INDEX

Page numbers referring to figures and tables are followed by an italicized *f* or *t*, respectively.

A

A/B (blue/green) testing, 263
ACLU (American Civil Liberties Union), 177
`add_edge` function, 33
`add_node` function, 33
AEQD (azimuthal equidistance), 152–153
`aeqd_to_wgs84` function, 152–153
AGP. *See* art gallery problem application
`all_pairs_lowest_common_ancestor` function, 86
`all_shortest_paths` function, 118
alpha parameter, 120
Amazon Web Services (AWS), 261
American Civil Liberties Union (ACLU), 177
Anaconda
  installing, 4–8
    Linux, 4–6
    macOS, 8, 8*f*
    Windows, 6–8, 6*f*, 7*f*
  installing Spyder IDE with, 10
  Jupyter Notebooks, 11–12
  virtual environment setup, 9, 63
anomaly detection, 47, 53
application state planning, 235–236
`apply` function, 70, 168–169, 192
apt-get, 11
area metrics, 190
areas of responsibility (AORs)
  art gallery problem, 219–223, 220*f*, 225
  emergency service planning scenario, 171, 173
art gallery problem (AGP) application, 209–232
  advanced features, 233–256
    graphics in Python, 245–250
    process parallelism, 241–245
    running example application, 254–255
    saving and reloading data, 251–254
    state manager development, 237–241
    user interaction mapping, 234–237
  algorithm and data structures, 216–231
    area of responsibility, 219–223, 220*f*
    complex polygons, 223–225, 224*f*
    field of view and effective range, 229–231, 230*f*
    greedy coloring, 218, 218*f*
    triangular tessellation, 216–217, 217*f*
    weighted and budgeted coverages, 225–229, 225*f*, 228*f*, 229*f*
  delivery pipeline, 257–267
    distributing with cloud microservices, 260–264, 261*f*
    licensing with PyArmor, 264–265
    open source delivery, 265–266

art gallery problem (AGP)
application *(continued)*
delivery pipeline *(continued)*
packaging with Python interpreters, 259–260
setup scripts, 258–259
existing research on, 212–214, 213*f*, 214*f*
geometric and graph representations, 214–216, 215*f*
premise of, 209–210
use cases, 211–212
`assign_triangles` function, 221
`association` function, 195
association matrices, 194–197, 194*f*
attribute characters, 22
authority score
HITS algorithm, 80–83
social network simulation, 104, 106
updating authority scores, 81
AWS (Amazon Web Services), 261
`axis` parameter, 71, 167
azimuthal equidistance (AEQD), 152–153
azimuth projection, 152

B

`background` variable, 246
bagging, 182
balanced exchange, 75
Baran, Paul, 121
base stations (towers)
defined, 142
gathering locations, 149–150
identifying, 144
rogue, 143
betweenness centrality, 35–38, 35*f*, 57
biased random walks (biased walks), 97, 100, 104, 111
BIM (building information modeling) programs, 211
Bledsoe, Woodrow Wilson, 176, 179
blind spots, 211, 232
`blit` function, 247, 250
blue/green (A/B) testing, 263
Booleans
defined, 18
intersections, 153–154
notation, 18–19, 21*t*
branch nodes, 84–85
Brandes, Ulrik, 36
broadcast addresses, 60
building information modeling (BIM) programs, 211

C

`-c` (`--count`) option, 63
C language, 92, 216
C++ language, 183
`capacity` attribute, 112, 114
capacity constraint, 121
`cascaded_intersections` function, 155, 157
cell IDs (CIDs), 146
cells, defined, 12
centrality, 35–38, 51*f*
betweenness centrality, 35–38, 35*f*, 57
defined, 35
degree centrality, 37–38
by protocol, 52–61
identifying unusual levels of traffic, 54–57
neighbors and information exchange ratio, 57–61
port numbers subgraphs, 52–54
centroid, defined, 136
centroid location, 136–137, 137*f*
`check_clicked_existing_vertex` function, 240
`check_clicked_within_room` function, 240
`choice` function, 103–104, 116, 122, 193
chroma key filming (green-screen filming), 249
Chvátal, Václav, 212
Chvátal AGP theorem, 212–213
Chvátal's upper bound, 212–213
CIDs (cell IDs), 146
`circle` function, 247
`city_gj` variable, 167
`city_shape` variable, 167, 170
`clear_surface` function, 249
cliques
analysis, 39–40, 39*f*, 61
defined, 76
identifying, 76–78, 78*f*
maximal cliques, 40, 77

closed chains, 129
closeness, 35, 38, 52
cloud microservices, distributing with, 260–264
co-location, 138–140, 139*f*
common ancestors, 84–86
complete subgraphs, 39
`complex` function, 150–151
computational geometry theory, 127–140. *See also* art gallery problem application
  common operations, 132–140, 133*f*
    centroid location, 136–137, 137*f*
    co-location, 138–140, 139*f*
    perimeter length, 137–138
    tessellation, 133–136, 134*f*
  defined, 127
  facial recognition, 175–205
  location triangulation, 141–160
  shapes, 128–132
    line segments, 128–129
    points, 128–129
    polygons, 129–132, 130*f*, 131*f*
    vertex order, 132
  Voronoi diagrams, 161–174
computer vision, 179, 184. *See also* facial recognition
`concat` function, 70–71
concert security scenario, 132–140, 133*f*
  centroid location, 136–137, 137*f*
  co-location, 138–140, 139*f*
  perimeter length, 137–138
  tessellation, 133–136, 134*f*
conda utility, 4–7, 9, 11
connected components, 41
connected graphs, 40–41
connectedness, 40–41, 41*f*
connections (edges), in graphs, 27
conservation constraint, 121
constrained Delaunay triangulation, 214, 216
`contains` function, 81, 138–139
`continue` keyword, 116, 118
`controlled` variable, 239–240
convex hulls, 187, 187*f*
`coords` parameter, 216
correlation ratio, 197–198
`correlation_ratio` function, 198
`create_using` function, 63
cross-validation, 188, 200
`cross_val_score` function, 200
cutset, 121–122
cv2 library, 184
`cv` parameter, 200
cycle of graphs, 32
cyclic graphs, 32

D

DAGs (directed acyclic graphs), 85
`DataFrame` objects, 69–73, 76, 81, 88, 107, 150, 167–170, 191–193, 195
Data Manager service, 261*f*, 262
`DataSaver` class, 227–228, 252
DDoS (distributed denial-of-service) attacks, 38
decision trees, 179–182, 180*f*, 201
degree centrality, 37–38
Delaunay triangulation, 216
delivery pipeline, 257–267
  distributing with cloud microservices, 260–264, 261*f*
  licensing with PyArmor, 264–265
  open source delivery, 265–266
  packaging with Python interpreters, 259–260
  setup scripts, 258–259
descriptive security analysis, 90
`detector` variable, 183, 186
deterministic FSMs, 95, 97
device tracking application, 148–159, 148*f*
  geodesic polygons, 150–153
  intersections, 153–157
  mapping and comparing results, 157–159, 158*f*
  reducing search area, 159
  tower locations, 149–150
dictionary comprehensions, 15–17, 137
`difference` function, 144
`DiGraph` objects, 38, 42, 42*f*, 52, 63, 108, 115
Dijkstra's algorithm, 77
`dim` parameter, 51
directed acyclic graphs (DAGs), 85

directed graphs, 31–32
betweenness centrality, 36
cliques, 40, 76–77
creating in NetworkX, 33
degree centrality, 37–38
edge multiplicity, 43
HITS algorithm, 80
network analysis graphs, 63
port numbers subgraphs, 52
resource allocation, 108–109
social network analysis, 73
state machine graphs, 94, 94*f*
directed preferential attachment (DPA), 114
disconnected graphs, 40–41, 41*f*
discrete classification, 176
`DisplayAGP` class, 242–243
`Display` class, 246–248, 247*f*
`distance` function, 138, 164, 222
distance metrics, 190
distributed denial-of-service (DDoS) attacks, 38
division of labor, 241
`.__doc__` attribute, 237
Docker, 262–263
docstrings, 236–237
DPA (directed preferential attachment), 114
`draw` function, 34, 247
Drawn state, 235
`dtypes` property, 72
`DummyClassifier` class, 200
dummy classifiers, 199–202
`dump` function, 203
`dumps` (dump string) function, 147, 252

E

edge attributes, 32–33, 43, 50, 226
edge capacity, 112–113
edge multiplicity, 32, 42–43, 42*f*
edges (connections), in graphs, 27
edges (sides; faces), in polygons, 129
edge weight, 32–33, 43, 52–53, 62–63, 73
effective range
cellular networks, 144
security sensors, 211, 213, 229–230
Electronic Frontier Foundation, 177
`eliminate_small_areas` function, 155–156
emergency service planning scenario, 163–173
city shape, 164–167, 165*f*
distance function, 164
generators, 167–169, 169*f*
Voronoi tessellation, 170–173, 171*f*, 172*f*
empirical mean (sample mean), 93
ensemble classifiers, 201
ethics
facial recognition, 178–179
social network analysis, 89
tracking, 144–145
Euclidean distance, 164
`event` class, 237–238
`exchange_ratios` function, 60
exploratory analysis (unsupervised learning), 176

F

face detector component, 183
faces (edges), in polygons, 129
Facial Identification Scientific Working Group (FISWG), 189
facial recognition, 175–205, 178
data loading, 191–193
data set, 177–178
decision tree classifiers, 179–182, 180*f*
defined, 175
ethics of, 178–179
facial statistics, 189–190, 189*f*
feature engineering, 193–198
association matrices, 194–196, 194*f*
correlation ratio, 197–198
mutual information classification, 196–197
locating facial landmarks, 185–188, 187*f*
memory management, 190–191
model persistence, 203–204
model training, 198–203
establishing baseline, 199–200
random forest, 201–202
splitting data, 199
testing holdout images, 202–203
overview of, 176–177

processing image data, 184–185, 185*f*
proof of concept, 188–204
representing facial geometry, 182–184, 182*f*
features, in databases, 71
field of view, 211, 213, 229–231, 230*f*
`file_to_graph` function, 62
`find_cliques` function, 40
finite state machines (FSMs), 93–95, 94*f*, 100–101
Fisk, Steve, 212
FISWG (Facial Identification Scientific Working Group), 189
fitting classifiers, 200, 203
`flip` function, 247
`Floor` class, 234
flow functions, 109
`font_color` function, 34
`format` parameter, 165
freezing applications, 259–260
`from_dict` function, 253–254
FSMs (finite state machines), 93–95, 94*f*, 100–101
functools library, 150

G

GCP (Google Cloud Platform), 261
General Data Protection Regulation, 178
General Game Playing (GGP), 98
`Generator` objects, 40
generators (seeds)
defined, 162
gathering, 167–169, 169*f*
Voronoi tessellation, 162–163
geocoding, 145
`GeoDataFrame` class, 150
`GeoDataFrame` objects, 168–169
GeoJSON, 150, 165–167
geolocation, 145, 149–150, 159, 167
GeoPandas library, 150, 169–170
geovoronoi library, 163, 170
`get_front_face_detector` function, 183
`get_image_files` function, 191
`get_mods` function, 239
`get_shapely_circle` function, 153
GGP (General Game Playing), 98
Gini impurity coefficient, 181
GIS Stack Exchange, 151
GitHub, xxiv, 10, 48, 258, 265
goal-oriented planning, 98
Google Cloud Platform (GCP), 261
Graph builder service, 261*f*, 262
graphic elements, 245–250
`Display` class, 246–248, 247*f*
layers, 248–250
`Sprite` class, 248–250, 250*f*
`Surface` class, 246–248, 247*f*
graphs, defined, 27
graph theory, 27–44. *See also* art gallery problem application
creating graphs in NetworkX, 32–34, 34*f*
graph properties, 34–43
centrality, 35–38
cliques, 39–40
closeness, 35
connectedness, 40–41
edge multiplicity, 42–43
graphs, defined, 27
overview of, 31–32
uses for, 28–31, 28*f*, 30*f*, 31*f*
Graphviz, 33
`greedy_color` function, 218
greedy coloring, 212–214, 213*f*, 218, 218*f*, 245
Greek letters and functions, 22, 22*t*
green-screen filming (chroma key filming), 249
Guggenheim Museum, 214, 214*f*. *See also* art gallery problem application

H

`-h` (`--help`) option, 63, 254
`handle_click` function, 240–241
`handle_keydown` function, 238–239, 241
`handle_keyup` function, 239–241
hard classification, 203
hardware parallelism, 263
`has_path` function, 110
Hay, Andrew, 29
higher-order functions, 150
histogram of oriented gradients (HOG), 183
HITS (Hyperlink-Induced Topic Search), 80–83, 104, 109

hits function, 81
HNIs (home network identities), 146
holdout sets
testing holdout images, 202–203
true, 192–193
hole_p variable, 224
holes. *See* linear ring polygons
home network identities (HNIs), 146
horizontal scaling, 262–263
hubs
defined, 80
HITS algorithm, 80–83
social network simulation, 104, 106
updating hub scores, 82
hub_send function, 106
Hyperlink-Induced Topic Search (HITS; Hubs and Authorities), 80–83, 104, 109
hypothesis testing, 200

I

-i (--iface) option, 63
-i all option, 64
ICMP (Internet Control Message Protocol) packets, 49–50
IER (information exchange ratio), 59–61
IFD (information flow distance), 100–104
imutils library, 184, 236
in-degree centrality, 37–38, 54–56, 59–60, 78, 85, 113, 115, 121
in_degree function, 55, 79
in_edges function, 59, 114
Indicators of Compromise (IoCs), 47
information entropy, 74
information exchange ratio (IER), 59–61
information flow distance (IFD), 100–104
information flow game, 110–124
edge capacity, 112–113
game phases, 113–117
message movement phase, 115–117
network disruption phase, 117
network evolution phase, 113–115
game simulation, 118–120
improvements to player 2, 120–124, 124*f*
source and sink node selection, 117–118
weighted random choice, 111–112
information propagation, 74–76
informed consent
facial recognition, 178
tracking, 145
__init__ function, 249
init function, 246
init_surface function, 250, 253
input alphabet, 94–96, 101–102, 113
instances, in decision trees, 180
Internet Control Message Protocol (ICMP) packets, 49–50
interpreters
defined, xxiii
packaging with, 259–260
intersection function, 144, 154
intersections, finding, 153–157
intersects function, 138
IoCs (Indicators of Compromise), 47
isinstance function, 157
items function, 186

J

joblib library, 203
join function, 244
json library, 251
json parameter, 168
Jupyter Notebooks, 11–12

K

key_features variable, 196
key parameter, 53, 222, 239–240
key space, 111–112
kinetic information, 75–76, 78–79
Klee, Victor, 212
Kubernetes, 262–263, 266

L

-l (--load) argument, 63
labels function, 34
LACs (location area codes), 146
lambda functions, 53
landmark detector component, 183, 188, 191
layers, in graphics, 248–250
LCA (lowest common ancestor), 84–87

leaf nodes, 85, 180–181, 201
leave one out (LOO) algorithm, 188–189, 199–200
`left_click` function, 241
`len` property, 50
libpcap library, 47
licensing, with PyArmor, 264–265
linchpin employees, 109–110, 112
linear ring polygons (rings; holes)
  art gallery problem, 223–225, 224*f*
  overview of, 130–131, 131*f*
line segments
  AGP algorithm, 218
  creating, 129
  overview of, 128–129
  perimeter length, 138
  polygons, 129–132
  Voronoi tessellation, 162–164
`LineString` class, 129
`LineString` objects, 129, 190
link prediction theory, 100
Linux
  installing Anaconda, 4–6
  installing IDE without Anaconda, 11
  Jupyter Notebooks, 47
  network card in promiscuous mode, 63
  open source delivery, 266
  packet capture library, 11
list comprehensions
  dictionary comprehensions vs., 16–17
  emergency service planning scenario, 167, 170
  facial recognition, 193, 197
  finding intersections, 157
  identifying cliques, 77
  identifying most absorbent node, 59
  identifying unusual levels of traffic, 54
  limitations of, 15
  overview of, 14–15
  port numbers subgraphs, 52
`load` function, 203–204, 227, 246
`loads` function, 70, 165, 253
`locate_landmarks` function, 186, 188, 191
location area codes (LACs), 146
`locations` variable, 169
location triangulation, 141–160
  device tracking application, 148–159, 148*f*
    geodesic polygons, 150–153
    intersections, 153–157
    mapping and comparing results, 157–159, 158*f*
    reducing search area, 159
    tower locations, 149–150
  ethics of, 144–145
  network interface data, 142–144
  OpenCellID API structure, 145–148, 147*f*
  proof of concept, 148–159
`loc` function, 73
logical statements, 18–20
LOO (leave one out) algorithm, 188–189, 199–200
`lookup_tower` function, 146, 148–149
loose coupling, 253–254
lowest common ancestor (LCA), 84–87
LucidChart, 234

M

MAC (media access control) addresses, 48–50
machine learning (ML), 18, 176–204
macOS
  installing Anaconda, 8, 8*f*
  network card in promiscuous mode, 63
  packet capture library, 47
Maltego, 29–30
Markdown, 12
Mastodon data, analysis of, 67–90
  converting data to graph, 69–73
    building graphs, 72–73, 74*f*
    examining data, 69
    structuring data, 69–72
  defined, 67
  proof of concept, 87–88
  research questions, 74–87
    cliques and most influential users, 76–78, 78*f*
    information propagation, 74–76
    most influenced users, 78–79, 79*f*

Mastodon data, analysis of *(continued)*
research questions *(continued)*
node ancestry, 83–87, 84*f*, 86*f*
topic-based information exchange, 79–83, 82*f*
math conventions. *See* programming and math conventions
mathematical notation, 18–22
attribute characters, 22
Boolean notation, 18–20, 19*t*
Greek letters and functions, 22, 22*t*
overloaded symbols, 18*f*
set notation, 20–22, 21*t*
Matplotlib library, 32–34
max-flow, min-cut theorem, 112, 121–122
maximal cliques, 40, 77
maximum area threshold, 219
`max_iter` parameter, 81
MCCs (mobile country codes), 146
media access control (MAC) addresses, 48–50
membership rules, 20, 21*t*
meshes, 218, 220, 220*f*, 225–229, 225*f*, 229*f*
metric space, 163–164, 167
MI (mutual information) classification, 196–197
microservices, 261–264, 261*f*
Milgram, Stanley, 68
`minimum_cut` function, 122
minimum viable product (MVP), 210, 257
`min_samples_leaf` parameter, 201
`min_samples_split` parameter, 201
ML (machine learning), 18, 176–204
mobile country codes (MCCs), 146
mobile network codes (MNCs), 146
model persistence, 203–204
model training, 198–203
establishing baseline, 199–200
random forest, 201–202
splitting data, 199
testing holdout images, 202–203
monetization, 258–259, 263–266
Monte Carlo simulations, 91–125
information flow game, 110–124
overview of, 92–93
proof of concept, 109–124
random walks and, 95–97, 96*f*, 99*f*
simulations, defined, 92
social network simulation, 100–109
most influenced users, identifying, 78–79, 79*f*
most influential users, identifying, 76–78, 78*f*
`mp_agp_floorplan` function, 245
`mp_agp_solver` function, 244
`mp_solve_floors` function, 244
multiclass classification, 179
`MultiDiGraph` objects, 38, 42, 42*f*, 49–50, 53, 58, 62
multiprocessing (processor parallelism), 243–245
mutual information (MI) classification, 196–197
MVP (minimum viable product), 210, 257

N

`names` parameter, 167
neighbors, 31, 57–61, 58*f*
nested objects, 70, 251
`n_estimators` parameter, 201
NetGear Ocuity cameras, 229
`net_graph` objects, 50, 58, 60
network analysis graphs, 45–65
building, 47–51, 51*f*
centrality, 52–61
examining neighbors, 57–61, 58*f*
identifying unusual levels of traffic, 54–57, 56*f*
port numbers subgraphs, 52–54
identifying data for, 48–49
network topology, 46, 46*f*
packet analysis, 46–51, 51*f*
proof of concept, 61–64
network interface cards (NICs), 48
NetworkX library
art gallery problem application, 236
betweenness centrality, 36
centrality by protocol, 52–53, 58–59
creating graphs in, 32–34, 34*f*
degree centrality, 38
greedy coloring, 212, 214–215
network analysis graphs, 62–63
packet analysis, 50

social network analysis, 70, 75, 80–81, 86–88
social network evolution, 93, 95, 109, 118
NICs (network interface cards), 48
node ancestry, 83–87, 84*f*, 86*f*
node attributes, 32
nodes. *See* vertices
Nominatim, 164–165
`non_edges` function, 118
nonsimple graphs (pseudographs), 32
`normalized` argument, 36
Npcap library, 47
`number_of_cliques` function, 40
NumPy library, xxiii, 55–56, 80, 103, 170, 186

O

obfuscation, 264–265
`Obstacle` class, 235
`Obstacle` objects, 251, 253
`obstacles` attribute, 253
`only_poly1` variable, 155
OpenCellID, 141–160
API structure, 145–148, 147*f*
device tracking application, 148–159, 148*f*
geodesic polygons, 150–153
intersections, 153–157
mapping and comparing results, 157–159, 158*f*
reducing search area, 159
tower locations, 149–150
network interface data, 142–144
proof of concept, 148–159
open source delivery, 265–266
open source intelligence (OSINT), 29–30, 30*f*
OpenStreetMap, 163–164, 168
optparse library, 62–63
`osm` function, 168
out-degree centrality, 37–38, 51–54, 57–59, 77, 85, 101–103, 113, 121
`out_degree` function, 53
`out_edges` function, 59
outliers, 55–56, 60, 181
`overlaps` function, 138
overloaded symbols, 18, 18*f*

P

packaging, 259–260
packet analysis, 46–51, 51*f*
packet capture (pcap) files, 47–48, 50, 61–63
`packet` objects, 50
packets, defined, 121
`packets` variable, 50
Pålsson, Mikael, 213
pandas library, 5, 69–72, 81, 88, 150, 167–169, 191–192
parallel development, 263–264
parallelism
hardware parallelism, 263
process parallelism, 241–245
processor parallelism, 243–245
threading parallelism, 241–243
`partial` function, 151–152
partial functions, 151
`partition` function, 155
path lengths
lowest common ancestor, 85
returning list of average, 117–119
self-loops, 32
small-world phenomenon, 68
paths, defined, 31
PBX (Private Branch Exchange), 54
pcap (packet capture) files, 47–48, 50, 61–63
`pcap_graph` function, 62
perimeter length, 137–138
personally identifiable information, 178
Phil's Game Utilities (PGU) library, 250
physical penetration testing, 212
pickled objects, 203–204
pickle library, 203–204, 251
Pinellas County Sheriff's Office, 177
pip tool, 10–12, 258
planar straight-line graphs, 216
`player_one_turn` function, 115–116
png library, 236
points, defined, 128–129
`Polygon` class, 129
`polygon_geojson` parameter, 165
polygons, 129–132, 130*f*, 131*f*
complex, 130, 132, 134–135, 138, 214, 223–225, 224*f*
concave, 130, 130*f*

polygons *(continued)*
converting `Point` objects to geodesic polygons, 150–153
convex, 130, 130*f*, 132, 136, 187
irregular, 130, 133–136
linear ring, 130–131, 131*f*, 223–225, 224*f*
orthogonal, 213–214
regular, 130
simple, 130–131, 134*f*, 210
`poly_shapes` variable, 170, 172
polytrees, 85
population mean, 120, 123–124
port numbers subgraphs, 52–54
`post_df` objects, 71–73, 76, 107
potential information, 75–76, 78
`predict` function, 200, 202
predictive analytics, 91
`predict_proba` function, 203
preferential attachment, 68, 113–114
preventative security analysis, 90
Private Branch Exchange (PBX), 54
`process_jpg` function, 184, 186
processor parallelism (multiprocessing), 243–245
process parallelism, 241–245
processor parallelism, 243–245
threading parallelism, 241–243
programming and math conventions, 13–23
mathematical notation, 18–22
attribute characters, 22
Boolean notation, 18–20, 19*t*
Greek letters and functions, 22, 22*t*
overloaded symbols, 18, 18*f*
set notation, 20–22, 21*t*
syntactical constructs, 13–18
dictionary comprehensions, 15–17
list comprehensions, 14–15
zipping and unpacking, 17–18
`Proj` class, 151–152
project managers, 264
proof of concept, ix
facial recognition, 188–204
location triangulation, 148–159
minimum viable product vs., 210
network analysis graphs, 61–64
social network analysis, 87–88
social network evolution, 109–124
Voronoi diagrams, 163–173
`protocol_subgraph` function, 52–55
protocol subgraphs, 54–56, 56*f*
proxy networks, 35–36, 35*f*
pseudographs (nonsimple graphs), 32
purity, 181
PyArmor, 264–265
PyGame library, 236–240, 246–250
events, 237–239
graphic elements, 246–247, 247*f*
PyInstaller, 259–260
PyPi, 258–259, 266
Pyplot library, 32–33
pyproj library, 150–152
Pythagorean theorem, 18, 18*f*, 164
Python
environment setup, 3–12
hardware requirements, 3
installing Anaconda, 4
Jupyter Notebooks, 11–12
Spyder IDE, 10–11
virtual environment setup, 9
virtualenv package manager, 10–11
interpreters, packaging with, 259–260
reasons for using, xxii–xxiii
shortcomings of, xxiii

Q

`Queue` class, 244

R

`-r` (`--raw-out`) parameter, 63
RA (resource allocation), 108–109
`randint` function, 201
`RandomForestClassifier` class, 201
`RandomForestClassifier` objects, 202–203
random forests, 179, 182, 201–203
`random_layout` function, 51
random walks, 95–104, 117–119, 125
biased, 97, 100, 104, 111
Monte Carlo simulations and, 97–99, 99*f*

social network simulation, 100–104
state machines and, 96–97, 96*f*, 99*f*
uniformly, 96–97, 96*f*, 101–104, 117
`range` function, 16–17, 22
`RangeIndex` property, 71
ratios
correlation ratio, 197–198
facial recognition, 190
information exchange ratio, 59–61
`rdpcap` function, 50, 62
`read_csv` function, 167, 192
`read_weighted_edgelist` function, 63
recursive functions, 31
Red Hat, 266
regression problems, 176
repeated-sampling algorithms, 98
`representative_point` function, 223
residual information (RI) score, 75–76
`resize` function, 184
resource allocation (RA), 108–109
resource planning problems. *See* art gallery problem application; emergency service planning scenario
`return_results` parameter, 195
`reverse` option, 53
`reverse` parameter, 59, 137
RI (residual information) score, 75–76
`right_click` function, 241
rings. *See* linear ring polygons
`Room` class, 249, 251, 253–254
root node, 84–85, 180–181
`row_to_str` function, 167
`r_posts` objects, 76

S

`-s` (`--graph-out`) option, 63
sample mean (empirical mean), 93
sample size determination, 98
Saved state, 235
`save_file` function, 227
`save_graph` function, 62
`save_packet` function, 62
`save_project` function, 227–228
saving and reloading data, 251–254
loading from JSON files, 252–254
saving to dictionaries, 251–252
scan codes, 239–240
Scapy library, 47, 50, 62–63
scikit-learn, xxiii, 196–197, 199–201, 203
SciPy library, 55, 80, 170
`scored_neighbor_select` function, 106
`scores` parameter, 111
`screen` attribute, 249–250
`seed` argument, 33–34
seeds. *See* generators
`select_dtypes` function, 192
self-looping, 31–32
`Series` objects, 71, 191, 198
`setattr` function, 253
`set_colorkey` function, 249
`set_file` method, 242
set generator notation (SGN), 22
`set_mode` function, 246
set notation
overview of, 20–22, 21*t*
reserved sets, 21, 21*t*
set generator notation, 22
`set_region_areas` function, 227
setup scripts, 258–259
setuptools library, 258
7Zip, 252
SGN (set generator notation), 22
Shapely library
art gallery problem application, 216, 219, 222–223, 235
computational geometry theory, 128–129, 131–132, 134, 138
emergency service planning scenario, 164, 170
facial recognition, 183, 188, 190
location triangulation, 144, 148–150, 152, 154
`shape_to_np` function, 186
shell layout, 58, 58*f*
Shewchuk, Jonathan, 216
`shifted` variable, 239, 241
shoelace algorithm, 136
`shortest_path_scores` function, 117–119
sides (edges), in polygons, 129
signature detection, 47
`simple` function, 151
simple graphs, 32, 85
single point of failure, 109–110
sink node, 110, 113, 117–119, 121
six degrees of separation, 68

small_area parameter, 157
small-world experiments, 68
Snort, 48
Snow, John, 163
social network analysis (SNA), 67–90
  cautions regarding, 89
  converting data to graph, 69–73
    building graphs, 72–73, 74*f*
    examining data, 69
    structuring data, 69–72
  defined, 67
  proof of concept, 87–88
  research questions, 74–87
    cliques and most influential users, 76–78, 78*f*
    information propagation, 74–76
    most influenced users, 78–79, 79*f*
    node ancestry, 83–87, 84*f*, 86*f*
    topic-based information exchange, 79–83, 82*f*
  small-world phenomenon, 68
social network evolution, 91–125
  finite state machines, 93–95, 94*f*
  information flow game, 110–124
    edge capacity, 112–113
    game phases, 113–117
    game simulation, 118–120
    improvements to player 2, 120–124
    source and sink node selection, 117–118
    weighted random choice, 111–112
  Monte Carlo simulations, 92–93, 97–100, 99*f*
  proof of concept, 109–124
  random walks, 95–97, 96*f*, 99*f*
  simulations, defined, 92
social network graphs, 29, 39–40, 39*f*
social network simulation, 100–109
  information flow distance, 100–104
  resource allocation, 108–109
  topic-based influence, 104–108
soft classifiers, 203
sorted function, 53, 137, 198
source node, 36, 73, 78, 110, 113, 117–119, 121
spaghetti models (storm path maps), 93
sparse adjacency matrices, 80
spring_layout function, 33, 82–83
Sprite class, 248–250, 250*f*
sprites, 248–250, 250*f*
Spyder IDE, 4, 10–11
Stack Overflow, 153
Ståhl, Joachim, 213
Stanford University, 98, 125
Started state, 234–235
Start state, 234–235
state machines
  finite state machines, 93–95, 94*f*, 100–101
  state machine graphs, 30, 31*f*
state managers
  application state planning, 235
  development of, 237–241
  event driven nature of, 237
  purpose of, 235, 237
Steiner points, 219, 228–229
stochastic FSMs, 95, 97
storm path maps (spaghetti models), 93
strip function, 70
substates, 238
sum function, 59
Super Bowl XXXV, 176–177
supervised learning, 176, 179
Surface class, 246–248, 247*f*
surface_size attribute, 249
sweep line algorithms, 154–155
syntactical constructs, 13–18
  dictionary comprehensions, 15–17
  list comprehensions, 14–15
  zipping and unpacking, 17–18

T

target parameter, 244
TCP handshake graphs, 42, 42*f*
TCP packets, 49–50, 61
terminal states, 96, 99–100
term_subgraph function, 106–107
tessellation, 133–136, 134*f*
  art gallery problem, 214, 216–219, 223–224, 226–228, 245, 252
  facial recognition, 189–190
  Voronoi tessellation, 162–173
theil_u parameter, 195

Theil's U, 195
Thread class, 242
threading parallelism, 241–243
three-color problem, 212–213
tiles, 133
timeline function, 88
to_dict function, 251, 253–254
tol parameter, 81
topic-based influence, 104–108
topic-based information exchange, 79–83, 82*f*
topography, 33, 113
topological ordering, 85
touches function, 138
towers (base stations). *See* base stations
transform function, 151–153
transitions, 31*f*, 94–96, 99–101
travel graphs, 28–29, 29*f*, 32
Triangle library, 216, 219, 223–229, 236, 245, 252, 262
Triangle Solver service, 261*f*, 262
triangulate function, 134, 136, 216–217, 219, 221, 223–224, 226–227
truth tables, 19
ttest_ind function, 123
Twitter, 80
two-sample t-testing, 123, 124*f*
type function, 203

U

UDP packets, 49–50
UML (Unified Modeling Language), 234
unbalanced exchange, 75
undirected graphs, 31–32
  betweenness centrality, 36
  cliques, 39–40, 39*f*, 76–77
  connected components, 41
  creating in NetworkX, 33, 34*f*
  degree centrality, 37
undirected preferential attachment (UPA), 114
unicode attribute, 239
Unified Modeling Language (UML), 234
uniform argument, 200
uniformly random walks, 96–97, 96*f*, 101–104, 117
uninstall functions, 257–259
unique function, 103
University of Essex, 177
University of Washington, 212
unsupervised learning (exploratory analysis), 176
unweighted graphs, 32, 43, 214
Unwired Labs, 142, 147
UPA (undirected preferential attachment), 114
urlencode function, 165
user interaction mapping, 234–237, 234*f*
  application state planning, 235–236
  documentation, 236–237
User Interface service, 261*f*, 262–263
user_to_series function, 70

V

vertex-coloring problem, 212
vertex_list attribute, 249
vertices (nodes)
  defined, 129
  graph theory, 27
  vertex order, 132
VirtualBox, 260
virtual environment setup, 9, 63
virtualenv package manager, 10–11
Voice over IP (VoIP), 54, 57
Voronoi diagrams, 161–174
  emergency service planning scenario, 163–173
    city shape, 164–167, 165*f*
    distance function, 164
    generators, 167–169, 169*f*
    tessellation, 170–173, 171*f*, 172*f*
  limitations of, 173–174
  proof of concept, 163–173
  tessellation, 162–163, 162*f*
voronoi_regions_from_coords function, 170
Voronoi tessellation, 162–163, 162*f*, 170–173, 171*f*, 172*f*

W

weighted_choice function, 106, 111–112, 114, 118
weighted graphs, 32–33, 52–53, 55, 62–63, 77

weighted random choice, 95, 97, 106, 111–115
`weight` parameter, 53, 77
WGS (world geodesic system), 152
`wgs84_to_aeqd` function, 152
`where` function, 55, 60
WiGLE, 144, 159
Windows
  frozen delivery, 260
  installing Anaconda, 6–8, 6*f*, 7*f*
  Jupyter Notebooks, 11–12
  network card in promiscuous mode, 63–64
  packet capture library, 47
  setting up virtualenv, 10
  Spyder IDE, 11
  temp directory, 252
WinPcap library, 47
WinPython, 11
WireShark, 46–47
world geodesic system (WGS), 152
write once, read many (WORM) workflow, 62
`write_weighted_edgelist` function, 62–63
`wrpcap` function, 62
`wrs_connect` function, 114, 116
`wrs_disconnect` function, 115–116

X

`X_test` variable, 200
`X_train` variable, 200

Y

`y_test` variable, 200
`y_train` variable, 200

Z

Zenmap, 46, 46*f*
zero-sum games, 98
`ZipFile` class, 252
`zip` function, 17, 197, 202
`zipf` variable, 252
zipping and unpacking, 17–18
`zscore` function, 55–56
Zychlinski, Shaked, 195