484
  Cartopy, 464–465
  choosing, 484–487
  folium, 470–473
  GeoPandas, 460–464
  Geoplot, 465–467
  GeoViews, 476–479
  ipyleaflet, 473–476
  KeplerGL, 479–481
  list of, 459
data visualization (continued)
geospatial libraries (continued)
Plotly, 467–469
purpose of, 458
pydeck, 481–484
plotting libraries, 419–420
Altair, 429–430
Bokeh, 430–456
choosing, 450–456
Datashader, 441–443
HoloViews, 436–441
for InfoVis, 421
Matplotlib, 422–456, 537–538
Mayavi, 443–445
pandas plotting API, 428–456
Plotly, 431–436
seaborn, 424–456
radial visualization, 605–608
types of, 420–445
data wrangling, 413
date offsets (pandas), 636, 645
date_range() method, 640–642
date ranges in pandas, 640–642
DatetimeIndex class (pandas), 637
datetime() method, 627–628
datetime module, 626
  converting strings to dates and times, 630–631
current date and time, 626–627
data types, 627
durations, 627–628
formatting dates and times,
  628–630
plotting with, 632–633
timestamps, 627–628
time zones, 633–636
date times (pandas), 636
dbm module, 336
Debugger pane (Spyder), 90
debugging, defined, 3, 90
decimal module, 186
decorators, 362–364
decrementing arrays, 528
deepcopy() method, 237
deep learning frameworks, 406–407
default key values, creating, 252–253
default parameters, 287–288
defining
  classes, 349–352, 364–365
code cells
    in Editor pane (Spyder), 81–83
    in JupyterLab, 150–152
    in Jupyter Notebook, 104–106
dataclasses, 365–368
  functions, 284–285
  instance methods, 352–353
def keyword, 284
deleting. See removing
delimiters in split() method, 198
dependencies
  circular, 309
  defined, 8, 23
depth of recursion, 298
derived classes, 355
describe() method, 597
describing DataFrames, 596–597
de-serialization, 334
designing functions, 298–299
dict() function, 248
dictionaries, 246–256
  combining, 252
  combining sequences into,
    248–249
  creating, 247–248
  creating empty, 249
  functions and methods for, 249–250
  key-value pairs
    adding, 251
    removing, 252
  key values
    creating default, 252–253
    retrieving, 251
  printing, 254–255
  retrieving contents, 250–251
  reverse lookups, 253–254
  sorting, 254
dictionary comprehensions, 273–274
difference() method, 243
differences between sets, finding, 243
dimension of arrays, 492–494
directories
  current working directory, 317–319
  for Jupyter notebooks, 98–100,
    144–145
  for Spyder project files, 69–72
directory paths
  absolute and relative, 319–320
  naming, 316
  normalizing, 319
  os module, 317–319
  pathlib module, 320–322
  shutil module, 322–324
dir() function, 306
disabling logging, 280
displaying
  DataFrames, 590–591
  image files in JupyterLab, 153–154
division operator (/), 176
docstrings, 74, 377, 382–395
  for classes, 386–387
  formats for, 392
  for functions and methods, 387–388
  for modules, 384–386
  in Spyder Code Analysis pane, 391–395
  updating, 388–391
doctest module, 388–391
documentation, 377–378
  comments, 378–382
  docstrings, 382–395
  with JupyterLab text editor, 164–165
dot() function, 528
dot notation, 180
dot product for matrices, 527–528
downloading notebooks, 123–125
downsampling time series in pandas, 650–656
drop_duplicates() method, 592
dropna() method, 594
dstack() function, 522
dtypes. See data types
dunder (double underscore) methods, 351, 365
duplicated() method, 592
duplicates
  in datasets, 592
  finding, 243–244
duplicating conda environments
  in Anaconda Navigator, 33
  with CLI, 44–46
durations in datetime module, 627–628
dynamic typing, 184, 211
E
Earth Engine, 483–484edit mode keyboard shortcuts in
  Jupyter Notebook, 111–112
Editor pane (Spyder), 78–84
  autocompleting text, 84
  defining code cells, 81–83
  setting run configuration, 83–84
  writing programs with, 78–81
elif clause, 260–262
else clause, 259, 260–262
embedding widgets, 122
empty dictionaries, creating, 249
empty() function, 503
enabling extensions in Jupyter
  Notebook, 113–115
end-of-line (EOL) markers, 329
enumerate() function, 269
environment files, creating, 44–46
environments. See conda environments
Environments tab (Anaconda
  Navigator), 15–17
error messages, 182–183
errors, ignoring, 277–278
escape sequences, 190–191
events, handling with widgets, 120–121
exceptions
  catching when opening files,
    342–343
  handling, 274–278
    ignoring errors, 277–278
    try and except statements,
    274–276
  raising, 182–183, 274, 276–277
  except statement, 274–276
exploring
  datasets, 596–609
  simulations in JupyterLab, 154–155
exponent operator (**), 177
expressions
  assigning variables, 204
  defined, 176
  generator, 302
  mathematical. See mathematical
    expressions
ternary, 262
extend() method, 231
extensibility, defined, 3
eXtensible Markup Language (XML), 343
Extension Manager, 166–169
extensions
  in JupyterLab
    creating custom, 171
    installing and managing with CLI, 169–170
    installing and managing with Extension Manager, 166–169
  list of, 165–166
  in Jupyter Notebook, 113–115

F
facet plots, 435
ffill() method, 648
Figure object (Matplotlib), 538–539
methods for, 559–560
File menu (Anaconda Navigator), 18–19
files. See also text files
  binary files, accessing, 335
  closing, 326, 329–330
  opening, catching exceptions, 342–343
filling missing values in datasets, 594–595
fillna() method, 594, 650
filter() function, 300
finding
  differences between sets, 243
  duplicates in sets, 243–244
  list index, 233–234
  missing values in datasets, 593
packages
  in Anaconda Navigator, 27–30
  with CLI, 40–42
flattening arrays, 519–520
flatten() method, 519
float() function, 187
floats, 186–189
  converting to/from integers, 186–187
  rounding, 187–189
floor division operator (//), 176
flow control statements, 258
  if statement, 258–264
  loops, 264–274
flow of execution, 258
  functions and, 292–297
  tracing, 278–281
folders. See directories
folium, 470–473
formats for docstrings, 392
formatting
  dates and times, 628–630
  strings, 192–194, 206
for statement, 267–269
frequencies for time series, 641
From Data to Viz website, 456
fromkeys() method, 253
frozenset() function, 245–246
frozensets, creating, 245–246
fruitful functions, 285
f-strings, 192–194, 206
full() function, 503
FuncAnimation class (Matplotlib), 569, 571–573
functions, 283. See also methods
  assigning variables, 204–205
  base condition, 297
  built-in, 290–292
  calling, 284
  creating, 253
  creating arrays
    arange() function, 499–501
    array() function, 497
    linspace() function, 501–502
    list of, 494
    prefilled, 502–504
  default parameters, 287–288
  defining, 284–285
  designing, 298–299
  for dictionaries, 249–250
  docstrings for, 387–388
  flow of execution, 292–297
  generators, 300–303
  global variables, 294–295
  lambda, 299–300
  for lists, 230–231
  main(), 295–297
math module, 179–181
namespaces and scopes, 293–294
naming, 290
in NumPy
  aggregate, 531–533
  pseudorandom numbers, 533
  universal, 529–531
parameters and arguments, 285–286
positional and keyword arguments, 286–287
recursion, 297–298
returning values, 289
for sets, 241–242
for tuples, 222

G

garbage collection, 203
generator expressions, 302
generators, 300–303
Gensim, 412
GeoPandas, 460–464
Geoplot, 465–467
geospatial data, 457–458
geospatial libraries
  Bokeh, 484
  Cartopy, 464–465
  choosing, 484–487
  folium, 470–473
  GeoPandas, 460–464
  Geoplot, 465–467
  GeoViews, 476–479
  ipyleaflet, 473–476
  KeplerGL, 479–481
  list of, 459
  Plotly, 467–469
  purpose of, 458
  pydeck, 481–484
  GeoViews, 476–479
  GeoVis, 420
getcwd() function, 309
get_dummies() method, 600, 611
get() method, 251
Gist, sharing notebooks, 125–128
GitHub, sharing notebooks, 125–128
global scope, 293–294
global statement, 295
global variables, 294–295
globular clusters, 144
Google Earth Engine, 483–484
GridSearchCV class (scikit-learn), 620–622
GridSpec class, creating multipanel displays, 549–555, 564–567
groupby() method, 600

H

handling
  events with widgets, 120–121
  exceptions, 274–278
    ignoring errors, 277–278
    try and except statements, 274–276
hash() function, 240
hash symbol (#), 378
hash tables, 239
HDF5 (Hierarchical Data Format), 343
head() method, 591, 596, 651
heatmaps in seaborn, 608–609
helper libraries, 413–418
Help menu (Jupyter Notebook), 109–110
Help pane (Spyder), 72–74
Heroku, 448
Hierarchical Data Format (HDF5), 343
History pane in IPython console (Spyder), 77
HoloViews, 436–441
HoloViz, 441
home() method, 322
Home tab (Anaconda Navigator), 13–14
hsplit() function, 523
hstack() function, 522
HTML (HyperText Markup Language), 413
HTTP (HyperText Transfer Protocol), 413
hvPlot, 440–441, 462
hyperparameters, 616, 620–622

I

identities of variables, 202–203
IDEs (integrated development environments), defined, 2, 3
idxmax() method, 618, 653
if statement, 258–264
  Boolean operators, 263–264
code blocks, 259–260
elif clause, 260–262
else clause, 259, 260–262
ternary expressions, 262
ignoring errors, 277–278
image files, displaying in JupyterLab, 153–154
image manipulation libraries, 409
images, adding in notebooks, 107–109
immutability of strings, 197
importing
  modules, 179, 304–306
  packages, 589
incrementing arrays, 528
indexes
  of list items, finding, 233–234
  in series, 584–585
indexing
  arrays
    Boolean indexing, 515–517
    multidimensional arrays, 511–514
    1D arrays, 507–509
    2D arrays, 509–511
time series in pandas, 646–647
index() method, 233–234
InfoVis plotting libraries, 420
  Altair, 429–430
  Bokeh, 430–431
  Datashader, 441–443
  HoloViews, 436–441
  list of, 421
  Matplotlib, 422–423
  pandas plotting API, 428–429
  Plotly, 431–436
  seaborn, 424–428
inheritance, 355–359
initialization methods, 351
__init__() method, 351
inline comments, 380–381
in operator, 196
input() function, 213–214
inserting list items, 232
insert() method, 232
insetting plots, 567–568
insignificant variables, 212
inspecting modules, 306–307
installing
  Anaconda
    on Linux, 12–13
    on macOS, 11–12
    space requirements, 9
    on Windows, 9–11
  extensions
    with CLI, 169–170
    with Extension Manager, 166–169
    in Jupyter Notebook, 113
    ipywidgets, 115–116, 170–171
    JupyterLab, 140–143
    with Anaconda Navigator, 140–142
    with CLI, 142
    Jupyter Notebook, 94–97
    with Anaconda Navigator, 95
    with CLI, 96
  Jupyter Qt console
    with Anaconda Navigator, 51–52
    with CLI, 52–53
  packages
    in Anaconda Navigator, 27–30
    with CLI, 40–42
    with conda and pip, 24
  RISE extension, 132–133
  seaborn, 50
Spyder
  with Anaconda Navigator, 62–63
  with CLI, 63–64
  for conda environments and packages, 66–68
instance attributes, 351
instance methods
  calling, 353–355
  defining, 352–353
instantiating objects, 353–355, 357–358
instantiation, 351
integers, 185–186
  converting to/from floats, 186–187
integrated development environments (IDEs), defined, 2, 3
interact class, creating widgets, 116–118
interactive class, creating widgets, 118–119
Interactive Python (IPython), 2
interfaces for Matplotlib, 539–540
object-oriented, 555–557
pyplot, 539–541
internment, 205–206
interpolate() method, 656–657
interpolation in pandas, 656–660
intersection() method, 243–244
int() function, 187
introspection, defined, 4
invoking, objects, 284
ipyleaflet, 473–476
IPython console (Spyder), 74–77
clearing namespace, 76–77
History pane, 77
kernels in, 76
output and plotting, 75
IPython (Interactive Python), 2
IPython notebooks. See notebooks
ipywidgets, installing, 115–116, 170–171
irregular time series in pandas, 656–660
isinstance() function, 185
isnull() method, 595
issubset() method, 245
issuperset() function, 245
items() method, 250–251, 339
iterables, 220

J
joining arrays, 521–522
join() method, 228
jointplot() method, 604–605
Jovian, 131
JSON (JavaScript Object Notation), 334
data
loading, 341
saving, 340
tuples, saving, 341–342
json.dumps() method, 254, 340
json module, 339–342
pickle vs., 334
Jupyter, 3
JupyterDash, 446
Jupyter-gmaps, 476
JupyterHub, 2, 131
JupyterLab, 94
code cells, defining, 150–152
consoles, adding, 152–153
defined, 2
extensions, 3
creating custom, 171
installing and managing
with CLI, 169–170
installing and managing
with Extension Manager, 166–169
list of, 165–166
image files, displaying, 153–154
installing, 140–143
with Anaconda Navigator, 140–142
with CLI, 142
vs. Jupyter Notebook, 140
left sidebar, 147–148
Markdown cells, 149–150
menu bar, 146–147
navigating, 145–146
notebooks
copying cells between, 158–159
creating, 148
naming, 149
opening multiple, 156
sharing, 171
project folders, creating, 144–145
purpose of, 139
simulations, exploring, 154–155
single document mode, 160
synchronized views, creating, 158
text editor
documentation with, 164–165
running scripts in notebooks, 163–164
running scripts in terminal, 162–163
writing scripts, 161–162
widgets, 170–171
workspace
clearing, 157
closing, 157
saving, 156–157
Jupyter Notebook. See also notebooks
command palette, 112
defined, 2
extensions, 113–115
Help menu, 109–110
installing, 94–97
with Anaconda Navigator, 95
with CLI, 96
vs. JupyterLab, 140
keyboard shortcuts, 110–112
navigating, 100–101
Palmer Penguins project setup, 587–588
purpose of, 93–94
widgets, 115
creating manually, 119–120
creating with interact class, 116–118
creating with interactive class, 118–119
customizing, 121–122
embedding, 122
handling events, 120–121
installing ipywidgets, 115–116
Jupyter Notebook Viewer, sharing notebooks, 128–129
Jupyter Qt console
defined, 2
installing and launching
with Anaconda Navigator, 51–52
with CLI, 52–53
as interactive, 53
keyboard shortcuts, 54–55
multiline editing, 58–59
printing and saving, 56–58
purpose of, 49
syntax highlighting, 53
syntax styles, choosing, 53–54
tab and kernel options, 55
kernels
defined, 4
in IPython console (Spyder), 76
in Jupyter Qt console, 55
restarting, 310
keyboard shortcuts
in Jupyter Notebook, 110–112
in Jupyter Qt console, 54–55
keys() method, 250–251, 338
key-value pairs
adding, 251
creating default, 252–253
removing, 252
retrieving, 251
keyword arguments, 286–287
k-NN (k-Nearest Neighbor), 609–622
converting categorical data to numerical data, 610–612
normalizing data, 613–615
optimizing
with cross-validation, 616–620
with GridSearchCV class, 620–622
prediction accuracy, 615–616
running, 615–616
training dataset, 612–613
lambda functions, 299–300
launching
Anaconda Navigator, 13, 25
CLI, 34
JupyterLab
with Anaconda Navigator, 140–142
with CLI, 142
Jupyter Notebook
with Anaconda Navigator, 95
with CLI, 96
Jupyter Qt console
with Anaconda Navigator, 51–52
with CLI, 52–53
Spyder
with Anaconda Navigator, 62–63
with CLI, 63–64
from Start menu, 64

K
KeplerGL, 479–481
Keras, 407–418
kernel density estimation (KDE), 604
Kernel menu (Jupyter Notebook),
checking and running notebooks, 123
lazy evaluation, 300
Learning tab (Anaconda Navigator), 17
left sidebar (JupyterLab), 147–148
len() function, 223–224
length of tuples, 223–224
libraries
Beautiful Soup, 414
for computer vision (image manipulation), 409
dashboards
Dash, 446–447
list of, 446
Panel, 449–450
Streamlit, 447–448
Voilà, 448–449
Dask, 416–418
deep learning frameworks, 406–407
defined, 8
geospatial libraries
Bokeh, 484
Cartopy, 464–465
choosing, 484–487
folium, 470–473
GeoPandas, 460–464
Geoplot, 465–467
GeoViews, 476–479
ipyleaflet, 473–476
KeplerGL, 479–481
list of, 459
Plotly, 467–469
purpose of, 458
pydeck, 481–484
helper libraries, 413–418
Keras, 407
list of, 400
for machine learning, 404–406
for natural language processing, 411–412
NLTK, 411–412
NumPy, 401–418
OpenCV, 409–410
pandas, 403–404
Pillow, 410
plotting libraries, 419–420
Altair, 429–456
Bokeh, 430–456
choosing, 450–456
Datashader, 441–443
HoloViews, 436–441
Matplotlib, 422–456, 537–538
Mayavi, 443–445
pandas plotting API, 428–456
Plotly, 431–436
seaborn, 424–456
PyTorch, 408
for regular expressions, 415–416
requests, 413–414
scikit-image, 410
scikit-learn, 404–407
SciPy, 401
spaCy, 412
statsmodels, 406
SymPy, 402–403
TensorFlow, 407
line continuation character (\), 189
lineplot() method, 619
linspace() function, 501–502
Linux, Anaconda installation, 12–13
list comprehensions, 272–273
list() function, 230, 251
lists, 229–238
adding items to, 231–232
changing item values, 233
checking for membership, 237–238
converting data types to, 230
copying, 235–237
creating, 230
finding index of items, 233–234
functions and methods for,
230–231
inserting items, 232
removing items, 232–233
sorting, 234–235
load_dataset() method, 590
load() function, 534
loading
data in JSON format, 341
datasets, 589–590
localize() method, 634
local scope, 293–294
logging.disable() function, 280
logging levels, 279
logging module, 278–281
loop control statements, 269–271
loops, 264–274
  animating plots, 569–571
break statement, 270
continue statement, 270
for statement, 267–269
pass statement, 271
replacing with comprehensions, 271–274
while statement, 265–267
M
machine learning, 404
  with k-NN, 609–622
libraries, 404–406
macOS, Anaconda installation, 11–12
magic commands
  defined, 57
  list of, 58
magic methods, 351, 365
main() function, 295–297
make_column_transformer() method, 614
maketrans() method, 199
manually creating widgets, 119–120
maps. See geospatial libraries
Markdown cells
  adding images in, 107–109
  adding text with, 102–104
  in JupyterLab, 149–150
markers, 470
mathematical expressions
  assignment operator, 177
  augmented assignment operators, 178
  defined, 176
  mathematical operators, 176–177
  math module, 179–181
  precedence, 178–179
math module, 179–181
Matplotlib, 422–456, 537–538
  with datetime module, 632–633
  interfaces for, 539–540
  object-oriented, 555–557
  pyplot, 539–541
multipanel displays, creating with GridSpec class, 549–555, 564–567
plots
  3D plots, 568–569
  animating, 569–573
creating with object-oriented approach, 557–561
creating with pyplot, 542–545
insetting, 567–568
styling, 573–580
subplots, 545–549, 561–564
terminology for, 538–539
matrix, 493
matrix dot product, 527–528
max() function, 224–225
maximum values of tuples, 224–225
max() method, 653
Mayavi, 443–445
mean() function, 531
melt() method, 619
membership operators, 196
memory allocation in NumPy, 496
menu bar (JupyterLab), 146–147
meshgrid() function, 502
meshgrid() method, 552
Method Resolution Order, 358
methods, 347. See also functions
  for dictionaries, 249–250
docstrings for, 387–388
  for file objects, 326
  initialization, 351
instance methods, 352–355, 353–355
  for lists, 230–231
object-oriented
  for Axes objects, 560–561
  creating plots, 559
  for Figure objects, 559–560
in os module, 317
in pandas
  aggregation methods, 650
  handling missing values, 594
I/O methods, 590
in pathlib module, 321, 332
in pyplot
  creating plots, 543
  manipulating plots, 544
in seaborn
  boxplot(), 603–604
countplot(), 597–601
jointplot(), 604–605
  list of, 598
pairplot(), 601–602
scatterplot(), 602–603
stripplot(), 603–604
for sets, 241–242
in shelve module, 338–339
in shutil module, 323
strings, 196–200
for tuples, 222
mgrid() function, 502
Microsoft Azure Notebooks, 131
min() function, 224–225
minimum values of tuples, 224–225
MinMaxScaler() method, 614
missing values in datasets, 592–596
Modin, 404
modular approach for installation
   JupyterLab, 142–143
   Jupyter Notebook, 96–97
Spyder, 66–68
modules, 283
   built-in, 311–313
   class modules, creating, 373–375
   copy, 237
dataclass, 361–362
datetime, 626–636
dbm, 336
decimal, 186
defined, 8
docstrings for, 384–386
doctest, 388–391
importing, 179, 304–306
inspecting, 306–307
json, 334, 339–342
logging, 278–281
math, 179–181
naming, 310
os, 317–319
pathlib, 320–322, 332–333
pickle, 334–336
purpose of, 303–304
re, 415–416
shelve, 336–339
shutil, 322–324
stand-alone mode, 310–311
string, 199
writing, 307–310
modulo operator (%), 176
moving averages, 660–663
MRO (Method Resolution Order), 358
multidimensional arrays, 493
   indexing and slicing, 511–514
multiline comments, 380
multiline editing in Jupyter Qt console, 58–59
multipanel displays, creating with
   GridSpec class, 549–555,
      564–567
multiple inheritance, 358
multiple notebooks, opening in
   JupyterLab, 156
mutability, 220
   hashability and, 240
tuples and, 227

N
naive approach for installation
   JupyterLab, 140–142
   Jupyter Notebook, 94–96
Spyder, 66
naive objects in datetime module,
   633–636
namespaces
   clearing in IPython console
      (Spyder), 76–77
   in functions, 293–294
naming
   directory paths, 316
   functions, 290
   modules, 310
   notebooks
      in JupyterLab, 149
      in Jupyter Notebook, 101–102
   variables, 206–213
natural language processing, 411–412
Natural Language Tool Kit, 411–412
navigating
   JupyterLab, 145–146
   Jupyter Notebook, 100–101
Navigator. See Anaconda Navigator
nbextensions. See extensions
nbviewer, sharing notebooks, 128–129
ndarray class, 494. See also arrays
nested code blocks, 260
NLP (natural language processing), 411–412
NLTK (Natural Language Tool Kit), 411–412
normalizing
  pathnames, 319
data with scikit-learn, 613–615
notebooks. See also Jupyter Notebook
closing, 109
code cells, defining, 104–106
consoles, adding, 152–153
copying cells between in JupyterLab, 158–159
creating
  in Jupyter Notebook, 100–101
  in JupyterLab, 148
images, adding, 107–109
Markdown cells
  adding images, 107–109
  adding text, 102–104
naming
  in JupyterLab, 149
  in Jupyter Notebook, 101–102
opening multiple in JupyterLab, 156
output cells, 106–107
project folders, creating, 98–100, 144–145
purpose of, 93–94
running scripts in, 163–164
saving, 109
sharing, 122
  via Binder, 129–130
  checking and running from Kernel menu, 123
  with Colab, 131
downloading, 123–125
  via GitHub and Gist, 125–128
  with Jovian, 131
  with JupyterHub, 131
in JupyterLab, 171
via Jupyter Notebook Viewer, 128–129
with Microsoft Azure Notebooks, 131
  trusting, 131–132
as slideshows
  creating, 133–136
  installing RISE extension, 132–133
  speaker notes, 136
synchronized views, creating, 158
text, adding, 102–104
not in operator, 196
now() method, 626–627
np.eye() function, 503
np.savez() function, 534
Numba, 441
numerical data, converting categorical data to, 610–612
NumPy (Numerical Python), 401
arrays
  accessing attributes, 504–506
  broadcasting, 526–527
  creating, 494–504
  describing with dimension and shape, 492–494
  flattening, 519–520
  incrementing and decrementing, 528
  indexing and slicing, 506–517
  joining, 521–522
  matrix dot product, 527–528
  printing, 497
  purpose of, 492
  reading and writing data, 533–535
  shaping, 518–519
  splitting, 522–524
  transposing, 520–521
  vectorization, 524–526
data types, 495
functions
  aggregate, 531–533
  pseudorandom numbers, 533
  universal, 529–531
memory allocation, 496
purpose of, 491–492

O
object-oriented approach (Matplotlib), 555–557
multipanel displays, 564–567
plots, 557–561
objects, 348
  controlling with objects, 359–361
  as instances, 351
  instantiating, 353–355, 357–358
invoking, 284
variables and, 202
ID arrays, 493, 507–509
ones() function, 502
OOP (object-oriented programming), 347
classes
defining, 349–352, 364–365
docstrings for, 386–387
inheritance, 355–359
instance methods, 352–353
instantiating objects, 353–355, 357–358
class modules, creating, 373–375
dataclasses, 361–362
decorators, 362–364
defining, 365–368
optimizing, 372–373
plotting with, 368–370
post-initialization processing, 370–372
object control, 359–361
when to use, 348
OpenCV, 409–410
open() function, 325
opening files, catching exceptions, 342–343
multiple notebooks in JupyterLab, 156
operators
assignment, 177
augmented assignment, 178
Boolean, 263–264
comparison, 214–217
mathematical, 176–177
membership, 196
overloading
in strings, 191–192
in tuples, 227
in variable assignment, 204
precedence, 178–179
optimizing
dataclasses, 372–373
k-NN
with cross-validation, 616–620
with GridSearchCV class, 620–622
os.chdir() method, 318
os.getcwd() method, 317
os module, 317–319
os.normpath() method, 319
os.path.join() method, 318
output cells in notebooks, 106–107
output in IPython console (Spyder), 75
overfitting, 616
overloading operators
in strings, 191–192
in tuples, 227
in variable assignment, 204
P
package cache, 23
cleaning, 48
packages
in Anaconda Navigator, 15–17
conda package manager, purpose of, 23
defined, 8
dependencies, defined, 23
finding
in Anaconda Navigator, 27–30
with CLI, 40–42
importing, 589
installing
in Anaconda Navigator, 27–30
with CLI, 40–42
with conda and pip, 24
managing
in Anaconda Navigator, 27–33
with CLI, 39–44
removing
in Anaconda Navigator, 30–33
with CLI, 42–44
Spyder, installing, 66–68
updating
in Anaconda Navigator, 30–33
with CLI, 42–44
pairplot() method, 601–602
pairplots, 424
Palmer Penguins project
displaying DataFrames, 590–591
duplicates in, 592
exploring dataset, 596–609
importing packages, 589
loading dataset, 589–590
missing values in, 592–596
Palmer Penguins project (continued)
predictions with, 609–622
purpose of, 586–587
renaming columns, 590–591
setup, 587–588
steps in, 587
pandas, 403–404, 583
aggregation methods, 650
alternatives to, 404
DataFrames, 585–586
converting categorical data to numerical data, 610–612
counting in, 600–601
describing, 596–597
displaying, 590–591
duplicates in, 592
missing values in, 592–596
quantifying correlations, 608–609
reindexing, 595–596
datasets, loading, 589–590
plotting API, 428–456
plotting syntax, 620
radial visualization, 605–608
resources for information, 623
series, 584–585
time series, 636–637
changing start date, 656
date offsets, 645
date ranges, 640–642
downsampling, 650–656
indexing and slicing, 646–647
interpolation, 656–660
parsing data, 637–640
resampling, 647–663
sliding window functions, 660–663
time deltas, 644
time spans, 642–644
upsampling, 648–650
Pandas-Bokeh, 431
Panel, 449–450
parallel processing, 417
parameters
default, 287–288
of functions, 285–286
ParaView, 444
parse() method, 630

parsing time series data in pandas,
637–640
pass statement, 271
pathlib module, 320–322
reading and writing text files,
332–333
pathnames, 316
absolute and relative, 319–320
normalizing, 319
os module, 317–319
pathlib module, 320–322
shutil module, 322–324
Pattern, 412
period_range() method, 643
periods (pandas), creating, 642–644
pickled data, shelving, 336–338
pickle.dump() function, 335
pickle.load() function, 335
pickle module, 334–336
pie charts, 553
PIL (Python Image Library), 410
Pillow libraries, 410
pip, conda and, 24
plaintext files, 325. See also text files
Plotly, 431–436, 467–469
Plotly Express, 433–436, 467–469
plots
with datetime module, 632–633
in Matplotlib
animating, 569–573
creating with object-oriented approach, 557–561
creating with pyplot, 542–545
inserting, 567–568
pyplot approach, 539–541
styling, 573–580
subplots, 545–549, 561–564
terminology for, 538–539
3D plots, 568–569
types of, 452
plotting
with dataclasses, 368–370
in IPython console (Spyder), 75
plotting libraries, 419–420. See also geospatial libraries
Altair, 429–456
Bokeh, 430–456
choosing, 450–456
Datashader, 441–443
HoloViews, 436–441
for InfoVis and SciVis, 421
Matplotlib, 422–456, 537–538
Mayavi, 443–445
pandas plotting API, 428–456
Plotly, 431–436
seaborn, 424–456

plt.pie() method, 553
plus sign (+)
   addition operator, 176
   string concatenation operator, 192
Polyglot, 412
pop() method, 232–233, 252
positional arguments, 286–287
postBuild files, 130
__post_init__ function, 370–372
post-initialization processing in
dataclasses, 370–372
power operator (**), 177
precedence, 178–179
predictions with k-NN, 609–622
predict() method, 615
prefilled arrays, creating, 502–504
print() function, 278
printing
   arrays, 497
dictionaries, 254–255
   in Jupyter Qt console, 56–58
tuples, 228–229
processes, 318
Profiler pane (Spyder), 89–90
profiling, defined, 4
project files in Spyder, 68–72
project folders, creating
   in JupyterLab, 144–145
   in Jupyter Notebook, 98–100
Project pane (Spyder), 72
projects, creating in Spyder, 316,
   348–349. See also Palmer
   Penguins project
prompt (conda), changing, 38–39
pseudorandom numbers in NumPy, 533
pure paths, 320
pydeck, 481–484
PyPI (Python Package Index), 24
pyplot, 539–541
   multipanel displays, 549–555
plots, 542–545
subplots, 545–549
Python. See also flow control statements;
functions; libraries; modules;
variables
comments, 210
data types, 184–200
datetime module, 626
   converting strings to dates
   and times, 630–631
current date and time,
   626–627
data types, 627
durations, 627–628
formatting dates and times,
   628–630
plotting with, 632–633
timestamps, 627–628
time zones, 633–636
dictionaries, 246–256
documentation, 377–378
comments, 378–382
docstrings, 382–395
error messages, 182–183
exception handling, 274–278
lists, 229–238
mathematical expressions, 176–181
objects, 202
reserved keywords, 207–209
resources for information, 174
scientific ecosystem, 397–398
sets, 239–246
standard library, 303
tuples, 220–229
Python Data Analysis library.
   See pandas
Python Image Library, 410
Python Package Index, 24
Python package management system.
   See pip
PyTorch, 408
pytz library, 634
Q
Qt console. See Jupyter Qt console
Qt, defined, 4
quantifying correlations, 608–609
quotation marks for strings, 189–190
radial visualization, 605–608
radviz() method, 605–608
raise keyword, 276–277
raising exceptions, 182–183, 274, 276–277
random() function, 504
random numbers in NumPy, 533
range() function, 300
rank of arrays, 492
raster data, 457
ravel() function, 519
raw strings, 191
RcParams class (Matplotlib), 574
reading
array data, 533–535
text files, 325–329
with Pathlib, 332–333
readline() method, 328
readlines() method, 328
read() method, 327
recursion, 297–298
regular expressions (regex), 415–416
reindexing DataFrames, 595–596
relational operators. See comparison operators
relative paths, 319–320
re module, 415–416
remove() method, 233
removing
conda environments
in Anaconda Navigator, 34
with CLI, 47
key-value pairs, 252
list items, 232–233
missing values in datasets, 594–595
packages
in Anaconda Navigator, 30–33
with CLI, 42–44
renaming columns in pandas, 590–591
replace() method, 198
replacing loops with comprehensions, 271–274
requesting data, 413–414
requests library, 413–414
resample() method, 647–650
resampling time series in pandas, 647–663
reserved keywords, 207–209
reset_index() method, 595
reshape() function, 518
reshaping arrays, 518–519
resizing multipanel displays, 554–555
resources for information on Python, 174
restarting kernels, 310
restoring conda environments with CLI, 46–47
retrieving current date and time, 626–627
return address of functions, 286
returning function values, 289
return keyword, 286
reverse lookups, 253–254
RISE extension, installing, 132–133
rmtree() method, 324
rolling() method, 661
round() function, 187–189
rounding floats, 187–189
row_stack() functions, 522
rstrip() method, 329
run configuration, setting in Editor pane (Spyder), 83–84
running
k-NN, 615–616
scripts
in notebooks, 163–164
in terminal, 162–163
runtime configuration parameters, changing, 574–576
Sankey diagrams, 466
save() function, 534
savez_compressed() function, 534
saving
data in JSON format, 340
in Jupyter Qt console, 56–58
notebooks, 109
tuples in JSON format, 341–342
workspaces in JupyterLab, 156–157
scatterplot() method, 602–603
scatterplots, 552
scientific libraries. See libraries
Scientific Python Development IDE.
   See Spyder
scikit-image, 410
scikit-learn, 404–407, 584
   cross-validation, 616–620
   GridSearchCV class, 620–622
   k-NN, 609–622
   normalizing data, 613–615
   prediction accuracy, 615–616
   resources for information, 623
   training datasets, 612–613
SciPy, 401
SciPy stack, 400
SciVis plotting libraries, 420
   Mayavi, 443–445
scopes in functions, 293–294
scripts
   running
      in notebooks, 163–164
      in terminal, 162–163
   writing in JupyterLab text editor,
      161–162
seaborn, 424–456, 584
   datasets
      loading, 590
      practice datasets, 589
   heatmaps, 608–609
   installing, 50
methods
   boxplot(), 603–604
   countplot(), 597–601
   jointplot(), 604–605
   list of, 598
   pairplot(), 601–602
   scatterplot(), 602–603
   stripplot(), 603–604
resources for information, 623
   wide-form and long-form data, 619
seek() method, 327
select_dtypes() method, 612
sequences, 220, 248–249
serialization
   with json module, 339–342
   with pickle module, 334–336
   pickling vs. JSON, 334
series, 584–585
set comprehensions, 274
setdefault() method, 252–253
set() function, 240
sets, 239–246
   combining, 244
   creating, 239–241
   differences between, 243
   duplicates in, 243–244
   frozensets, 245–246
   functions and methods for,
      241–242
   supersets, 245
shape() function, 519, 520
shape of arrays, 492–494
shaping arrays, 518–519
shared package cache, 23
sharing
   conda environments
      with CLI, 44–46
   notebooks, 122
      via Binder, 129–130
      checking and running from
         Kernel menu, 123
      with Colab, 131
      downloading, 123–125
      via GitHub and Gist, 125–128
      with Jovian, 131
      with JupyterHub, 131
      in JupyterLab, 171
      via Jupyter Notebook Viewer,
         128–129
      with Microsoft Azure
         Notebooks, 131
      trusting, 131–132
   shell utilities (shutil) module, 322–324
   shelve module, 336–339
   shelve.open() method, 336
   shelves, closing, 338
   shelving pickled data, 336–338
   shutil module, 322–324
   simulations, exploring in JupyterLab,
      154–155
   single document mode
      (JupyterLab), 160
   single-line comments, 379
   size of datasets, 451
slicing
arrays
  multidimensional arrays, 511–514
  1D arrays, 507–509
  2D arrays, 509–511
strings, 194–196
time series in pandas, 646–647
slideshows, notebooks as
  creating, 133–136
  installing RISE extension, 132–133
  speaker notes, 136
sliding window functions, 660–663
__slots__
  class variable, 372–373
sorted() function, 254
sorting
  dictionaries, 254
  lists, 234–235
sort() method, 234–235
spaCy, 412
spatial indexing, 463
speaker notes for slideshows, 136
special consoles in Spyder, 77
special methods, 351, 365
specifications files, creating, 46
spines, 557
splat (*) operator, 226, 228
split() function, 523
SplitMap, 474
split() method, 198
splitting arrays, 522–524
Spyder
  Code Analysis pane, 85–86, 391–395
  configuring interface, 64–66
  Debugger pane, 90
  defined, 2
  Editor pane, 78–84
    auto-completing text, 84
    defining code cells, 81–83
    setting run configuration, 83–84
    writing programs with, 78–81
  Help pane, 72–74
  installing
    with Anaconda Navigator, 62–63
    with CLI, 63–64
for conda environments and
  packages, 66–68
IPython console, 74–77
  clearing namespace, 76–77
  History pane, 77
  kernels in, 76
  output and plotting, 75
launching
  with Anaconda Navigator, 62–63
  with CLI, 63–64
  from Start menu, 64
Profiler pane, 89–90
project files and folders, 68–72
Project pane, 72
projects, creating, 316, 348–349
purpose of, 61
special consoles, 77
  Variable Explorer pane, 86–89
SQLite, 343
stacking, arrays, 521–522
stack overflow, 298
stand-alone mode for modules, 310–311
standard library (Python), 303
start date, changing, 656
Start menu (Spyder), launching, 64
statements, 177
static typing, 184
statistical methods in NumPy, 531–533
statsmodels, 406
storage locations for conda
  environments, specifying, 37–39
Streamlit, 447–448
strftime() method, 628–630
str() function, 190
string concatenation operator (+), 192
string module, 199
string replication operator (*), 192
strings, 189–200. See also text
  converting to dates and times, 630–631
  escape sequences, 190–191
  formatting, 192–194, 206
  immutability of, 197
  interning, 205
  membership operators, 196
methods, 196–200
operator overloading, 191–192
quotation marks for, 189–190
raw, 191
slicing, 194–196
stripplot() method, 603–604
stripplots, 425
strptime() method, 631
structured arrays, 535
style files, creating, 576–578
style sheets, applying, 578–580
style.use() method, 577
styling plots, 573–580
runtime configuration parameters, 574–576
style files, 576–578
style sheets, 578–580
subclasses, 355
subplot() method, 545–549
subplot_mosaic() method, 566
subplots in Matplotlib
  object-oriented approach, 561–564
  pyplot, 545–549
subplots() method, 561–564
subtraction operator (-), 176
superclasses, 355
super() function, 358–359
supersets, 245
swapaxes() function, 521
swapping array axes, 520–521
SymPy, 402–403
synchronized views, creating in
  JupyterLab, 158
syntactic sugar, 363–364
syntax highlighting, 53
syntax styles, choosing in Jupyter Qt
  console, 53–54
T
tabs in Jupyter Qt console, 55
TensorFlow, 407
tensors, 493
terminal
  defined, 4
  running scripts in, 162–163
terminal window. See CLI
ternary expressions, 262
text. See also strings
  adding in notebooks, 102–104
  autocompleteing in Editor pane
    (Spyder), 84
TextBlob, 412
text editor, JupyterLab
  documentation with, 164–165
  running scripts
    in notebooks, 163–164
    in terminal, 162–163
  writing scripts, 161–162
text editor, Spyder. See Editor pane,
  (Spyder)
text files
  closing, 329–330
  reading, 325–329
    with pathlib, 332–333
  writing to, 330–331
    with pathlib, 332–333
  text() method, 551
threads, 417
3D arrays. See multidimensional arrays
3D plots, 568–569
thresholding, 515
tight_layout() method, 547, 554
tile maps, 463
timedelta object, 627–628
timedelta_range() method, 644
time deltas (pandas), 636, 644
time series
datetime module, 626
  converting strings to dates
    and times, 630–631
  current date and time,
    626–627
  data types, 627
  durations, 627–628
  formatting dates and times,
    628–630
  plotting with, 632–633
  timestamps, 627–628
  time zones, 633–636
pandas, 636–637
  changing start date, 656
  date offsets, 645
  date ranges, 640–642
downsampling, 650–656
time series (continued)
pandas (continued)
    indexing and slicing, 646–647
    interpolation, 656–660
    parsing data, 637–640
    resampling, 647–663
    sliding window functions, 660–663
    time deltas, 644
    time spans, 636, 642–644
    upsampling, 648–650
    time spans (pandas), 636, 642–644
    Timestamp class (pandas), 637
    timestamps
        in datetime module, 627–628
        in pandas, 637
    time zones
        in datetime module, 633–636
        in pandas, 639–640
    to_datetime() method, 637–640
    traceback, 182
    tracing flow of execution, 278–281
    training datasets, 612–613
    train_test_split() method, 612–613
    translate() function, 199
    transpose() function, 520
    transposing arrays, 520–521
    trusting notebooks, 131–132
    try statement, 274–276
tuple() function, 221–222
tuples, 220–229
    converting data types to, 221–222
    creating, 221
    functions and methods for, 222
    length of, 223–224
    minimum and maximum values, 224–225
    mutability and, 227
    operator overloading, 227
    printing, 228–229
    saving in JSON format, 341–342
    unpacking, 225–227
    2D arrays, 493, 509–511
    type annotations, 366–367
    type casting, 186–187
    type() function, 185
    type hints, 184, 366–367
tz_convert() method, 639
tz_localize() method, 639
U
    unary universal functions in NumPy, 529–530
    Unicode, 190
    union() method, 244
    universal functions in NumPy, 529–531
    unpacking tuples, 225–227
    update() method, 251
    updating
        docstrings, 388–391
        packages
            in Anaconda Navigator, 30–33
            with CLI, 42–44
    upsampling time series in pandas, 648–650
    user input, 213–214
    UTC (coordinated universal time), 633
V
    Vaex, 404
    value_counts() method, 659
    values() method, 250–251
    Variable Explorer pane (Spyder), 86–89
    variables, 201
        assigning, 177, 203–206
        comparison operators, 214–217
        global, 294–295
        identities of, 202–203
        insignificant, 212
        naming, 206–213
        for user input, 213–214
    vector data, 457
    vectorization, 524–526
    vectors, 493
    visualizing data
        dashboards, 445–450
        Dash, 446–447
        Panel, 449–450
        Streamlit, 447–448
        Voilà, 448–449
    geospatial libraries
        Bokeh, 484
        Cartopy, 464–465
choosing, 484–487
folium, 470–473
GeoPandas, 460–464
Geoplot, 465–467
GeoViews, 476–479
ipyleaflet, 473–476
KeplerGL, 479–481
list of, 459
Plotly, 467–469
purpose of, 458
pydeck, 481–484
plotting libraries, 419–420
Altair, 429–430
Bokeh, 430–431
choosing, 450–456
Datashader, 441–443
HoloViews, 436–441
for InfoVis, 421
Matplotlib, 422–456, 537–538
Mayavi, 443–445
pandas plotting API, 428–456
Plotly, 431–436
seaborn, 424–456
radial visualization, 605–608
types of visualizations, 420–445
void functions, 285
Voilà, 448–449
vsplit() function, 523
vstack() function, 522

W
Wang, Peter, 443
web scraping, 413
while statement, 265–267
whitespace in mathematical expressions, 179
widgets, 115
creating
with interact class, 116–118
with interactive class, 118–119
manually, 119–120
customizing, 121–122
embedding, 122
handling events, 120–121
installing ipywidgets, 115–116
in JupyterLab, 170–171
Windows, Anaconda installation, 9–11
with statement
closing files, 329–330
closing shelves, 338
workspaces (JupyterLab)
clearing, 157
closing, 157
saving, 156–157
writelines() method, 331
write() method, 330
writing
array data, 533–535
with Editor pane (Spyder), 78–81
modules, 307–310
scripts in JupyterLab text editor, 161–162
to text files, 330–331
with pathlib, 332–333

X
XML (eXtensible Markup Language), 343

Y
YAML, 44, 343
yield statement, 301

Z
zero() function, 502
zip() function, 248