

INDEX

Symbols

- && (AND logical operator), 55
- * (asterisk), Loop sketch example, 169–170
- { } (curly brackets), 12–13, 53, 210
- % (modulo function), 181
- ! (NOT logical operator), 55
- || (OR logical operator), 55, 58
- () (parentheses), 13
- ; (semicolon), 13
- [] (square brackets)
 - arrays, 175
 - JSON arrays, 213, 214

A

- abs() function, 201
- absolute value, for sound
 - visualization, 201
- abstract clock project
 - combining with pixel art, 90
 - general discussion, 71–75
- Add File option, 94, 100, 155, 172, 188
- Add Library option, 168
- Add Mode option, 76
- add-on header, Digital Sandbox, 236
- Add Tool option, 125
- advanced data types, 111
- album() method, 198
- alert bar
 - Arduino IDE, 241
 - Processing, 7, 8–9
- alligator clips, 160, 161, 162
- American Standard Code for Information Interchange (ASCII), 149–151
- amplitude, sound, 198, 199–202
- analog data, 246
- analogRead() function, 247
- analogWrite() function, 247, 264
- AND logical operator (&&), 55
- Android mode, 6
- animation
 - basic, 55–59
 - with matrices, 88–90
- API menu, OpenWeatherMap, 215
- Arduino 233–234. *See also* sensor data
 - dashboard
 - analog versus digital data, 246
 - blinking LED project, 243–245
 - defined, 235
 - experimenting with, 261–265
 - Fio board, 242
 - IDE, 240–241
 - Serial Monitor, 248, 249, 250
 - installing software, 237–240
 - logging sensor data, 259–260
 - and MaKey MaKey, 160
 - microcontroller, 234–235
 - and Processing, 5
 - reading data from sensors, 247–250
 - reading versus writing data, 246–247
 - receiving Processing data on, 263–265
 - RGB LED, controlling, 261–265
 - selecting board and
 - choosing port, 242
 - sending data from Processing to, 261–263
 - serial ports, 242, 251, 255
 - sketches, 243
 - and SparkFun Digital Sandbox, 235–237
 - SparkFun Redboard, 235
 - Uno, 235
- arrays
 - general discussion, 175–177
 - JSON, 213–215, 225–226

- arrow keys, moving shape with, 151–153
- art, Processing as tool for, 4–5
- ASCII (American Standard Code for Information Interchange), 149–151
- aspect ratio, 96
- asterisk (*), Loop sketch example, 169–170
- ATMEGA328 microcontroller, 234
- AudioInput class, 186, 199, 203
- audio library, Processing, 186
- AudioMetaData class, 186, 195–198
- AudioOutput class, 186
- AudioPlayer class, 186, 188, 192, 194–195, 206
- AudioPlayer object, 187, 189, 196
- audio processing, 185–187
 - audio input, 198–199
 - AudioMetaData class, 195–198
 - AudioSample class, 193–195
 - basic audio player application, 187–189
 - experimenting with, 206–207
 - MaKey MaKey controller, 186
 - musical synthesizer project, 206–207
 - playback controls, creating, 189–193
 - soundboard, creating, 203–206
 - visualizing sound, 199–202
- AudioRecorder class, 186, 203–206
- AudioSample class, 186, 193–195, 203, 206
- author() method, 198
- autoConnect() custom function, 259
- avr-gcc, installing on Ubuntu Linux, 239

B

- background, for sound visualization, 199
- background() function
 - abstract clock project, 74
 - basic animation, 56, 57
 - digital collage, 42
 - image processing, 98, 105
 - pixel art, 29
 - text, 123, 127

- bar graph, visualizing sensor data in, 256–259
- baud rate, 247, 251, 264
- BeatDetect class, 186
- begin() function, 247, 264
- blinking LED project, 243–245
- blue() function, 156
- BLUR filter, 109, 110–111, 173–174
- board, selecting Arduino, 242
- boolean data type, 50
- bouncing ellipse animation
 - basic, 56–59
 - multiple ellipses, 59–62
- bounding box, maze game, 157–159
- bracelet, for MaKey MaKey controller, 162
- buffer
 - audio, 195, 201
 - Processing, 251–252, 260
- bufferUntil() function, 251–252
- built-in values, 66–68

C

- cameras array, 178–179, 180
- cam object, 178, 180, 181
- cam.start() function, 178
- Capture.list(), 180–181
- Capture object, 177, 178, 180, 181
- capturing video, 177–180
- Cartesian coordinate plane, 13–15, 19
- CENTER mode
 - imageMode() function, 97
 - textAlign() function, 122, 131
- central processing unit (CPU), 234
- char data type, 46, 50, 120
- chips, FTDI, 238
- City Symphonies*, 5
- classes. *See also specific classes*
 - libraries, 170
 - in OOP, 111–115
 - overarching, 187
- clock project
 - combining with pixel art, 90
 - general discussion, 71–75
- close() function, 260
- "cLOUDS" JSON object, 218
- CODED constant, 150, 153

- code window, 7–10
- collage
 - assembling shapes into, 42–45
 - photo, 100–105
- color
 - adding to pixel art, 23–25
 - color-changing feedback box, 141–142
 - line, 36–37
 - pen, for simple painting program, 143
 - sound visualization, 201–202
 - tints, applying to images, 105–108
- Color Selector tool, 25, 26
- command line, Ubuntu Linux, 239–240
- comma-separated values (CSV), 249, 250
- comments, 22
- communication, serial. *See* serial communication
- COM port options, Arduino, 242
- compound logic, 55
- computer vision (CV), 183
- concatenation, string, 120–121, 132
- conditional argument, 53–54
- console
 - Arduino IDE, 241
 - Processing, 7, 8–9, 46–47
- constant, `pinMode()` function, 243
- controller, maze game, 159–163
- `coord.getFloat()`, 219, 225
- coordinate plane, 13–15, 19
- "coord" object, JSON data, 217, 219, 225
- CORNER mode
 - `imageMode()` function, 98
 - `textAlign()` function, 122
- CORNERS mode
 - `imageMode()` function, 98
 - `textAlign()` function, 122
- CPU (central processing unit), 234
- Create Font tool, 124–125, 126
- `createRecorder()` function, 204
- `createWriter()` function, 259, 260
- creative tool, Processing as, 4–5
- CSV (comma-separated values), 249, 250
- `cue()` function, 188
- curly brackets (`{ }`), 12–13, 53, 210
- Current Weather Data page, OpenWeatherMap, 215–216
- cursor
 - coordinates of, 46–47
 - data dashboard project, 130
 - system variables, 67
- custom data parsing function, 220
 - listing data variables, 221–222
 - parsing weather data, 224–227
 - starting new function tab, 220–221
 - writing in `update_data` tab, 222–224
- CV (computer vision), 183

D

- Dafont, 124
- dashboard. *See also* sensor data dashboard; weather data dashboard
 - data, 129–132
 - OpenProcessing, 75–76
- `data` folder, 95, 100, 124, 125, 204
- data object, 196
- data pairs, JSON object, 210–211
- data parsing function. *See* custom data parsing function
- data types, 13. *See also specific data types*
 - advanced, 111
 - custom functions, 223
 - printing to console, 46
 - variables, 50
- `date()` method, 198
- `day()` function, 67, 71–72
- `delay()` function, 128–129, 245, 248
- "description" property, JSON data, 217
- design, Processing as tool for, 4–5
- Di Fede, Damien, 186
- digital data, 246
- `digitalRead()` function, 247
- Digital Sandbox, SparkFun. *See* SparkFun Digital Sandbox
- `digitalWrite()` function, 245, 247
- DILATE filter, 109
- documentation, library, 190
- DOWN global variable, 151, 153, 205

- downloading Processing, 6
 - draw() function, 11–12
 - abstract clock project, 74
 - AudioMetaData class, 196
 - basic animation, 56, 57, 58
 - basic audio player application, 188
 - custom functions, 221
 - data dashboard, 130–132
 - event functions, 136, 137, 138
 - get() function, 156–157
 - image processing, 105
 - JSON data, 220
 - live video, 179
 - matrices, 85
 - maze game, 155–156, 157, 158
 - mouseDragged() event function, 138, 139
 - moving shapes with arrow keys, 151
 - multiple ellipses, animating, 60
 - noLoop() function, 44
 - object-oriented programming, 114–115
 - photo booth project, 181–182
 - pixel art, 21–23
 - println() function, 46–47
 - removing or modifying outlines, 27
 - RGB LED, controlling, 262, 263
 - running video in loop, 170–171
 - sensor data dashboard, 252, 254
 - simple painting program, 140–143
 - single-song audio player, 191, 192
 - snowman, drawing, 44
 - for soundboard project, 204, 206
 - sound visualization, 200, 201
 - video filters, 173
 - visualizing sensor data, 257
 - weather dashboard project, 227, 228, 229–230, 231
 - drawing applications, 135–136
 - event functions, 136–138
 - experimenting with, 144
 - mouseDragged() event function, 138–139
 - mouse input variables, 136
 - mousePressed() event function, 140
 - rainbow-colored drawings, 138–140
 - simple painting program, 140–143
 - drivers, Arduino, 237–238
 - dynamic sketches, 49
 - basic animation, 55–59
 - experimenting with, 62
 - if() statement, 52–54
 - logic, 52–55
 - logical operators, 54–55
 - mathematical operators, 51–52
 - multiple ellipses, animating, 59–62
 - relational operators, 54
 - variables, 50–51
- E**
- ellipse() function, 34–35, 60–61, 70–71, 131, 157
 - ellipseMode() function, 98
 - ellipses. *See also* maze game
 - basic animation, 56–59
 - data dashboard, 130–131
 - drawing, 34–35
 - matrices, 83–88
 - modes for, 98
 - moving with arrow keys, 151–153
 - multiple, animating, 59–62
 - for snowman scene, 43
 - else if() statement, 142
 - else statement, 52, 55, 206
 - end shapes, line, 38–39
 - ERODE filter, 109
 - error messages, 9
 - event functions. *See also specific event functions*
 - keyboard, 149
 - libraries, 170–171
 - mouse, 136–138
 - for soundboard project, 204–206
 - example sketches, library, 190
 - Examples option, 201
 - Export button, 8
 - Export for Web button, 77, 78
 - Extract Here option, Ubuntu Linux, 239
- F**
- feedback box, color-changing, 141–142
 - FFT class, 186
 - fields, class, 111, 112, 113

- filename
 - font, 125, 126
 - image, 94
- fileName() method, 198
- fill() function
 - abstract clock project, 71–72
 - adding color with, 23–24
 - built-in values, 68
 - data dashboard project, 131
 - extending range of values, 68–70
 - order, importance of, 25–26
 - snowman project, 44
 - text boxes, 122–123
 - typewriter application, 127
- filter() function
 - experimenting with, 115–117
 - image processing, 108–111
 - object-oriented programming, 112, 114–115
 - videos, 173–174
- filters
 - applying to images, 108–111
 - applying to videos, 172–174
- Fio board, Arduino, 242
- float data type, 13, 50, 192
- flushing buffer, 260
- fonts, 124
 - creating, 124–125, 126
 - data dashboard, 130
 - loading, 125–126
 - size, 125, 127
- for() loop, 174–175, 176–177, 181, 201
- Fragmented Memory*, 5
- frame rate, 199
- frameRate() function, 228–229
- frequency, 198
- fruit controller, MaKey MaKey, 160–163
- Fry, Ben, 4
- functions. *See also* event functions; *specific functions*; *transformation functions*
 - defined, 11
 - library, 170–171
 - nested, 253
 - structural, 11–12
 - system, 66–67
 - time-related, 67–68, 71–72
- Future Technology Devices International (FTDI) drivers, 238

G

- g variable, 262, 263
- genre() method, 198
- getFloat() function, 219, 225
- get() function, 156–157, 201
- getString() function, 226
- getter functions, JSONObject class, 219
- global variables
 - bouncing ellipse animation, 55
 - custom functions, 221–222
 - data dashboard, 129
 - defined, 51
 - painting program, 140–141
 - PImage data type, 95
 - RGB LED, controlling, 262, 264
 - for sensor data dashboard, 251
 - for x- and y-coordinates, 151
- GPS coordinates, weather dashboard project, 217, 219–220, 225
- graph
 - sound visualization, 199–202
 - visualizing sensor data in, 256–259
- graphic buttons, 7, 8
- GRAY filter, 109, 110–111, 173–174
- grayscale values, 23–24
- green() function, 156
- grow global variable, 55, 58, 59–60

H

- height. *See* y-coordinates
- height value, 85
- Hello World program
 - Arduino, 243–245
 - Processing, 10–11
- hex values, 25
- HID (Human Interface Device) protocol, 160

- holiday card project, 33–34
 - animating with matrices, 88–90
 - digital collage, programming, 42–45
 - printing to console, 46–47
- Holm, Pete, 143
- hour() function, 67, 71–72, 73
- Human Interface Device (HID)
 - protocol, 160
- I**
- icon, for weather dashboard project, 229–230
- "icon" property, JSON data, 217
- IDE (integrated development environment)
 - Arduino, 237–241
 - Processing, 7–10
- if() statement, 52–54
 - abstract clock project, 74
 - ASCII and keyCode, 150
 - basic animation, 57–59
 - data dashboard, 130–131
 - event functions, 136
 - get() function, 157
 - keyPressed() event function, 153
 - library functions, 171
 - live video, 179
 - logical operators, 54–55
 - matrices, 83, 86, 87, 88
 - maze game, 158
 - mousePressed() event function, 140
 - multiple ellipses, animating, 61
 - photo booth project, 181
 - relational operators, 54
 - RGB LED, controlling, 264
 - sensor data dashboard, 252, 254
 - simple painting program, 141–142
 - single-song audio player, 191
 - soundboard project, 205, 206
 - sound visualization, 201
 - typewriter application, 127
 - update_data() function, 228
- image() function, 96
 - adding tints and filters to video, 172–173
 - library functions, 170
 - live video, 179, 180
 - photo collage, 101
 - photo for weather dashboard, 231
 - placing image, 97
 - weather dashboard project
 - icon, 229
- image processing, 93–94
 - advanced data types, 111
 - aspect ratio, 96
 - colored tints, 105–108
 - experimenting with, 115–117
 - filter() function, 108–111
 - finding image to use, 94–95
 - image() function, 96
 - imageMode() function, 97–98
 - matrices, 102–103
 - object-oriented programming, 111–115
 - photo collage, 100–105
 - PImage data type, 95–96
 - resolution, 96
 - transformation, 99–100, 103–105
- imageMode() function, 97–98, 105
- img variable, 95–96, 97
- import keyword, 170, 187
- Import Library option, 187, 250
- index, 175
- initializing
 - strings, 120
 - variables, 50–51, 95–96
- INPUT constant, pinMode() function, 243
- Install button, Video library, 168
- installing
 - Arduino software, 237–240
 - JavaScript mode, 76
 - Minim library, 187
 - Processing, 6–7
- inString, 252–253
- int data type, 13, 50, 140–141
- integrated development environment (IDE)
 - Arduino, 237–241
 - Processing, 7–10
- INVERT filter, 109
- isRecording() function, 206

J

- Java, installing on Ubuntu Linux, 239
- javaDocs, 190
- JavaScript library, 6
- JavaScript mode, 76–77
- JavaScript Object Notation (JSON), 210–213
 - arrays, 213–215, 225–226
 - custom data parsing function, 220–227
 - drawing weather dashboard in main tab, 227–229
 - experimenting with, 231
 - formatted data, 212–213, 216–217
 - getting weather data in, 215–218
 - nested objects, 211
 - objects, 210–211
 - pulling weather icon from Web, 229–230
 - unformatted data, 211–212
 - using data in Processing, 218–220
- `json.getJSONArray("weather")`, 226
- `json.getJSONObject()`, 219
- JSONLint, 212–213
- json object, 223
- JSONObject class, 218–220, 224, 225, 226
- JST right-angle connector, SparkFun Digital Sandbox, 236

K

- keyboard event functions, 149
- `keyCode` variable, 149
 - ASCII and, 149–151
 - soundboard project, 205
- `keyPressed()` event function
 - AudioMetaData class, 196
 - logging sensor data, 260
 - maze game, 156
 - musical synthesizer project, 206
 - overview, 149, 152–153
 - RGB LED, controlling, 262, 263
 - single-song audio player, 191
 - for soundboard project, 204–206
- `keyPressed` variable, 127, 149
- keypresses, 67, 127–128, 149

- `keyReleased()` event function, 149
- `keyTyped()` event function, 149
- key-value pairs, JSON object, 210–211
- key variable, 67, 149
- Khan Academy, 83

L

- latching, 205–206
- `lat` variable, 219, 225
- LEDs (light-emitting diodes)
 - blinking, 243–245
 - controlling, 261–265
 - SparkFun Digital Sandbox, 236, 240
- LEFT global variable, 151, 153, 205
- `length()` function, 189
- libraries, 168. *See also* Minim library
 - adding to Processing, 168
 - audio, 186
 - calling functions, 170–171
 - documentation, 190
 - example sketches, 190
 - JavaScript, 6
 - OpenCV for Processing, 183
 - for sensor data dashboard, 250–251
 - Serial, 250, 259, 261
 - tips for working with, 190
 - Unfolding, 217
 - Video, 168, 180, 182–183
 - Loop sketch example, 168–170
- Library Manager, 168, 187, 190
- license agreement, Arduino, 237
- light-emitting diodes (LEDs)
 - blinking, 243–245
 - controlling, 261–265
 - SparkFun Digital Sandbox, 236, 240
- light sensor, SparkFun Digital Sandbox, 236, 247–250, 256, 258
- light variable, 249, 251, 254, 256
- `line()` function, 13, 35–36, 138–139, 199
- lines
 - color, 36–37
 - drawing, 35–36
 - end shapes, 38–39
 - thickness, 37–38

- Linux
 - Arduino serial ports, 242
 - installing Arduino software on, 239–240
 - installing Processing on, 6
 - live video, 177–180
 - loadFile() function, 187, 195
 - loadFont() function, 126
 - loadImage() function, 96, 101, 231
 - loadJSONObject() function, 219, 223
 - loadSample() function, 195
 - local variables, 51
 - local weather dashboard. *See* weather dashboard project
 - logging sensor data, 259–260
 - logic, in Processing
 - if() statement, 52–54
 - logical operators, 54–55
 - overview, 52
 - relational operators, 54
 - logical operators, 54–55, 87
 - lon variable, 219, 225
 - loop, defined, 12
 - loop() function, 170, 243, 244–245, 247, 264
 - Loop sketch example, 168–170
 - adding video to, 171–172
 - library functions, 170–171
 - modifying to capture video, 177–180
 - tints and filters, 172–174
- M**
- Maeda, John, 4
 - "main" JSON object, 218
 - "main" property, JSON data, 217
 - mainCond object, 226
 - MaKey MaKey, 159–160
 - audio processing controller, 186
 - building controller, 160–162
 - connecting to computer, 162–163
 - experimenting with, 164
 - materials for use with, 148
 - musical synthesizer project, 206–207
 - tutorials, 162
 - map() function, 99, 193, 254
 - maps, 217
 - Massachusetts Institute of Technology (MIT), 4
 - mathematical operators, 51–52
 - matrices, 81–83
 - animating snowman with, 88–90
 - functions defining, 83–84
 - image processing, 99, 100, 102–103
 - mashup of projects with, 90
 - math behind, 83
 - origin of, 84
 - simplifying code with, 90
 - transformations with, 84–88
 - maze game, 147–148
 - detecting wall touches with get(), 156–157
 - experimenting with, 164
 - generating maze, 154–155
 - MaKey MaKey controller, 159–163
 - materials for, 148
 - moving shapes with arrow keys, 151–153
 - reading input, 149–153
 - theme song, adding, 206
 - victory condition, adding, 157–159
 - writing sketch for, 155–156
 - Maze Generator, 154–155
 - maze.png file, 154, 155
 - McKeague, Mark, 5
 - menu bar, 7, 8
 - metadata, 195–198
 - methods, class, 111, 112, 113, 114. *See also specific methods*
 - mic.left.get() function, 200
 - mic object, 199, 204
 - microcontroller, 234–235, 261
 - microphone
 - audio input with, 198–199
 - SparkFun Digital Sandbox, 236
 - millis() function, 67, 73–74
 - Minim class, 187, 195, 199, 203
 - Minim library, 185–187
 - audio input, 198–199
 - AudioMetaData class, 195–198
 - AudioSample class, 193–195
 - basic audio player application, 187–189
 - experimenting with, 206–207

- Minim library (*continued*)
 - information about, 186, 187
 - MaKey MaKey controller, 186
 - musical synthesizer project, 206–207
 - playback controls, creating, 189–193
 - soundboard, creating, 203–206
 - visualizing sound, 199–202
- minim object, 187, 196, 199, 203, 204
- minute() function, 67, 71–72, 73
- MIT (Massachusetts Institute of Technology), 4
- Mode drop-down menu, 76
- Mode Manager window, 76
- modifiers, 24
- modulo (%) function, 181
- month() function, 67
- mouseButton variable, 136
- mouseClicked() event function, 136, 138
- mouse cursor
 - coordinates of, 46–47
 - data dashboard project, 130
- mouseDragged() event function, 136, 138–139, 143
- mouseMoved() event function, 137
- mousePressed() event function, 137, 140
- mousePressed variable, 136
- mouseReleased() event function, 137
- mouse system variables, 67, 136
- mouseWheel() event function, 144
- mouseX variable, 46, 67
 - data dashboard project, 130, 131
 - get() function and, 157
 - images, 98, 99, 107
 - matrices, 85–86
 - maze game, 157–158
- mouseY variable, 46, 67
 - data dashboard project, 130, 131
 - get() function and, 157
 - images, 98
 - matrices, 85–86
 - maze game, 157–158
- Movie class, 170
- movieEvent() function, 170, 171
- movie object, 170, 172
- movies. *See* video

- MP3 file, playing, 188–189
- multimedia presentation, 182–183
- musical synthesizer project, 206–207
- mute() function, 188
- myPort object, 250, 251, 252, 255

N

- nested functions, 253
- nesting, JSON data, 211, 215
- New button
 - Arduino IDE, 241
 - Processing, 8
- New Tab option, 220
- noLoop() function, 44, 46
- noStroke() function, 27, 28, 36, 131, 257
- noTint() function, 106, 173
- NOT logical operator (!), 55
- numberLine array, 176

O

- object-oriented programming (OOP), 111–115
- objects. *See also specific objects*
 - adding to sketch, 195
 - creating for soundboard project, 203
 - defined, 111
 - JSON, 210
- Open button
 - Arduino IDE, 241
 - Processing, 8
- open source project, Processing as, 5–6
- OpenCV for Processing library, 183
- OpenProcessing, sharing projects on, 30–31, 75–76, 77–79
- OpenWeatherMap, 215–218, 219, 228, 229
- OR logical operator (||), 55, 58
- origin
 - ellipse, 35
 - imageMode() function, 97–98
 - of matrices, 84
 - rectangle, 19–20, 35
 - sketch window, 14, 15
 - text, 121–122

- OS X
 - Arduino serial ports, 242
 - installing Arduino software on, 238, 240
 - installing Processing on, 6, 7
- outlines, removing or modifying, 26–28
- OUTPUT constant, `pinMode()` function, 243
- output object, 259
- `outString` string, 263
- overarching class, 187

P

- package manager, Ubuntu Linux, 239
- painting program, 140–143
- pairs, JSON object, 210–211
- parameters, defined, 13
- parentheses (`()`), 13
- `parseInt()` function, 264
- parsing data. *See* custom data parsing function
- `pause()` function, 182, 188, 191
- pen color, simple painting program, 143
- `penSize` global variable, 144
- PFont data type, 124
- photo booth project, 180–182
- photo collage, 100–105
- photos, adding to weather dashboard, 231. *See also* image processing
- PImage data type, 95–96
 - maze game, 155
 - object-oriented programming, 112, 113, 114
 - photo collage, 100
 - photo for weather dashboard, 231
 - weather dashboard project icon, 229
- `pinMode()` function, 243–244, 245, 264
- pin names, Arduino, 236–237
- pins, microcontroller, 235
- pixel art, 17–18
 - adding color, 23–25
 - combining with time-based art, 90
 - drafting, 18–21
 - experimenting with, 30–31
 - order, importance of, 25–26
 - removing or modifying outlines, 26–28
 - scaling, 28–30
 - translating sketch into code, 21–25
- pixels, 13–14, 15
- PlayAFile.pde* example file, 198, 201
- playback controls, for audio player, 189–193
- `play()` function, 188, 191, 195
- `pmouseX` variable, 67
- `pmouseY` variable, 67
- `popMatrix()` function, 83, 90
- `position()` function, 189
- POSTERIZE filter, 109, 110
- `pos` variable, 192
- Preferences window, 9–10
- printing to console, 46–47
- `println()` function, 46–47
 - Arduino, 248, 249, 251, 254
 - custom functions, 223, 227
 - logging sensor data, 260
 - maze game, 157
 - RGB LED, controlling, 263
- PrintWriter class, 259–260
- Processing, 3–4
 - Cartesian coordinate plane, 13–15
 - data types, 13
 - error messages, 9
 - Hello World program, 10–11
 - IDE, 7–10
 - installing, 6–7
 - as open source project, 5–6
 - Preferences window, 9–10
 - as programming language, 4
 - structural functions, 11–12
 - syntax, 11, 12–13
 - as tool for art and design, 4–5
- Processing Foundation, 4, 6
- programming language, Processing
 - as, 4
- PROJECT parameter, `strokeCap()` function, 38–39
- properties, JSON object, 210–211
- pulsating shapes, creating, 59
- push button, SparkFun Digital Sandbox, 236
- `pushMatrix()` function, 83, 90

Q

quad() function, 39–40, 41
quadrilaterals, drawing, 39–40

R

r variable, 262, 263
RADIUS mode, 98
rainbow-colored drawings application,
138–140
reading data
from sensors, 247–250
versus writing data, 246–247
readStringUntil() function, 252, 253
Reas, Casey, 4
rectangles
adding color to, 23–25
color-changing feedback box,
141–142
displaying song position with, 193
drawing basic, 22–23
drawing pixel art with, 19–21
modes for, 98
order, importance of, 25–26
origin of, 19–20, 35
removing or modifying outlines,
26–28
scaling, 29–30
for snowman scene, 43
visualizing sensor data, 257
rect() function, 22–23
rectMode() function, 98
RedBoard, SparkFun, 235
red() function, 156–157
red green blue (RGB) color format,
23–24, 25, 140–142
relational operators, 54
reset button, SparkFun Digital
Sandbox, 236
Resig, John, 6
resolution
image, 96
video, 172
webcam, 181
rewind() function, 188
RFID reader kit, 264

RGB (red green blue) color format,
23–24, 25, 140–142

RGB LED, SparkFun Digital Sandbox,
236, 261–265

RIGHT global variable, 151, 153, 205

rotate() function, 70

abstract clock project, 74

image processing, 105

matrices, 84, 86–87

ROUND parameter, strokeCap() function,
38–39

Run button, 8, 26

RX LED, SparkFun Digital Sandbox, 245

S

Save button

Arduino IDE, 241

Processing, 8

saveFrame() function, 182

scale() function, 70

image processing, 99, 105

matrices, 87–88

scaling pixel art, 28–30

second() function, 67

abstract clock project, 71–72, 73, 74

extending range of values, 68–70

matrices, 86, 88

semicolon (;), 13

sensor data dashboard, 250

fetching serial data, 252–254

importing libraries and creating
variables, 250–251

preparing Processing for serial
communication, 251–252

testing serial connection, 254–256

visualizing data, 256–259

sensors, 233–234. *See also* sensor
data dashboard

analog versus digital data, 246

Arduino

defined, 235

IDE, 240–241

installing software, 237–240

blinking LED project, 243–245

experimenting with, 261–265

installing Arduino software, 237–240

- logging data from, 259–260
- microcontroller, 234–235
- reading data from, 247–250
- reading versus writing data, 246–247
- RGB LED, controlling, 261–265
- selecting board and choosing port, 242
- SparkFun Digital Sandbox, 235–237
- `Serial.available()` function, 264
- `Serial.begin()` function, 247
- serial communication, 250
 - analog data, 247
 - fetching serial data, 252–254
 - preparing Processing for, 251–252
 - testing serial connection, 254–256
 - tip for Windows users, 259
 - visualizing sensor data, 256–259
- `serialEvent()` function, 252
- Serial library, 250, 259, 261
- `Serial.list()` function, 251
- Serial Monitor, Arduino IDE, 248, 249, 250
- Serial object, 250
- `Serial.parseInt()` function, 264
- serial ports, Arduino, 242, 251, 255
- `Serial.println()` function, 248
- `setup()` function, 11–12
 - arrays, 176
 - audio input, 199
 - AudioMetaData class, 196
 - basic animation, 55–56, 57
 - basic audio player application, 188
 - blinking LED project, 243–244
 - capturing video, 177–179
 - custom functions, 221, 223
 - data dashboard, 129–130
 - image processing, 105
 - JSON data, 220
 - JSONObject class, 219
 - local variables, 51
 - maze game, 155
 - Minim library, 187
 - moving shapes with arrow keys, 151
 - multiple ellipses, animating, 60
 - object-oriented programming, 114
 - parameters, 13
 - photo booth project, 180–181
 - pixel art, 21–22
 - preparing for serial communication, 251
 - reading data from sensors, 247
 - RGB LED, controlling, 262, 264
 - single-song audio player, 190
 - for snowman scene, 42
 - for soundboard project, 203–204
 - text, 127
 - weather dashboard project, 227, 229–230, 231
- shapes. *See also* ellipses; rectangles
 - abstract clock project, 73–74
 - animating, 56–59
 - creative uses, 47
 - drawing, overview, 34
 - lines, 35–39
 - moving with arrow keys, 151–153
 - moving with matrices, 82
 - quadrilaterals, 39–40
 - for snowman scene, 43–45
 - triangles, 41
- Show Sketch Folder option, 95, 100
- sine wave
 - analog signal, 246
 - audio, 200
- single-song audio player, 189–193
 - AudioSample class, 193–195
 - displaying metadata, 195–198
- size, font, 125, 127
- `size()` function, 13, 22, 42
- sketchbook, explained, 8
- sketch folder, 95
- Sketch menu, 94, 95
- sketch window, 8
 - for audio input, 199
 - Cartesian coordinate plane, 13–15, 19
 - displaying song position in, 191–193
 - scaling, 30
- `skip()` function, 189
- slide potentiometer, SparkFun Digital Sandbox, 236
- slide switch, SparkFun Digital Sandbox, 236
- `smooth()` function, 55–56

- snowman project
 - animating with matrices, 88–90
 - drawing snowman, 42–45
- software, Arduino, 237–241
- song.mp3* file, 188
- song object, 196
- `song.position()`, 192
- sound. *See* audio processing
- soundboard project, 203–206
- sound page tutorial, 207
- sound sensor, SparkFun Digital
 - Sandbox, 249, 256, 258
- sound variable, 249, 251, 254, 256
- SparkFun Digital Sandbox. *See also*
 - sensor data dashboard
 - Arduino IDE, 240–241
 - blinking LED project, 243–245
 - overview, 234, 235–237
 - reading data from sensors, 247–250
 - RGB LED, controlling, 261–265
 - selecting board and choosing
 - port, 242
- SparkFun Inventor's Kit, 211–213
- SparkFun RedBoard, 235
- `split(inString)`, 253
- square brackets ([])
 - arrays, 175
 - JSON arrays, 213, 214
- SQUARE parameter, `strokeCap()`
 - function, 38–39
- square wave, 246
- stacking filters, 110–111
- stand-alone filters, 110
- statistical data, dashboard for, 129–132
- Stearns, Phillip, 5
- Stillman, Dave, 171
- Stop button, 8
- `str()` function, 263
- String data type, 46, 50, 120–121,
 - 127, 132
- strings, JSON data, 210
- stroke, defined, 27
- `stroke()` function, 27–28, 36, 138,
 - 143, 201
- `strokeCap()` function, 38–39, 45
- `strokeWeight()` function, 37–38, 138
- structural functions, 11–12
- syntax, Processing, 11, 12–13
- synthesizer project, 206–207
- "sys" property, JSON data, 217
- system functions, 66–67
- system variables, 66–67, 136, 149

T

- tabs, in Processing, 220–221
- temperature, weather dashboard
 - project, 218
- temperature sensor, SparkFun Digital
 - Sandbox, 236, 249,
 - 256, 258
- temp variable, 249, 251, 254, 256
- testing serial connection, 254–256
- text, 119–120
 - data dashboard, 129–132
 - `delay()` function, 128–129
 - experimenting with, 133
 - fonts, 124–126
 - modifier functions, 122–123
 - origin of, 121–122
 - String data type, 120–121
 - `text()` function, 121–122
 - typewriter application, 126–129
- `textAlign()` function, 122, 131
- text boxes, 122–123, 127
- text file, logging sensor data in,
 - 259–260
- `textFont()` function, 127, 131
- `text()` function, 121–122
 - AudioMetaData class, 196–197
 - color-changing feedback box, 141
 - data dashboard project, 131
 - single-song audio player, 191
 - typewriter application, 127
 - weather dashboard project, 227
- `textSize()` function, 122, 127, 131
- thickness, line, 37–38
- this keyword, 170
- THRESHOLD filter, 109, 110, 117
- time-based art, 65–66
 - abstract clock, 71–75
 - built-in values, 66–68
 - data dashboard project, 132

- extending range of values, 68–70
- matrices, 90
- mouse and keypresses, 67
- sharing, 75–79
- time-related functions, 67–68, 71–72
- transformation functions, 70–71
- useful, 79
- time-lapse program, 182
- tint() function, 105–108, 172–173
- tints
 - applying to images, 105–108
 - applying to videos, 172–174
- title() method, 198
- Tools Manager, 125
- Tools menu, 25, 125
- touch variable, 157
- transformation functions, 70–71. *See also specific functions*
 - image processing, 99–100, 103–105
 - matrices, 84–88
- translate() function, 70–71
 - image processing, 103
 - matrices, 85–86, 90
- transparency, image, 105, 107
- triangle() function, 41
- triangles, drawing, 41
- trigger() function, 195
- trim() function, 253
- TX LED, SparkFun Digital Sandbox, 245
- typewriter application, 126–129, 133

U

- Ubuntu Linux, installing Arduino
 - software on, 239–240
- Unfolding library, 217
- unMute() function, 188
- update_data() custom function, 223–229
- update_data tab
 - creating, 220–221
 - listing data variables, 221–222
 - writing basic custom function in, 222–224
- UP global variable, 151, 153, 205
- Upload button, Arduino IDE, 241, 245

- uploading to OpenProcessing, 76, 77–79
- USB Mini-B connector, SparkFun Digital Sandbox, 236

V

- values
 - built-in, 66–68
 - JSON object, 210–211
- vals float array, 253–254
- val variable, 247–248
- variables. *See also specific variables*
 - basic animation, 55–59
 - custom functions, 221–222
 - defined, 46
 - fonts, 125–126
 - global, 51, 55, 95, 129, 140–141
 - for images, creating, 100–101
 - initializing, 50–51, 95–96
 - local, 51
 - mathematical operators, 51–52
 - multiple ellipses, animating, 59–62
 - parts of, 50–51
 - for sensor data dashboard, 250–251
 - system, 66–67, 136, 149
 - where to use, 51
- Verify button, Arduino IDE, 241
- victory condition, adding to maze game, 157–159
- video, 167–168
 - adding to sketch, 171–172
 - applying tints and filters, 172–174
- arrays, 175–177
 - capturing, 177–180
 - experimenting with, 182–183
 - for() loop, 174–175, 176–177
 - libraries, 168–171
 - photo booth, 180–182
- Video library, 168, 180, 182–183
- visualizing
 - sensor data, 256–259
 - sound, 199–202
- .v/w files, 125
- volume, 199

W

- walls
 - creating for bouncing ellipse animation, 57–59
 - touches, detecting with `get()`, 156–157
- Warhol, Andy, projects inspired by, 115–117
- weather array object, 226
- weather dashboard project
 - custom data parsing function, 220–227
 - drawing dashboard in main tab, 227–229
 - experimenting with, 231
 - getting weather data in JSON, 215–218
 - JSON arrays, 213–215
 - JSON data overview, 210–213
 - overview, 209–210
 - pulling weather icon from Web, 229–230
 - using JSON data in Processing, 218–220
- `weather.getJSONObject()`, 226
- `weatherIcon` object, 229
- "weather" object, JSON data, 217
- webcam, capturing video with, 177–180
- web-export folder, 77, 78
- whitespace, 253
- width. See x-coordinates
- width value, 58, 85
- "wind" JSON object, 218
- Windows
 - installing Arduino software on, 237–238, 240
 - installing Processing on, 6, 7
 - serial communication tip, 259
- Wolfram MathWorld, 83
- writing data, 246–247

X

- x-coordinates
 - Cartesian coordinate plane, 14, 15
 - for ellipses, 34
 - maze game, 155
 - of mouse cursor, 46–47
 - moving shapes with arrow keys, 151, 153
 - for pixel art, 20–21
 - `rect()` function, 22
- x global variable
 - audio input, 199
 - bouncing ellipse animation, 55, 56, 58
 - multiple ellipses, animating, 59–60, 62
 - sound visualization, 201

Y

- Y1 value, sound visualization, 200
- Y2 value, sound visualization, 200
- y-coordinates
 - Cartesian coordinate plane, 14, 15
 - for ellipses, 34
 - maze game, 155
 - of mouse cursor, 46–47
 - moving shapes with arrow keys, 151, 153
 - for pixel art, 20–21
 - `rect()` function, 22
- `year()` function, 67