

INDEX

Note: *Italicized page numbers locate definitions of terms.*

NUMBERS AND SYMBOLS

7×7 magic square, testing code
for, 33

% (modulo) operator
Euclid's algorithm, 21
Kurushima's algorithm, 27–28
RPM (Russian peasant
multiplication), 19
rules, 32

[] (square brackets)
using with list comprehension,
152
using with loc functionality, 19

A

acceleration
estimating for thrown ball, 10
observing for thrown ball, 9
AI (artificial intelligence). *See also*
decision trees; game trees;
random forests
adding enhancements, 199
drawing the board, 187–188
game trees and winning games,
190–199
la pipopipette, 186–187
representing games, 188–189
scoring games, 189–190
algebra, 5
algorithmic approach
Chapman's algorithm, 9–10
thinking with your neck, 6–9

algorithms, 13
adding theoretical
precision, 63–64
alpha–beta pruning, 199
avoiding use of, 48–49
Babylonian, 90
Bowyer-Watson, 136
comparing to functions, 60–63
counting steps, 57–60
divide and conquer, 69
doing more with, 202–203
finding maximum, 42
gaining expertise, 209
measuring efficiency, 55–57
measuring time, 57
merging sorted lists, 67
minimax, 195–198
performing “by hand,” 14–18,
20–21
perturb search, 112
refraining from using, 48–49
solving problems with, 10–11
tax rates, 39
using big O notation, 64–65
within algorithms, 17
Al-Khwarizmi, 5, 10
alpha–beta pruning algorithm, 199
analytic approach
Galilean model, 2–4
inner physicist, 5–6
solve-for- x strategy, 4–5
angle, tangent of, 8–9

annealing, process of, *117*
 antidiagonal of square matrix, 26–27
 append() method, RPM (Russian peasant multiplication), 18
 arguments, magic squares, 31–34
 artificial intelligence (AI), *185–186*
 adding enhancements, 199
 drawing the board, 187–188
 game trees and winning games, 190–199
 la pipopipette, 186–187
 representing games, 188–189
 scoring games, 189–190
 asymptote, relationship to maximum, 39–40

B

Babylonian algorithm, 90
 ball. *See also* the outfielder problem
 horizontal position of, 7
 plotting trajectory of, 1–2, 4, 7
 tangent calculation, 8–9
 ball_trajectory() function, 3–4
 baseball, scientific features of, 6
 bell curve, *95–96*
 between_spaces variable, creating, 154
 big O notation
 sleep sort’s runtime, 72
 using, *64–65*
 billiard balls and randomness, 91
 binary branching process, using
 with decision trees, 166–167
 binary expansion, *17*
 binary search, *73–75*
 bisect, geometric terminology, *130*
 bits, string of, *97–98*
 board, drawing for dots and boxes game, 187–189
 bootstrapping, *91*
 Bowyer-Watson algorithm, 136.
 See also DT (Delaunay triangulation);
 triangulation
 brain, “wetware” of, 5
 branching process, using with
 decision trees, 166–167

brute force solution, using in
 TSP (traveling salesman problem), 107
 Bush, Vannevar, 6

C

calculus, rules of, 38
 centroid of triangle, finding, *131–133*
 Chapman, Seville, 6
 Chapman’s algorithm, 9–11. *See also*
 the outfielder problem
 chatbot, building, 203–208
 chess, solving eight queens puzzle, 209–212
 Chesterton, G. K., 151
 circles, drawing, 133
 circumcenters
 finding for triangles, 131–133
 plotting, 145
 relationship to triangles, 134
 circumcircles
 plotting, 145
 relationship to triangles, *132, 134*
 combinatorial explosion, using in
 TSP (traveling salesman problem), 108
 compound words, dealing with, 152–153. *See also* words
 constructive methods of Euclid, *20*
 continued fractions. *See also*
 fractions to radicals
 algorithm for generating, 82–85
 compressing and communicating Phi, *79–80*
 versus decimals, 86–88
 overview, 78, *80–82*
 to radicals, 88
 continued square roots, 88
 corpus, *149, 160. See also* imported corpus
 cosine similarity, *206–208*
 Counter() function, using with
 n + 1-gram, 161
 counting steps, *57–60*

D

decimals to continued
 fractions, 86–88

decision trees. *See also* AI (artificial intelligence) game trees; machine learning; random forests

- adding depth to, 175–177
- building, 167
- calculating happiness levels, 170
- choosing split points, 182
- choosing splitting variables, 173–175, 182
- downloading datasets, 168
- evaluating, 178–182
- looking at data, 168–169
- nodes, 167
- out-of-sample observations, 180
- overfitting, 181–182
- overview, 165–166
- prediction errors, 171–172
- problem of overfitting, 179–181
- pruning, 182, 199
- in-sample observations, 180
- simplifying, 181–182
- split points, 171
- splitting data, 169–173
- test sets, 180
- training sets, 180
- underfitting, 181–182
- using nested lists with, 176

Delaunay triangulation (DT).
 See also geometry

- generating, 136–139
- implementing, 139–143
- overview, 134–136
- purpose of, 136
- returning from points, 142
- to Voronoi, 143–147

derivative, calculating, 38

Devlin, Keith, 5–6

dictionary object, creating for chatbot, 203

Diehard tests for randomness, 95–97

divide and conquer algorithm, 69

dogs, catching Frisbees, 6

dots and boxes game. *See also* games

- drawing board for, 187–188
- playing, 186–187
- scoring, 190

doubling column, RPM (Russian peasant multiplication), 14–20

down_left, Kurushima’s algorithm, 28–29

drawgame() function, using with games, 188–189

drawing circles, 133

drawlattice() function, using with games, 188–189

DT (Delaunay triangulation).
 See also Bowyer-Watson algorithm; triangulation

- generating, 136–139
- implementing, 139–143
- overview, 134–136
- purpose of, 136
- returning from points, 142
- to Voronoi, 143–147

E

education and lifetime income, 42–45

Elements, 20

equilateral, geometric terminology, 130

ESS (European Social Survey), using with decision trees, 168

Euclid’s algorithm, 20–22, 84–85

exclusive OR operation, 98

exponential function, 60–61

F

False, Kurushima’s algorithm, 27

feedback shift register, 98

file-sorting method, 52–54. *See also* sorted filing cabinets

fillsquare() function, Kurushima’s algorithm, 31–32

finding words, 151–152

finditer() function, using with words, 152

`findnearest()` function, using in TSP (traveling salesman problem), 109
`float('nan')` function, using with Kurushima's algorithm, 24
`floor()` function, using for binary search, 73–74
for loop, using with words and spaces, 157
fractions to radicals, 88. *See also* continued fractions
Franklin, Benjamin, 126
Frisbee, trajectory vectors, 6
functions
 inverting, 75
 recursion, 22

G

Galilean model, 2–5
game trees. *See also* AI (artificial intelligence); decision trees; random forests
 building, 192–195
 and winning games, 190–192
games. *See also* dots and boxes game
 choosing moves, 195–198
 minimax algorithm, 195–198
 representing, 188–189
 scoring, 189–190
 winning, 195–198
Gaussian normal curve, 96
`gen_delaunay()` function, passing x and y values to, 143
`generate_tree()` function, using with games, 194
`genlines` function, using with triangles, 129
`genlines` function, TSP (traveling salesman problem), 104
geometry. *See also* DT (Delaunay triangulation)
 postmaster problem, 126–128
 representing points, 128
 tangent of angle, 8–9
 terminology, 130
 triangles, 128–134
`get_number()` function, using with continued fractions, 85

`get_prediction()` function, using with decision trees, 178–179
`get_split()` function, using with decision trees, 174–176
`get_splitpoint()` function, using with decision trees, 174
`git bisect` software, using for binary search, 75
global variables, defining for simulated annealing, 122
golden ratio, 78–79
gradient ascent, 35
 climbing income hill, 44–45
 implementing, 40–41
 local extrema, 42–44
 objections, 41–42
 using, 49
gradient descent, 35, 47
Gravity's Rainbow, 3
greedy algorithms, TSP (traveling salesman problem), 112–113
guided search, using in TSP (traveling salesman problem), 112

H

`half_double` dataframe, RPM (Russian peasant multiplication), 18
halving column, RPM (Russian peasant multiplication), 14–20
happiness levels, calculating with decision trees, 170
hill climbing, 47–48
`howfull` argument, Kurushima's algorithm, 31–32

I

if statement
 inserting `pop()` function into, 66–67
 using with words and spaces, 151
imported corpus, using to check for valid words, 154–155. *See also* corpus

inner physicist theory, 5–6
in-sample observations, using with
 decision trees, 180
insert() function, using with bits, 98
insertion sort, 52–55
 comparing to exponential
 function, 61
 counting steps in, 63–64
 step counter, 58
installing, matplotlib module, 3
integers, dividing to get
 quotient, 84
inverse_sin(0.9) function, using for
 binary search, 75
inverting functions, 75
irrational number, 79

J

Japanese magic squares. *See also*
 magic squares; squares
 Kurushima's algorithm in
 Python, 24–30
 Luo Shu square in Python,
 22–23

K

Kepler, Johannes, 78
k-means machine-learning
 method, 56
k-NN machine-learning method, 56
Kurushima's algorithm
 function, 30–31
 rules, 25–28

L

la pipopipette, 186–187
language algorithms
 difficulty, 150
 phrase completion, 159–163
 space insertion, 150–158
lattice, using with la pipopipette,
 186–187
LCGs (linear congruential
 generators), 92–93
left and right variables, Python, 66
Leibniz, Gottfried Wilhelm,
 130–131

LFSRs (linear feedback shift
 registers), 97–99
lifetime income and education,
 42–45
lines of sight, plotting for thrown
 ball, 7–8
list comprehensions, 149, 156
list indexing syntax, Python, 68–69
lists, sorting, 153
loc functionality, RPM (Russian
 peasant multiplication), 19
local extrema, problem, 42–45
loops, RPM (Russian peasant
 multiplication), 18
lower bound, defining for binary
 search, 73
lower() method, using with
 chatbot, 203
Lucas, Édouard, 186
Luo Shu square, creating in
 Python, 22–23

M

machine learning. *See also* decision
 trees
 overview, 165
 random forests, 182–183
machine-learning methods,
 k-means clustering and
 k-NN, 56
magic eye, 147
magic squares, 22–23. *See also*
 Japanese magic squares;
 squares
 arguments, 31–34
 Kurushima's algorithm, 30–31
 of odd dimension, 24
 patterns, 34
 “walk” through, 28
*The Math Instinct: Why You're a
 Mathematical Genius (Along
 with Lobsters, Birds, Cats, and
 Dogs)*, 5–6
math library, Python, 73–74
mathematical physics,
 interpretation of, 92
math.floor(), RPM (Russian peasant
 multiplication), 18

- matplotlib module
 - setting tax rates, 36–37
 - using with dots and boxes game, 187–188
 - matplotlib module, installing, 3
 - max() function, using with numpy, 162
 - maxima and minima, 35
 - maximization and minimization, 45–48
 - maximum
 - and asymptote approach, 39–40
 - education and lifetime income, 44–45
 - and minimum of step values, 60–61
 - revenue, 39
 - solving first-order conditions, 42
 - taxation/revenue curve, 41–42
 - maxitin argument, adding, 122
 - merging to sorting, 65, 68–70. *See also* sorting
 - Mersenne Twister PRNG, 99
 - metaheuristics, metaphor based, 117–118
 - Mikami, Yoshio, 22
 - Millennium Prize Problems, 212
 - minimax algorithm
 - using to make decisions, 199
 - using to win games, 195–198
 - minimax() function, calling, 198
 - modulo (%) operator
 - Euclid’s algorithm, 21
 - Kurushima’s algorithm, 27–28
 - RPM (Russian peasant multiplication), 19
 - rules, 32
 - Monte Carlo methods, 199
 - mystery number and continued fraction, 81
- N**
- $n + 1$ -grams, finding, 161–163
 - n queens completion problem, solving for chess, 210–211
 - nan entries, filling in, 25–28, 30–31
 - Navier-Stokes equations, 5
 - nearest neighbor algorithm, TSP (traveling salesman problem), 108–110
 - nested lists, using with decision trees, 176
 - nested radicals, 88
 - next_random() function, 93
 - n-gram, tokenizing and getting, 159–160
 - Norvig, Peter, 160
 - NP (nondeterministic polynomial) complexity class, 212–213
 - numbered file, inserting, 54
 - numpy module
 - importing, 60
 - using to select phrases, 162
 - using with decision trees, 180–181
- O**
- optimization, 101–102. *See also* simulated annealing; TSP (traveling salesman problem)
 - the outfielder problem, 1–2, 6–9. *See also* ball; Chapman’s algorithm
 - out-of-sample observations, using with decision trees, 180
 - overfitting decision trees, 181–182
 - overlapping sums test, 95–96
- P**
- P complexity class of problems, 212–213
 - pandas module, using in Python, 19
 - percentile, using with decision trees, 172–173
 - perpendicular, geometric terminology, 130
 - perturb() function
 - modifying, 116
 - showing end of, 121
 - updating, 119
 - using for simulated annealing, 123
 - using in TSP (traveling salesman problem), 111–112

perturb search algorithm, 112. *See also* simulated annealing

phi

- compressing and communicating, 79–80
- and golden ratio, 78

phrase completion, 159–163

plot() function, using with dots and boxes game, 187–188

plot_triangle() function

- defining, 129
- improving, 133–134

plotitinerary() function, using in TSP (traveling salesman problem), 105

plotting capabilities, Galilean model, 3

.png file, saving to, 129–130

points, representing, 128–130

points_to_triangle() function

- defining, 128
- using in triangulation, 134

polynomial, Galilean model, 3

polynomial time, verifying solutions in, 212

pop() method

- inserting into if statements, 66–67
- using with bits, 98

pop() method, sorting via insertion, 55

postmaster problem, 126–128

potential words. *See also* words

- checking for, 153–154
- finding halves of, 156–158

prediction errors, decision trees, 171–172

print(cities) function, TSP (traveling salesman problem), 103

print(lines) function, TSP (traveling salesman problem), 104

print(square) function, using with Kurushima’s algorithm, 24–25

PRNGs (pseudorandom number generators), 92–99

problems, solving with algorithms, 10–11

Project Gutenberg, 160

pruning decision trees, 182, 199

pseudorandomness, 92–93

Pynchon, Thomas, 3

Pythagorean theorem

- using, 105
- using with triangles, 130
- using in TSP (traveling salesman problem), 108–109

Python

- creating Luo Shu square, 22–23
- Euclid’s algorithm, 20–22
- feedback shift register, 98
- Galilean model, 3
- implementing RPM (Russian peasant multiplication), 18–20
- Kurushima’s algorithm, 24
- left and right variables, 66
- list indexing syntax, 68
- math library, 73–74
- ordered pairs in, 152
- overlapping sums test, 95–96
- pandas module, 19
- random module, 58–59
- random.choice() function, 28
- rules for Kurushima’s algorithm, 27–28, 30–31
- square roots in, 90–91
- timeit module, 57
- using tuples with words and spaces, 152

Q

quotient, getting by dividing integers, 84

R

radicals and fractions, 88

radius, returning for triangle, 132–133

Ramanujan, Srinivasa, 88

random forests, 182–183. *See also* decision trees; game trees

- random model, Python, 58–59
- random number generators
 - judging PRNGs
 - (pseudorandom number generators), 93–95
 - LCDs (linear congruential generators), 92–93
 - LFSRs (linear feedback shift registers), 97–99
 - overview, 91
- random.choice() function, Python, 28
- randomness
 - Diehard tests for, 95–97
 - possibility of, 91–92
- random.seed() function, 59
- recursion
 - of functions, 22
 - implementing merge sort
 - with, 69
 - using with Euclid’s algorithm, 85
- re.finditer() function, using with words, 152
- reindex() method, using with decision trees, 181
- remove() function, using with words and spaces, 155
- replace() function, using with words and spaces, 155
- resetthresh variable, adding, 122
- revenue
 - maximum, 39
 - showing for tax rates, 36–37
- right and left variables, Python, 66
- RPM (Russian peasant multiplication), 13–20
- rules, applying with Kurushima’s algorithm, 27, 30–31

S

- science, laws of, 130–131
- scoring games, 189–190
- search suggestions, strategy for
 - generating, 160, 162–163
- searching versus sorting, 72–75
- Shakespeare’s works, accessing, 160–161, 163
- siman() function, using for simulated annealing, 122–123
- Simmons, Joseph, 179
- simulated annealing, 115–124. *See also* optimization; perturb search; TSP (traveling salesman problem)
- sleep sort, 70–72. *See also* sorting
- Smith, David Eugene, 22
- solve-for-*x* strategy, 4–5, 10–11
- sorted filing cabinets, merging, 62, 64–65. *See also* file-sorting method
- sorting. *See also* merging to sorting; sleep sort
 - lists, 153
 - via insertion, 54–55
 - to searching, 72–75
- space insertion
 - checking for potential words, 153–154
 - checking for valid words, 154–156
 - dealing with compound words, 152–153
 - defining word lists, 151–152
 - finding halves of potential words, 156–158
 - finding words, 151–152
 - overview, 150–151
- spaces
 - getting substrings between, 153–154
 - inserting into texts, 158
 - words ending with, 156
- split points, choosing for decision trees, 171, 182
- splitting variables, choosing for decision trees, 182
- square brackets ([])
 - using with list comprehension, 152
 - using with loc functionality, 19
- square matrix, antidiagonal of, 26–27
- square roots, 89–91
- squares, filling in, 30–34. *See also* Japanese magic squares; magic squares
- start() function, using with words, 153

- statistical methods, bootstrapping
 - as, 91
- steps
 - counting in insertion sort, 57–60, 63–64
 - exponential growth, 60–61
- stochastic gradient ascent, 45
- strings, splitting into words, 159–160
- substrings, getting between spaces, 153–154
- sudoku puzzles, solving, 211–212

T

- tangent of angle, 8–9
- tax rates, setting, 36–41
- taxation/revenue curve, gradient ascent, 41
- tax/revenue curve, flipping, 46–47
- temperature function, TSP (traveling salesman problem), 113–115
- test sets, using with decision trees, 180
- text normalization, using with chatbot, 203
- text vectorization, 204–206
- TFIDF (term frequency-inverse document frequency) method, 204–205, 207–208
- theta, applying to thrown ball, 8–9
- thinking with your neck, 6–9
- time, measuring precisely, 57
- timeit module, Python, 57
- Titanic* lifeboat example, using sleep sort with, 71–72
- tokenization, performing with chatbot, 204
- tokenizing n-grams, 159–160
- training sets, using with decision trees, 180
- translate() method, using with chatbot, 203–204
- triage and decision trees, 166
- triangles
 - centroid, 131–133
 - creating for postmaster problem, 128–134

- finding circumcenter of, 131–133
- plotting, 129, 145–146
- replacing, 140–143

triangulation. *See also* Bowyer-Watson algorithm; DT (Delaunay triangulation)

- defined, 134
- of seven points, 135

True, Kurushima's algorithm, 27

TSP (traveling salesman problem). *See also* optimization; simulated annealing

- greedy algorithms, 112–113
- improving, 110–112
- nearest neighbor algorithm, 108–110
- overview, 102–103
- versus postmaster problem, 127
- setting up, 103–108
- temperature function, 113–115

tuples, using with words and spaces, 152

U

- underfitting decision trees, 181–182
- up_right, Kurushima's algorithm, 28–29
- upper bound, defining for binary search, 73

V

- vector similarity, determining, 206–208
- vertex, geometric terminology, 130
- Voronoi diagram
 - generating, 143–147
 - for postmaster problem, 128

W

- while loop, Kurushima's algorithm, 31
- while loop
 - using for binary search, 74
 - using with bits, 99
 - using with continued fractions, 85

- using with merge sort, 67
- using with square roots, 90–91
- while loop, RPM (Russian peasant multiplication), 18
- winning games, 195–198
- word list, defining, 151–152
- words. *See also* compound words;
potential words
 - checking validity with imported corpus, 154–156

- ending with spaces, 156
- finding, 151–152
- tokenizing, 159–160

X

- XOR operation, 98