

INDEX

Note: *Page numbers in italics refer to figures or tables*

Symbols

- > (angle bracket), 8–9
- * (asterisk), 122–123, 154
- @ (at sign), 226–227
- \ (backslash), 6
- : (colon), 119, 289
- { } (curly brackets), 33
- \$ (dollar sign), 7
- = (equality operator), 35
- = (equal sign), 29
- / (forward slash), 6, 122–123
- > (greater than) operator, *149*
- >= (greater than or equal to) operator, *149*
- # (hash mark), 29
- < (less than) operator, *149*
- <= (less than or equal to) operator, *149*
- != (not equal to) operator, *149*
- < > (not equal to) operator, *149*
- operator, 122
- % (percent sign), 149
- + (plus sign), 122–123
- ; (semicolon), 141
- ~ (tilde), 7
- """ (triple quotes), 29
- _ (underscore), 149, 245

A

- accessor property, 124
- ACID (Atomicity, Consistency, Isolation, Durability), 135
- add() function, 122–123
- ADD keyword, 144
- add_trace() method, 193, 195, 209
- advanced navigation, 20–21
- aggregations, 128, 154–155
- AI (artificial intelligence), xxv

- alias, 112
- Altair, 188
- ALTER, 140
- ALTER TABLE command, 144
- Amazon Aurora, *137*
- Amazon Web Services (AWS),
 - xxx, 136
- American Standard Code for Information Interchange (ASCII), 33
- any() function, 275
- API (application programming interface)
 - authentication handling, 176–180
 - breaking large requests into smaller batches, 174–176
 - defined, 162
 - error handling, 171–174
 - errors, 173–174
 - GraphQL, 163
 - overview, 161–162
 - REST, 162–179
 - SOAP, 163
 - streamlining requests with reusable functions, 180–183
 - WebSocket, 163
- API keys
 - making requests with, 178–179
 - storing in environment variables, 177–178
- append() method, *40*
- application-specific environment variables, 10
- app.py file, 216
- app variable, 216
- arguments, 5, 39, 60–61
 - adding, 170–171
 - keyword, 61
 - positional, 61
 - system, 72–73

- arithmetic operators, 122–123.
 - See also* operators
- artificial intelligence (AI), xxv
- asana library, xxx
- ASCII (American Standard Code for Information Interchange), 33
- AS keyword, 147–148
- assembly language, 26
- assets, CSS, 222
- assets/styles.css* file, 229
- assignment statements, 29–30
- `astimezone()` method, 48
- `astype()` method, 116
- at sign (@), 226–227
- attributes, 27
- authentication. *See also* API
 - with API keys, 177–179
 - handling, 176–180
 - with OAuth, 179–180
- automation, xxviii–xxx
- AVG function, 155
- AWS (Amazon Web Services), xxx, 136
- Azure, 214

B

- backslash (\), 6
- bar charts. *See also* charts
 - basic, 194–196
 - multi-trace, 196–198
- bash, 5, 11, 12
- batches, 174
- BETWEEN operator, 149
- `bgcolor` subproperty, 204
- bin* subdirectory, 19
- Bitbucket, 78
- block elements, 217–218, 217*t*
- blocks, 28
- Bloomberg API, xxxi
- BLS API, 241
- `BLSDataset` class, 251–252
 - lazy loading, 248
 - unemployment object, 243
 - `yearly_summary()` method, 244
- blue screen of death, xxxi
- Bokeh, 188
- BOOLEAN data type, 144*t*
- Boolean indexing, 119

- Boolean variables, 31
- `bordercolor` subproperty, 204
- `borderwidth` subproperty, 204
- Boto3, xxx
- boundaries, 13
- branches, 87–88
- break keyword, 57–58
- breakpoints, 21, 258–259
 - conditional, 264
 - hit counts, 266
- `b` subproperty, 204

C

- callbacks, 226–227
- call stack, 262
- Cascading Style Sheets. *See* CSS
- catching exceptions, 172
- categorical data, 116
- `cd` (change directory) command, 6, 8
- `celsius_to_fahrenheit()` function, 280
- central processing unit (CPU), 26
- characters, 32–33
- charts
 - bar charts, 194–198, 196, 197
 - colors, 205
 - common properties, 202–204
 - Excel vs. Python, 186–187
 - fonts and font families, 202–205
 - layout properties, 201–205
 - line charts, multi-trace, 193–194, 194
 - multidimensional data, 198–200, 200
 - scatter plots, 198–200, 200
 - themes, 205
- Chinese characters, 33
- classes, 53, 236
 - constructors, 237–238
 - custom, 237–239
 - encapsulation with, 245
 - extendability with, 244–245
 - inheritance, 249–250, 253–254
 - methods, 238–239
 - parent class, 251–252
 - in Python libraries, 237
 - reusability with, 243–244
 - superclass, 249
- class selectors, 221
- class variables, 261
- CLI (command line interface), 4–5, 69

- clipboard, 117
- cloning repositories, 83–84
- co-authoring, xxiv
- code, 26
 - completion, 20
 - pausing, 265–266
 - running, 17–18
 - testing, 266–272
 - assertions, 268
 - test cases, 271–272
 - unittest module, 269–271
- code cells, 98–99
- code editor, 19–21
- coding habits, 273–283
 - to avoid, 274–279
 - improving code, 282–283
 - iterative development and, 282
 - managing code, 279–282
 - modularization and, 279–282
 - overview, 273–274
 - premature optimization and, 282
 - refactoring and, 282
 - simplicity over complexity, 274–278
- collaboration, xxiv
- colon (:), 119, 237
- color property, 203
- column transformations, 154
- command line interface (CLI), 4–5, 69
- Command Prompt (CMD), 5, 69
- commands, 5
- comma-separated values (CSV),
 - 67–69, 167
- commits, 85
- comparison operators, 35, 149.
 - See also* operators
- component_id property, 227
- component property, 227
- composability, 254
- composite data types, 36–38
- concatenation, 126
- concat() function, 126
- conditional breakpoints, 264
- conditionals, 41–45
 - elif clauses, 43–44
 - else clauses, 43
 - if statements, 42–43
 - nested, 44–45
- conditions, 59
- conflicts, 88–90
- ConnectionError, 174
- console, 5
- constructors, 237–238
- containers, 27
- continue keyword, 59
- control flow, 41
- copies, 120
- Copilot, xxv, 20
- corporate IT policies, working within,
 - 13–14
- COUNT function, 155
- COUNTIFS, 129
- COVID test results, xxi–xxii
- C programming language, 32
- CREATE command, 140, 142
- CSS (Cascading Style Sheets)
 - assets, 222
 - basics, 221–222
 - classes, 221
 - in Dash, 222–223
 - ID selectors, 222
 - overview, 216–217
 - properties, 221
 - selectors, 221
- CSV files, 67–69, 167
- csv module, 66–73

D

- Dash
 - basics of, 214–216
 - creating interactive reports with,
 - 212–214
 - dashboard, 215–216
 - dependencies submodule, 227
 - installing, 215
 - Plotly charts in, 223–227
 - vs. Power BI, 214
 - web development concepts and,
 - 215–223
 - CSS, 221–223
 - HTML, 217–220
 - overview, 216–217
- data
 - aggregating, 128, 154–155
 - categorical, 116
 - grouping, 129–130
 - manipulating, 121–124

- data (*continued*)
 - missing, handling, 124–125
 - temporal, 116
- database management systems, 137–138
- databases
 - adding value to Excel workflow
 - with, 134
 - atomicity, 135
 - consistency, 135
 - corporate, accessing, 136
 - costs of connection, 136–137
 - creating, 138–142
 - document-oriented, 137
 - durability, 135
 - incorporating into workflow, 135
 - location of, 135–136
 - managing, 140–141
 - nonrelational, 137
 - NoSQL, 137
 - relational, 137
 - remote, connecting to, 142
 - retrieving data, 146–156
 - tables, 143–145
 - task automation with Python and SQL, 157–160
- DataFrames, 110–116
 - accessing values, 118–119
 - arithmetic operations, 122–123
 - combining, 125–129
 - creating, 112–114
 - date operations, 123–124
 - handling missing data, 124–125
 - moving data to Excel from, 117
 - reading data from files into, 114–115
 - string operations, 123–124
- datasets
 - for Plotly charts, 189–191
 - sorting, 121
- data types
 - basic, 31–34
 - composite, 36–38
 - importing multiple, 46
 - None, 34
 - SQL, 143–144, 144
- date object, 46
- date operations, 123–124
- dates, 45–47
- datetime library
 - format codes and Excel
 - counterparts for, 47
 - library, 46–47
 - objects, converting to/extracting
 - from string, 47–49
 - working with time zones, 49–50
- datetime64 type, 116
- DATEVALUE function, 45
- dcc.Checklist() component, 226, 228
- dcc.Graph() component, 224, 228
- Debug Console, 262
 - Step Into command debugger
 - command, 264–265
 - Step Out command debugger
 - command, 263–264
- debugger, 21
- debugging, 257–258
 - code with errors, 263–265
 - with different inputs, 266–267
 - integrated, 21
 - Step Into command, 264–265
 - Step Out command, 263–264
 - in VS Code, 258–267
- Debug Panel, 260–262
- decorator, 226–227
- def keyword, 60
- Desktop directory, 8–9
- dictionaries, 38
- dictionary methods, 41
- dir command, 8, 9, 10
- directed acyclic graph (DAG), 92–93
- directories, 5
 - creating, 8
 - deleting, 8–9
 - listing, 8
 - navigating, 6, 7–9
 - paths, 6
- DIV/0! error, xxiii–xxiv
- div element, 217–218
- div() function, 122–123
- docstrings, 29
- documentation, xxiii
- document-oriented databases, 137
- dollar sign (\$), 7
- dot notation, 39
- download() method, 239–243, 244–245, 249, 252

- dragmode property, 209
- Dropbox, 78
- dropbox-sdk-python, xxx
- DROP keyword, 144
- dropna() function, 125
- DROP TABLE command, 145
- dt.strftime() function, 124
- dtypes property, 116

E

- edge cases, 271
- element sectors, 221
- elif clauses, 43–44
- else clauses, 43
- emojis, 33
- encapsulation, 245
- encoding, 33
- encryption, 164
- environment variables, 10, 177–178
 - commands, 11
 - modifying on macOS, 11–12
- errors
 - common, 173–174
 - detecting, xxii–xxiii, 20
 - handling, xxii–xxiii, 171–174
- Excel
 - aggregation operations in, 128
 - charting in, 186–188
 - collaboration in, xxiv
 - comparison operations in, 35*t*
 - error handling in, xxii–xxiii
 - learning to code in Python with, xx
 - logical operations in, 35–36
 - moving between pandas and, 116–118
 - as productivity tool, xix
- excel_utils.py* file, 70
- except blocks, 181
- exceptions, 172
- execute() method, 159
- extendability, 244–245

F

- family property, 203
- feature branches, 87–88
- fetchall(), 159
- fetchmany(), 159
- fetchone(), 159

- ffill() function, 125
- fig.update_layout() function, 193, 224
- filesystem navigation commands, 6
- fillna() function, 125
- filters, 148
- FIND function, 45
- firewalls, 6, 14, 142
- floats, 31–32
- folders, 5
- font property, 202–203
- fonts, 202–205
- font subproperty, 204
- for loop, 55–56. *See also* loops
- FROM clause, 151
- f-strings, 33
- full outer joins, 153–154
- functional programming, 59–64, 239
- function definitions, 53
- functions, 38
 - components of, 60–63
 - higher-order, 227
 - modifier, 64
 - modularity of, 60, 65
 - in Python, 39
 - reusable, 60, 180–183
 - simplification with, 60
 - user-defined, 59–64
- function variables, 261

G

- get() method, 41, 171, 174
- GET request, 165
- getters, 246–247
- Git, 77–78
 - best practices, 90–91
 - branches, 87–88
 - commits, 85
 - configuring, 82–83
 - conflicts, 88–89
 - diagnostic commands, 91
 - directed acyclic graph (DAG)., 92–93
 - file structure of Python
 - project in, 80
 - vs. GitHub, 78
 - hashing, 93
 - installing, 82
 - integrating with VS Code, 92

Git (*continued*)

- managing histories in, 87–90
- repositories, 78–87
 - centralizing work in single folder, 78–80
 - cloning, 83–84
 - creating from scratch, 83
 - local, 80–81
 - maintaining *.gitignore* file, 81–82
 - remote, 80–81
 - setting up, 81–84
 - syncing, 86–87
- staging area, 84–85
- tracking changes to files, 84–85
- troubleshooting with status messages, 91–92
- `git blame` command, 91
- `git branch` command, 91
- `git commit` command, 85
- `git diff` command, 91
- `git fetch` command, 85
- .git* folder, 80
- GitHub, 78
- GitHub Copilot, xxv, 20
- .gitignore* file, 80, 81–82
- GitLab, 78
- `git log` command, 91
- `git pull` command, 85
- `git push` command, 85–86
- `git show` command, 91
- `git status` command, 91
- global variables, 54
- `go.Bar()` function, 195, 224
- `google-api-python-client`, xxx
- Google Colab, 14
- Google Drive, 78
- GraphQL APIs, 163
- greater than (>) operator, 149
- greater than or equal to (>=) operator, 149
- `gridcolor` subproperty, 203
- `gridwidth` subproperty, 203
- gross domestic product (GDP), xxi
- `GROUP BY` clause, 155–156
- `groupby()` function, 129–130
- grouping, 129–130, 155–156

H

- hashing, 93
- hash mark (#), 29
- hash tables, 38, 93
- `HAVING` clause, 156
- `HAVING` keyword, 155
- headers, 178–179
- `head()` method, 115, 190
- hexadecimal codes, 205
- higher-order functions, 227
- high-level languages, 26, 27
- hit counts, 266
- horizontal concatenation, 126
- `hovermode` property, 209
- `hovertemplate` argument, 210
- HTML (hypertext markup language)
 - basics, 217–219
 - block-level elements, 217–218, 217
 - in Dash, 219–220
 - inline level elements, 218–219, 219
 - overview, 216–217
 - tags, 217*t*
- HTTP (Hypertext Transfer Protocol), 163
- `HTTPError`, 174
- HTTPS (Hypertext Transfer Protocol Secure), 164
- `hubspot-api-client`, xxxi

I

- IBM Db2, 137
- iCloud, 81
- IDE (integrated development environment), 21
- ID selectors, 222
- `if...else` statement, 266–267
- `IF` function, 42–43
- `if` statements, 42–43
- `iloc[]`, 119
- `imports`, 53
- inconsistency in coding, 277
- `INDEX/MATCH`, 127
- indexes, 37, 110
- inheritance, 249–251
- `__init__()` method, 237, 242
- inline elements, 218–219, 219
- inner joins, 150–151
- `IN` operator, 149
- `Input()` parameter, 227

- INSERT INTO command, 145
- insert() method, 40
- installing
 - Dash, 215
 - JupyterLab, 101
 - pandas, 111–112
 - Plotly, 189
 - PostgreSQL, 138–140
 - Psycopg 2, 158
 - Python, 14–18
 - VS Code, 21–23
- instances, 236
- INTEGER data type, 144
- integers, 31–32
- integrated development environment (IDE), 21
- interactive applications, 27
- interactive reports
 - creating with Dash, 213–214
 - reasons for using, 212–213
 - sharing, 230–231
- intermediate-level languages, 26–27
- Internet Protocol (IP), 164
- interoperability, xxx–xxxi
- interpreter, 4, 22–23
- int() function, 39
- InvalidURL error, 174
- .ipynb file, 102, 106
- isin() method, 224
- isna() function, 125
- IS NOT NULL operator, 148
- IS NULL operator, 148
- items() method, 41
- iterations, 54, 282

J

- JavaScript, 216–217
- JavaScript Object Notation (JSON), 167–169
- JOIN keyword, 150
- joins, 148–149. *See also* databases
 - full outer, 153–154
 - inner, 150–151
 - left, 151–153
 - right, 151–153
- JPMorgan Chase, xxi
- JSONDecodeError, 174
- json.dump() function, 168

- json.dumps() function, 168
- JSON files, 167–169
- json.load() function, 168
- json.loads() function, 168
- json() method, 170
- JSONPlaceholder, 166
- JupyterLab, 101–102, 102
- Jupyter Notebooks
 - building, 103–105
 - closing, 103
 - code cells, 98–99
 - code to put in, 106
 - creating, 102–103
 - markdown cells, 99–100, 103–104
 - pandas and, 111
 - raw text cells, 100–101
 - renaming, 103
 - using Plotly in, 189
 - version control, 107

K

- Kaleido library, 197
- key, 10
- KeyError, 173
- keys() method, 41
- key-value pairs, 38
- keyword arguments, 61

L

- large language models (LLMs), xxv, 20
- launch.json file, 266–267
- lazy loading, 248–249
- LEFT function, 45
- left joins, 151–153
- legend property, 204
- len() function, 39
- less than (<) operator, 149
- less than or equal to (<=) operator, 149
- libraries, 18. *See also* pandas; Plotly
 - Altair, 188
 - asana, xxx
 - Bokeh, 188
 - charting, 188
 - csv, 66–67
 - Dash, 215, 216
 - datetime, 46–47
 - JSON, 168, 169
 - Kaleido, 197

- libraries (*continued*)
 - Matplotlib, 188
 - NumPy, 110, 116, 237
 - openai, xxx
 - os module, 178–179
 - Psychopg 2, 157, 158, 160
 - PyDataset, 190, 223, 228
 - Python Standard Library, 5
 - pytz, 48
 - Requests, 169–171
 - Seaborn, 188
 - sys module, 72
 - timezone, 49–50
 - unittest, xxiii, 268, 269–271, 272
 - xlwings, 67
- LIKE operator, 149
- LIMIT clause, 147
- line charts
 - basic, 192
 - multi-trace, 193–194
 - simple, 191–192
- linecolor subproperty, 201, 203
- linewidth subproperty, 203
- list comprehension, 56–57
- list methods, 40
- lists, 36–37, 55
- loc[], 119
- localize() method, 48
- logic, functionalizing, 65–66
- logical operators, 35–36.
 - See also* operators
- Looker, 213
- loops, 54–59
 - control keywords for, 57–59
 - for, 54–55
 - while, 59
- low-code tools, 213
- lower() method, 40
- low-level languages, 26
- ls command, 6, 9
- l subproperty, 204

M

- machine code, 26
- macOS. *See also* Windows
 - directory navigation on, 7–8
 - environment variable
 - commands for, 11

- file manipulation on, 9
- file paths, 6*t*
- filesystem navigation commands, 6
- installing PostgreSQL on, 139
- modifying environment variables
 - on, 11–12
- troubleshooting, 17
- main block, 71–72
- main branches, 87
- major versions, 15
- make directory (mkdir) command, 6, 8
- makedirs() function, 66
- manual execution time, xxviii–xxx
- margin property, 204
- MariaDB, 137
- markdown cells, 99–100, 103–104
- master branches, 87
- Matplotlib, 188
- max() function, 139
- MAX function, 155
- mean() function, 128, 129
- merging, 88, 127
- methods, 39–41, 236, 238–239.
 - See also specific methods*
- Microsoft SharePoint, 81
- micro versions, 15
- MID function, 45
- MIN function, 155
- minor versions, 15
- mixed scopes in coding, 276–277
- mkdir (make directory) command, 6, 8
- modifier functions, 64
- modularization, xxviii, 279–282
- modules, importing scripts as, 70–71
- MongoDB, 137
- mul() function, 122–123
- multiplication operator (*), 154
- multi-trace bar chart, 196–198, 197
- multi-trace line chart, 193–194
- mutable data structures, 64
- MySQL, 137

N

- NA marker, 124–125
- NaN marker, 125
- nano (file editor), 12
- negative indexing, 37
- nested conditionals, 44–45, 275

- no-code tools, 213
- None data type, 33
- non-English writing systems, 33
- None return value, 63
- nonrelational databases, 137
- NoSQL databases, 137
- not equal to (`!=`) operator, 148
- `NotImplementedError`, 250
- `notna()` function, 125
- NULL data type, 144
- NUMERIC data type, 144
- NumPy, 110

O

- O365, xxx
- OAuth, 179–180
- object-oriented programming (OOP)
 - encapsulation, 244
 - extendability, 244–245
 - vs. functional programming, 239
 - getters, 246–247
 - inheritance, 249–251
 - lazy loading, 248–249
 - overview, 235–236
 - properties, 245–246
 - reusability, 243–244
 - setters, 247–248
- objects, 27–28, 236
- OneDrive, 78
- ON keyword, 150–151
- OpenAI, xxx
- openai library, xxx
- Open Authorization (OAuth), 179–180
- openpyxl, 118
- OpenWeather API, 177–178, 181
- operators
 - arithmetic, 34–35
 - comparison, 35
 - defined, 33
 - logical, 35–36
- optimization, 282
- Oracle, 137*t*
- ORDER BY clause, 148
- orientation subproperty, 204
- os module, 66–73
- `Output()` parameter, 227

P

- pad subproperty, 204
- pagination, 174
- pandas library, 109–110
 - aggregation operations in, 128
 - charting functionality, 188
 - data analysis with, 110–111
 - DataFrames data types, 110–111, 115–116
 - installing, 111–112
 - Jupyter Notebooks and, 111
 - missing data operations in, 125
 - moving between Excel and, 116–118
 - pd as alias for, 112
 - reshaping data with, 129–131
 - Series data types, 110–111, 115–116
 - sorting datasets in, 121
- `paper_bgcolor` property, 204
- `params` argument, 171, 180–181
- parent class, 251–253
- PATH environment variable, 15, 17
- `path.exists()` function, 66
- paths, 6
- `pd.concat()` function, 126
- `pd.DataFrame()` function, 113
- `pd.merge()` function, 127
- `pd.read_clipboard()` function, 117
- `pd.read_csv()` function, 114–115, 118
- `pd.read_excel()` function, 115, 118
- `pd.read_json()` function, 115
- percent sign (%), 149
- pg8000, 158
- pip (pip installs packages), 18–19
- pip command, 18, 190, 215
- pivoting, 130–131
- `pivot()` method, 130–131
- `plot_bgcolor` property, 204
- Plotly, 188
 - charting multidimensional data with, 198–200, 200
 - creating simple charts with, 189–198
 - bar chart, 194–196, 196
 - datasets for, 189–191
 - line chart, 191–192, 192

Plotly (*continued*)

creating simple charts (*continued*)

multi-trace bar chart,

196–198, 197

multi-trace line chart, 193–194

in Dash, 223–227

importing, 189

installing, 189

integrating Dash and, 223–230

interactivity features, 209–210

layout properties, 202–205

saving charts as images, 197

styling charts in, 200–209

layout properties, 201–205

themes, 205–209

using in Jupyter Notebooks, 189

polymorphism, 35

pop() method, 41

positional arguments, 61

PostgreSQL, 137

installing, 138–140

managing users and databases,
140–141

using in VS Code, 141–142

POST request, 165–166

Power BI, 213, 214

PowerShell, 5

premature optimization, 282

primary keys, 143

print() function, 39

pro-code tools, 213

profile, 12

programming languages, 26

prompt, 5

psql, 139–140

Psycopg2, 157

closing database connection, 160

installing, 158

running query from Python, 159

Public Health England, xxi–xxii

pwd command, 6

pydataset.data() function, 190

.py file, 106

pyodbc, 158

Python

alternative to, xxxii–xxxiii

automating database tasks with
SQL and, 157–160

charts, 186–188

code editors, 21–23

common functions in, 39

dictionary operations in, 41

installing, 3–4

directory's contents, 16

macOS troubleshooting, 17

running code, 17–18

running installer, 14–16

verifying installation, 16–17

Windows troubleshooting, 17

string manipulations in, 40

syntax, 28–30

version numbering, 15

Python in Excel, xxxiii–xxxiv

Python interpreter, 5

Python Package Index (PyPI), 19

Python Standard Library, 5

Python *vs.* Excel

automation, xxviii–xxx

interoperability, xxx–xxxii

modularity, xxviii

overview, xxv–xxvi

security, xxx–xxxiv

speed, xxvi–xxvii

py-trello, xxx

pytz library, 48

Q

qualifiers, 146

R

Random User Generator API, 165, 169,
170, 174, 180

random_users.py file, 182

range() function, 39, 55–56

ranges, creating for loop using, 55–56

range subproperty, 203

raw text cells, 100–101

read_excel() function, 118

README.txt file, 80

records, 142

refactoring, 282

#REF! error, xxiii

Reinhart, Carmen, xxi

relational database management system
(RDBMS), 137–138

relational databases, 137

- remote database, connecting to, 142
- remove() method, 40
- repetition in coding, 278–279
- replace() function, 40, 125
- repositories, 75, 77
 - centralizing work in single folder, 78–80
 - cloning, 83–84
 - creating from scratch, 83
 - local, 80–81
 - maintaining .gitignore file, 81–82
 - remote, 80–81
 - setting up, 81–84
 - syncing, 86–87
- requests
 - GET, 165
 - POST, 165–166
 - streamlining with reusable functions, 180–183
 - using API keys, 178–179
- requests.get(), 241
- reserved words, 57
- reset_index() function, 128, 129
- REST API, 162
 - internet communication protocols and, 163–164
 - Requests library, 169–171
 - request types, 165–166
 - response types, 166–168
- returns, 62–63
- reusability, 243–244
- reusable functions, 180–183
- RIGHT function, 45
- right joins, 151–153
- rmdir (remove directory) command, 8–9
- Rogoff, Kenneth, xxi
- roles, 140
- root directory, 6
- rows, adding, 145
- r subproperty, 204

S

- sandbox environments, 138
- scatter plots, 198–200, 200
- scope, 10, 62
- scripts, 51–54
 - building and running, 66–73

- code to put in, 105
- importing as modules, 70–71
- main block, 70–71
- parts of, 53–54
- running on command line, 69
- when to use, 52–53
- Scripts subdirectory, 19
- SE (Standard Edition), 146
- Seaborn, 188
- SELECT commands, 136
- SELECT DISTINCT, 148
- SELECT keyword, 146–147
- selectors, CSS, 221, 223
- semicolon (;), 141
- sequence, 55
- Series data types, 110–111, 115–116
- set_index() method, 113–114
- setters, 247–248
- shell, 5, 12
- SHELL environment variable, 12
- show_docs=True parameter, 190
- showgrid subproperty, 203
- showLegend property, 204
- showline subproperty, 203
- showticklabels subproperty, 203
- Simple Object Access Protocol (SOAP) APIs, 163
- simple-salesforce, xxx
- size property, 203
- slackclient, xxx
- slices, 119
- sort_index() function, 121
- sort_values() function, 121
- special variables, 261
- split() method, 40, 124
- spreadsheets
 - drawbacks of, xxi–xxii
 - errors in, xxii–xxiii
- SQL (Structured Query Language)
 - aggregation functions, 155
 - automating database tasks with Python and, 157–160
 - comparison operators for WHERE clauses, 149*t*
 - data types, 143–144, 144
 - overview, 133

SQL (*continued*)

queries

- aggregations, 154–155
- column transformations, 154
- complex, 156–157
- filters, 148–149
- groupings, 155–156
- joins, 148–149
- selections, 146–147

SQLAlchemy, 158

SQLite, 137

SQL Server, 137

SQL Shell (psql), 139

staging area, 84–85

Standard Edition (SE), 146

statements, 29–30

Step Into debugger command, 264–265

Step Out debugger command, 263–264

str accessor property, 124

strftime() method, 48

str() function, 39

string methods, 40

string operations, 123–124

strings, 32–33, 56

strip() method, 40

strptime() method, 48

Structured Query Language. *See* SQL

styles.css file, 222–223

sub() function, 122–123

sum() function, 39

SUM function, 155

SUMIFS, 129

super().__init__(), 252, 254

superclass, 249

superuser, 139

symbols, 33

syntax, Python

- assignments, 29–30
- case sensitivity, 28
- comments, 29
- indentation, 28–29
- statements, 29–30

system arguments, 72–73

system environment variables, 10

system settings, customizing and
storing, 10–13

T

Tableau, 213

tables. *See also* databases

- adding rows to, 145
- altering structure of, 144–145
- creating, 142–143
- deleting, 145
- deleting date from, 146
- primary keys, 143
- updating records in, 145–146

tags, HTML, 217, 219

TCP (Transmission Control
Protocol), 164

template attribute, 206

temporal data, 116

Terminal, 5, 69

testability, xxii–xxiii

text highlighting, 20

themes

- built-in, 205–206
- custom, 206–209

themes.py file, 206–207

tickangle subproperty, 203

tickcolor subproperty, 203

tickfont subproperty, 203

tickformat subproperty, 201

ticklen subproperty, 203

tickwidth subproperty, 203

tilde (~), 7

time commitment, 46

timedelta64[ns] type, 116

timedelta object, 47

time object, 46

Timeout error, 174

time zones, 49–50

title property, 202–203

title subproperty, 201, 203, 204

to_clipboard() function, 117

to_excel() function, 118

touch command, 9

Transmission Control Protocol
(TCP), 164

Trello, xxx

triple quotes ("""), 29

troubleshooting, 17

try...except blocks,
172–173

t subproperty, 204

- tuples, 37–38
- `TypeError`, 249, 263
- `type()` function, 39
- type subproperty, 203

U

- underscore (`_`), 149, 245
- Unicode, 33
- unique argument, 64
- unittest module, 269–271
- UPDATE command, 145
- `update_layout()` method, 195, 201–202, 206, 208, 209
- `update()` method, 41
- `upper()` method, 40
- user account, 6
- user-defined functions, 59–64
- user environment variables, 10
- user level, 6
- users
 - managing, 140–141
 - roles, 140
 - superuser privileges, 139

V

- VALUE, 11
- `ValueError`, 172, 247
- values, 10
 - accessing, 118
 - assigning to variables, 35
 - comparing, 35
 - labels, 118
 - position, 118
 - slicing, 119
 - updating, 120
- `values()` method, 41
- VARCHAR data type, 144
- variable names, 277–278
- variables, 27–28, 236
 - assigning values to, 35
 - class, 261
 - environment, 10, 177–178
 - function, 261
 - global, 54
 - names of, 277–278
 - special, 261
- VBA (Visual Basic for Application),
xxxii–xxxiii

- version control
 - charting with Excel *vs.* Python, 187
 - Jupyter Notebook, 107
 - overview, xxiv
 - reasons for using, 76
- version numbering, 15
- vertical concatenation, 126
- virtual private network (VPN), 142
- Visual Basic for Application (VBA),
xxxii–xxxiii
- VLOOKUP function, 127, 149–150
- VPN (virtual private network), 142
- VS Code, 21–23
 - debugger, 258–267
 - breakpoints, 258–259
 - Debug Console, 262
 - Debug Panel, 260–262
 - installing, 21–23
 - integrating with Git, 92
 - testing examples in, 30–31
 - using PostgreSQL in, 141–142

W

- WebSocket APIs, 163
- WHERE clause, 148, 156
- while loop, 59, 181. *See also* loops
- wildcard characters, 149
- Windows. *See also* macOS
 - directory navigation on, 8–9
 - environment variable commands
 - for, 11
 - file manipulation on, 10
 - file paths, 6*t*
 - filesystem navigation
 - commands, 6
 - installing PostgreSQL on, 139–140
 - modifying environment variables
 - on, 12–13
 - troubleshooting, 17
- `withdraw()` function, 240

X

- `xaxis_properties`, 201–202, 203
- `xaxis_title`, 201
- `xbbg`, xxxi
- xlwings library, xxix, 67
- x subproperty, 204

Y

yaxis_linecolor, 202
yaxis_properties, 201–202, 203
y subproperty, 204

Z

Zendesk API, xxxi
Zenpy, xxxi

zerolinecolor subproperty, 203
zeroline subproperty, 203
zerolinewidth subproperty, 203
Zoom, xxx
zoomus, xxx
Z shell (zsh), 5, 12