

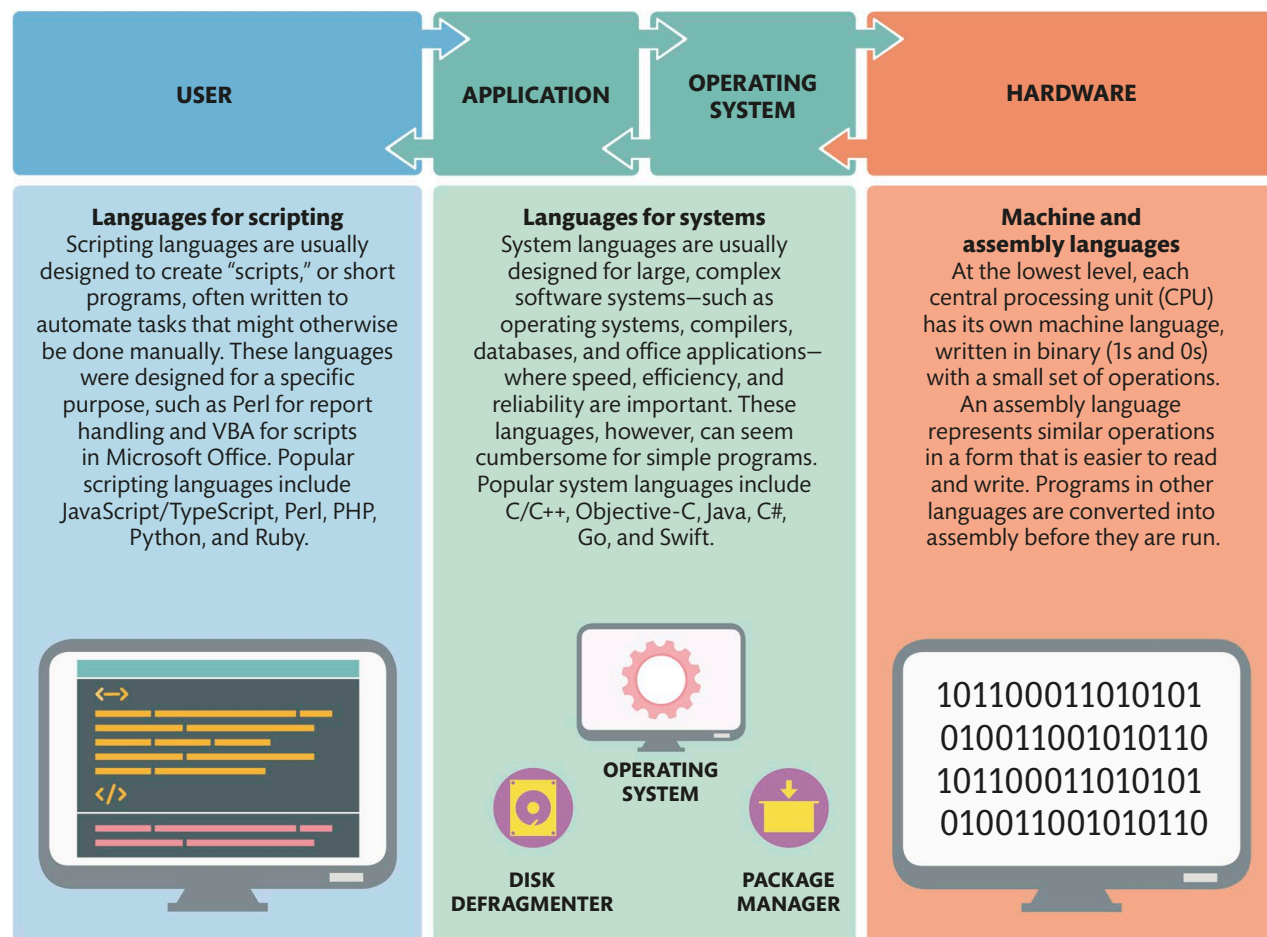
Other programming languages

Every profession has its own vocabulary and ways to describe common problems and solutions. Programming languages were developed to help humans communicate with computers. Most languages are designed for a specific task or domain but are often adapted for other purposes.

Grouping programming languages

Human languages are grouped into families (such as Germanic or Dravidian) that use similar alphabets, vocabulary, and structures. If you know one language in a family, it is easier to learn others.

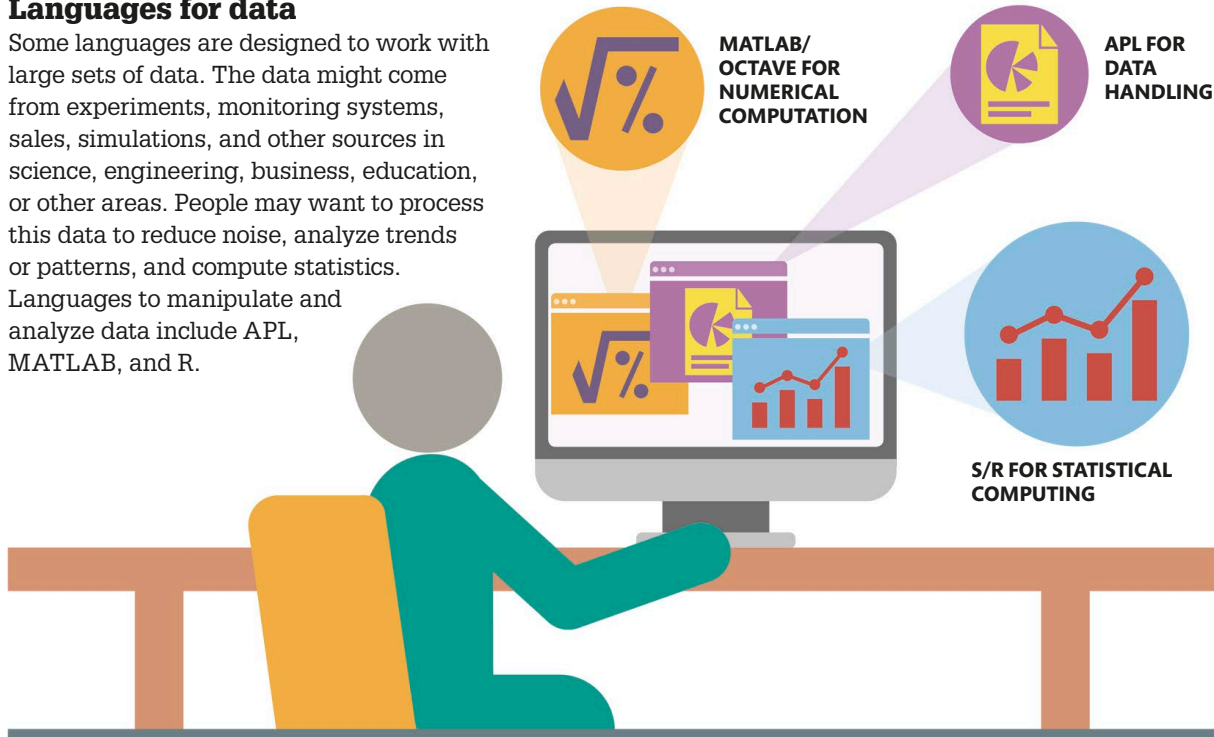
Programming languages are also grouped into families and often borrow words and structures from each other. For example, C, C++, Objective-C, Java, C#, Go, and Swift are all related, so developers who know one of them can learn the other languages more easily.





Languages for data

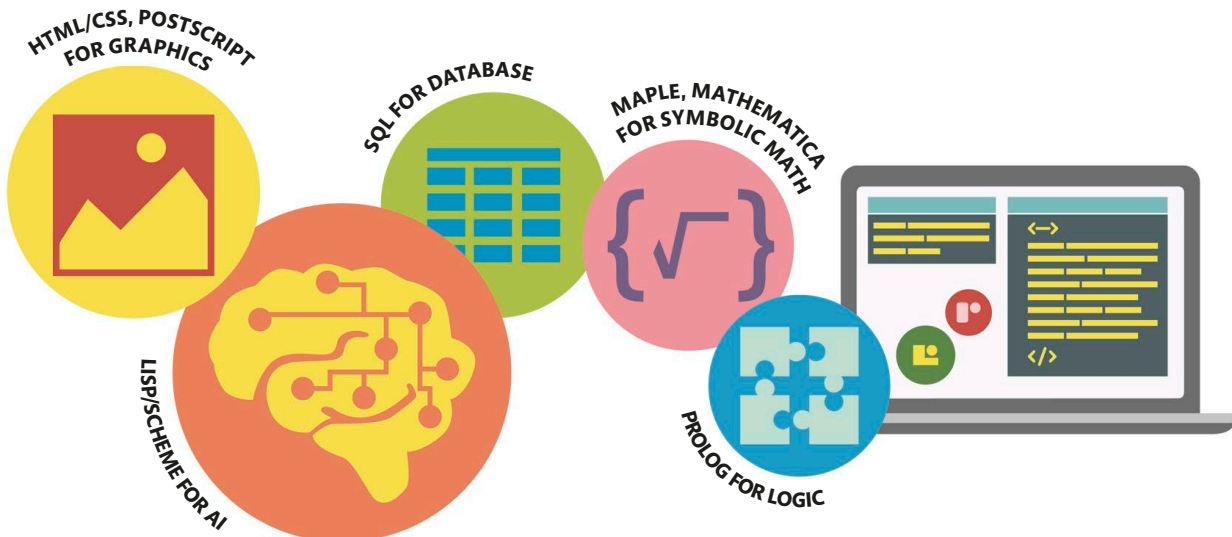
Some languages are designed to work with large sets of data. The data might come from experiments, monitoring systems, sales, simulations, and other sources in science, engineering, business, education, or other areas. People may want to process this data to reduce noise, analyze trends or patterns, and compute statistics. Languages to manipulate and analyze data include APL, MATLAB, and R.



Languages for special purposes

Some programming languages are designed to solve specific problems and might not be useful in other areas. PostScript, TeX, and HTML describe the content and layout of pages with text, images, and

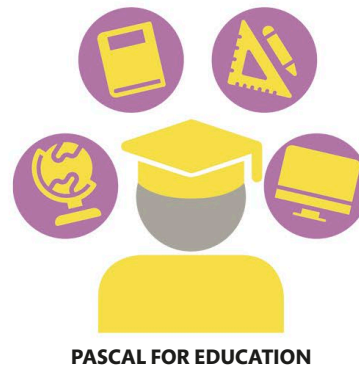
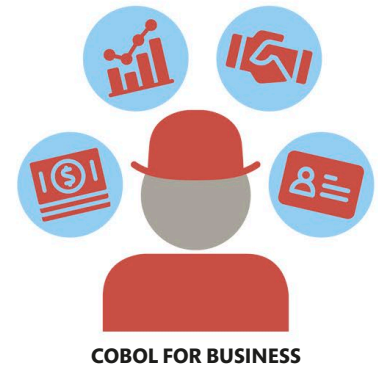
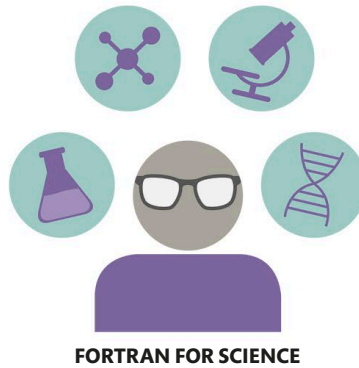
other information. SQL is used to manage databases. Maple and Mathematica are used for symbolic mathematics. LISP and Scheme are useful for AI (artificial intelligence). Prolog is used for logic programming.



EARLY PROGRAMMING LANGUAGES

Early programming languages

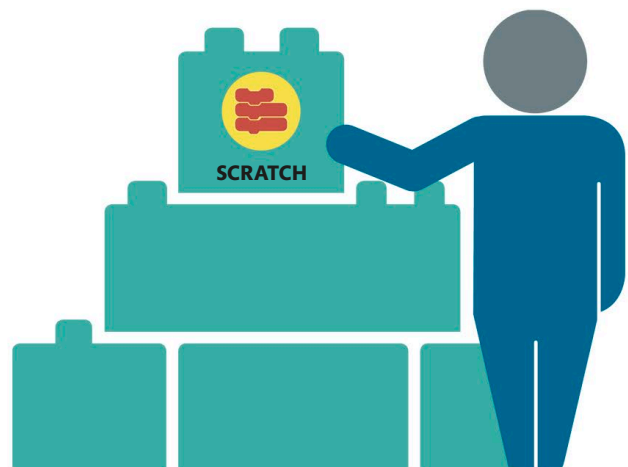
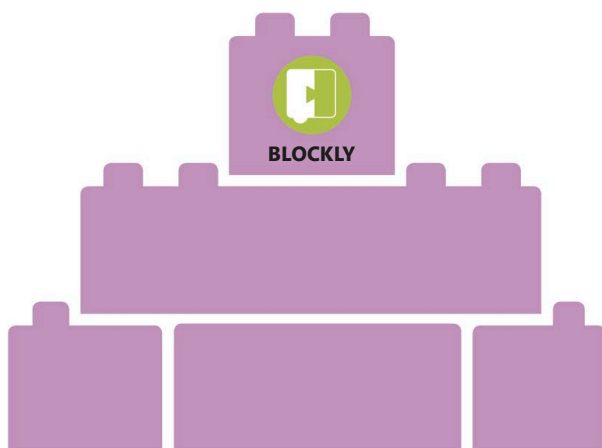
There are other widely used languages, some with a long history. Fortran was created in the 1950s for scientific and engineering applications. COBOL was created around 1960 for business applications. BASIC was an easy language created in the mid-1960s for students. Pascal was created around 1970 to encourage structured programming practices and was widely used in education. Ada was created around 1980 to reduce the number of different languages used across the US Department of Defense (DoD).



Visual languages

In visual (or block-based) languages, programs are created graphically rather than with text. For example, a user might drag elements into position, connect them, and then type in number values or text

messages. Such languages are often designed for nonprogrammers in specific areas, such as education, multimedia, and simulation. Popular visual languages include Blockly, Alice, and Scratch in education, and Kyma, Max, and SynthEdit for music.





Other programming languages

There are hundreds of programming languages, and most developers are proficient in a few, familiar with more, and expect to learn new languages throughout

their careers. The table below lists some of these programming languages, with information on when they were first developed, the lead creator, and a brief summary of key ideas and major uses.

POPULAR PROGRAMMING LANGUAGES	
Language, date, and creator	Key ideas and major uses
C (1972) Dennis Ritchie	Designed to be concise and portable and to generate efficient machine code. It is widely used for operating systems, compilers, interpreters, and large applications. Many other languages have adapted features and syntax from C.
C++ (1983) Bjarne Stroustrup	Designed to add object-oriented features to C (in C, "C++" adds 1 to the variable c). Widely used for operating systems, compilers, interpreters, and large applications.
Java (1995) James Gosling	Designed to be an object-oriented language based on C and C++. Java was meant to be a "write once, run anywhere" programming language—code written on one type of computer can be run on other types. Widely used for desktop applications and browser-server applications.
Python (1991) Guido van Rossum	Designed for readability and to support multiple programming styles. Uses a small core language with libraries that add more specialized functions. Widely used in web applications, in scientific computing, and for scripting in other software products.
PHP (1994) Rasmus Lerdorf	Designed for web development ("PHP" originally meant "Personal Home Page"), where it is widely used.
JavaScript (1995) Brendan Eich	Designed to create interactive web pages and applications, where it is widely used. JavaScript is also used in some web servers so that a web application can use the same language in the browser and server.
Fortran (1950s) John Backus	Designed at IBM (International Business Machines Corporation) for scientific and engineering applications, which often involve many numeric calculations. Named from "FORMula TRANslation."
COBOL (1959)	Designed for data processing, COBOL was based on the earlier work of Grace Hopper. It was supported by the US Department of Defense, which led to its wider adoption. Named from "COMmon Business-Oriented Language."
BASIC (1964) John Kemeny and Thomas Kurtz	Designed to be easy to use for students in many fields, not just science and mathematics. It expanded into Microsoft Basic (1975) and Visual Basic (1991). Named from "Beginner's All-purpose Symbolic Instruction Code."
Ada (1980s) Jean Ichbiah	Designed for embedded and real-time systems and to reduce the number of languages used across the US Department of Defense (DoD). Named after Ada Lovelace, often described as the first computer programmer.
SQL (1970s) Donald Chamberlin and Raymond Boyce	Designed to edit and search databases, especially "relational databases" (when data is stored in tables that are related to each other in various ways). SQL is short for "Structured Query Language."

Glossary

algorithm

A sequence of steps or instructions that complete a task or solve a problem. In programming, an algorithm often includes repeated steps, decisions between two or subsequences of steps, and steps that refer to other algorithms to do subtasks or solve subproblems.

API (Application Programming Interface)

A set of definitions that programmers can use to access another system without having to understand all of its details. The definitions might include functions, classes, data structures, and data values. Originally named because it defines an interface for programmers to develop applications using an underlying system. *See also* library.

array

A collection of items stored in adjacent locations in the system's memory, using a single name and a numeric index. The index usually starts at 0. Often, all elements in the array have the same type. For example, all integers or all strings of characters. An array is one way to store a list. *See also* list.

attribute

A specific piece of information associated with a data object. For example, an image would have attributes for height and width, and a sound would have attributes for length and sampling rate.

binary

A numbering system used by computers that has only two digits (0 and 1), not the usual decimal system with ten digits (0 to 9). In binary, each position is two times the position to its right, rather than ten times in decimal. For example, $101101 = 1 \times 32 + 0 \times 16 + 1 \times 8 + 1 \times 4 + 0 \times 2 + 1 \times 1 = 45$.

bit

Shortened from "binary digit," it is the basic unit for information or communication. The value of a bit can be either 0 or 1. Thus, an eight bit device mostly uses storage elements with 8 bits, which can store $2^8 = (256)$ different values.

block element

An HTML element that breaks the flow of text and changes the layout of the page. For example, paragraphs (`<p>`), lists (``, ``, ``), and tables are all block elements. *See also* inline element.

branching statement

A program statement that chooses one of several possible paths or sets of steps, usually based on the value of an expression. For example, an "if-then-else" statement takes the "then" path if an expression is true, and the "else" path if an expression is false. Also called a conditional statement.

Boolean

A value that can be either true or false. Named after George Boole, who defined a logic system based on such values.

bug

A defect or an error in a program or other system that prevents it from working correctly. The term was used in engineering long before computers, but is often associated with a story told by Grace Hopper about a moth stuck in an early computer, causing wrong results.

call

A program statement that causes the computer to run another function, and return to the original function when done.

carousel

A software component in Bootstrap that cycles through a set of elements, like a slideshow.

CDN (Content Delivery Network)

A network of servers spread across different places that can deliver the same content (data or services). For example, when a web browser loads content for a page, the CDN can deliver content from nearby servers, which reduces the wait time and the network traffic.

child object

An object created from a prototype in a parent object. The child shares (inherits) all functions and properties of the parent, but can override them. For example, the parent might define functions and properties for any book, and each child would define the author, title, publisher, and date for a specific book.

class

(1) A definition or description of a category, which usually includes data and functions, and is used to create (instantiate) objects in that category. For example, the class for employees might specify that every employee has a name,

phone, and email address, and provide functions to set or display them. (2) In CSS, a style definition that can be added to any number of elements.

cloud

A set of internet servers that can be used instead of a local computer. Cloud storage stores files and other data, and cloud computing does computation.

compiler

A program that analyzes a computer program and converts (compiles) it into machine code so it runs faster. *See also* interpreter.

composite data

Data that is created by combining other simpler data. For example, a string of characters, an array of numbers, or an object. *See also* primitive data.

concatenate

To combine items, usually character strings, one after another. For example, concatenate "snow" and "ball" to get "snowball."

conditional statement

See branching statement

constructor

A special function used to create new objects of a class. Typically, the constructor allocates memory, initializes variables, and does other setup.

data

Any information stored in or used by a computer.

data binding

Connecting (binding) the data values in two or more objects or systems so that changing one also changes the others. For example, binding a GUI element to a data object ensures that changes to the object appear in the GUI, and GUI changes also change the object.

debug

To remove bugs in a program. This might involve running the program with different inputs, adding statements to print or store values

as the program runs, or watching memory values and how they change. *See also* bug.

directory

(1) Also called a folder; a structure to store files, and subdirectories with other files. (2) A list of resources and how to access them.

ECMAScript

The official definition for the scripting language used in browsers and servers, to provide a standard that could be used by JavaScript, JScript, ActionScript, and other web languages.

element

(1) A single value in a larger set, such as an array. (2) In HTML, a part of a document, often with a start tag, content, and a stop tag. For example, “`DANGER`” is an element that shows “DANGER” as emphasized text.

event

A description of something that has happened, often used as a signal to trigger responses in a program. For example, a mouse-click event could submit a form or display a menu.

execute

Also called run; the command to start a program.

file object

An object that describes or gives access to a file stored in the system’s memory.

float

A number with a decimal point in it. It allows a computer to store very large and very small numbers more efficiently. Also called a floating point number.

flowchart

A graphical way to show the steps, branches, and loops in an algorithm.

framework

A collection of software elements that can be combined and customized in various ways, usually to create user applications. For example, Angular, Django, Express, jQuery, React, and Ruby on Rails are all frameworks used for websites and web applications.

function

Code that carries out a specific task, working like a program within a program. Often, a function has a name, a set of input

parameters to give information to the function, and a result for the code that called the function. Also called procedures, subroutines, and methods (especially in object-oriented languages).

git

A popular version control system, used to track changes in a set of files, so that users can easily collaborate and access different versions of the same file. *See also* version control system.

global variable

A variable that can be used anywhere in a program. *See also* local variable; variable.

GUI (Graphical User Interface)

Often pronounced “gooey,” a user interface is the name for graphical elements, such as buttons, menus, text fields, and checkboxes that make up the part of the program that a user can see and interact with. It is different from a command line interface where everything is displayed as text.

hardware

The physical parts of a computer such as the processor, memory, network connectors, and display. *See also* software.

hack

(1) An ingenious change to code that makes it do something new or simplifies it. (2) To break into other computer systems.

hosting

Also called web hosting; providing server and internet access to clients for their own websites. In dedicated hosting, each client gets their own server; in shared hosting, many clients share a single server.

hover state

The appearance of a GUI element when the cursor or pointer hovers above it. For example, a button or text field might have a different color or border when the mouse hovers above it, to indicate that it is active or ready to use. Also called “mouse-over” state. *See also* normal state.

hyperlink

A text or graphical element that can be clicked, tapped, or otherwise selected to access other information, often using a URL. The other information can be in the same document, another document, or on another website.

index number

A number indicating the position (index) of an element in an array. Many programming languages use square brackets with arrays, so “myArr[3]” means the element in position “3” of array “myArr.”

inline element

An HTML element that does not break the flow of text or change the layout of the page. *See also* block element.

input control

A part of a user interface, such as a button, checkbox, or text field, that allows a user to provide input to a program.

instantiate

To create a new object, usually using its class definition.

integer

A number without a fractional part, also called a whole number. Usually, a computer can represent a large, but not infinite, set of integer values.

interface

A boundary between two parts of a system. Thus, a user interface (UI) is how a user interacts with the system, and an API (Application Programming Interface) is a set of definitions to help programmers develop applications using an underlying system.

internet

The global computer network, which is actually a network of networks. Shortened from “interconnected network.”

interpreter

A program that executes computer programs one statement at a time, without first converting (or compiling) the program to machine code.

iterate

To execute a task or set of statements repeatedly. Most programming languages have special syntax to make it easier for programs to iterate, either a set number of times or until some condition is met. For example, a program might iterate through an array to perform the same actions on each element.

iteration

The general process of iterating, or the process of repeatedly going through a set of statements in the code.

library

A set of resources that can be reused in other projects. These resources might include functions, classes, data structures, and data values. A library is similar to an API. For example, a math library might have a constant value for pi and functions to compute the sine, cosine, and tangent of an angle. *See also* API.

literal

A fixed value written in source code. In most programming languages, integer and real number literals are written normally, and strings of characters are written between quotation marks.

list

A set of data values, where each value has a specific position in the list. One way to store a list is as an array. *See also* array.

local variable

A variable that can only be used with a particular function or other limited part of a program. *See also* global variable; variable.

loop counter

A variable that counts (tracks) the number of times a loop has been repeated.

machine code

The set of instructions that is used by a computer processor. It is difficult for users to read or write machine code, so other programming languages are used with a compiler or interpreter to convert them to machine code.

malware

Short for “malicious software”; any software designed to gain illegal access to a computer or system. Malware includes viruses, worms, spyware, and ransomware.

memory

Storage used by a computer, using a wide range of technologies, including ROM (read-only memory), RAM (random access memory), solid states drives (SSDs), hard disk drives, and optical drives (e.g., CDs or DVDs). In general, faster technologies are much more expensive, so most computers use smaller amounts of expensive memory (RAM) and larger amounts of cheaper memory (hard disk drives).

metadata

Data that describes other bits of data. For example, web pages use metadata to specify the page title, language, and HTML version, while music files use metadata to specify the composer, performer, title, date recorded, style of music, copyright status, and other information.

module

A package of premade code that can be imported into a program, making lots of useful functions available.

network

A set of computers connected together to allow the sharing of data and resources.

normal state

The way a GUI element (for example, a button) appears normally. *See also* hover state.

object

In object-oriented programming, an object is a component that consists of data and code to manipulate the data.

object-oriented

An approach to coding where programs are organized into classes and objects, which typically contain data values and functions that use or change those values.

opcode

Part of a machine code instruction that specifies the operation rather than other information (such as the memory locations) to use. Shortened from “operation code.” *See also* operand.

operand

Parts of a machine code instruction that do not specify the operation, but other information such as the memory locations to use. More generally, a parameter passed to a function. *See also* opcode.

OS (operating system)

The underlying software system that manages resources (both hardware and software) and provides services used by other software. For example, Microsoft Windows, Apple’s macOS, and Linux.

output

The result of a program, which might be displayed on a screen, stored in a file, or sent to another program or computer.

parameter

An input for a function. In most languages, a function definition includes a name for each input. For example, the function “sum(x,y)” has two formal parameters “x” and “y”.

parent object

An object used to create child objects. The parent has a prototype with functions and properties that can be used by each child. *See also* child object.

parse

To take text or other input data and create a more useful structure out of it. For example, a browser parses a file of characters and creates a data structure (called the Document Object Model, or DOM) that shows which elements contain which other elements.

payload

The actual message within a larger communication. For example, when a browser loads a web page, the payload is the actual HTML that will be displayed.

port

(1) A virtual connection point used to contact a specific service or process. (2) To adapt software to run on another operating system or on other hardware.

primitive data

The basic data type that is used to build more complex data types. For example, characters, integers, and real numbers. *See also* composite data.

primitive variable

A variable that contains primitive data. *See also* reference variable.

processor

The hardware that actually executes a program. Also called the central processing unit or CPU.

protocol

A set of rules that define how something works. For example, HTTP is a high level protocol that describes how a browser and a web server communicate using lower level protocols that handle other details.

prototype

In JavaScript, a built-in variable with functions and properties that can be used by each child object.

pseudo-class

In CSS, a way to define a special state of an element. For example, the pseudo-class “:hover” defines an element’s hover state.

reference variable

A variable that does not contain primitive data, but refers to a location in the system’s memory where the data is stored. Typically used for arrays, strings, and other composite data. *See also* primitive variable.

run

See execute

run time

(1) The period of time during which a program runs. (2) Software that is used to help programs run.

SASS variable

A variable defined using SASS (Syntactically Awesome Stylesheet). SASS is an extension to CSS and adds features, including variables, that make it easier to develop style sheets.

scope

The parts of a program in which the specific name of a variable, function, or class has meaning. For example, a global variable’s scope is the entire program, while a local variable’s scope is a single function.

screen reader

A program that finds text on the screen and reads it aloud, to assist users with limited vision.

script

A program written in a scripting language, usually intended for an interpreter rather than a compiler. Originally, scripts were short programs that performed very specific tasks, but over time scripting languages have been used for many other purposes.

semantic

The part of code that is focused on the underlying meaning of text, rather than the rules it follows (the syntax). Most HTML tags focus on the meaning and role of the data, not the appearance. For example, <h1> marks a heading and marks emphasized text, but neither describes how the text should be displayed.

server

A hardware or software system that provides services to other systems or clients. Software servers include database servers, mail

servers, and web servers. A hardware server can run more than one software server.

software

A set of instructions or data that tells a computer what to do, including the operating system, libraries, server software, and user applications. *See also* hardware.

source code

The set of instructions that is read and written by users. Source code can also be intended for an interpreter or a compiler.

state

The way a GUI element (a button or a text field) looks, which may change over time. For example, a button might be in its normal state most of the time, but may switch to its hover state when a cursor or pointer moves over it.

string

A sequence of characters that are stored together. This includes letters, numbers, and punctuation. In most languages, literal strings are written within quotation marks.

style definition

In CSS, the definition of a specific style for a category of text. For example, the style definition for a list might include what type of bullets to use and how much to indent.

subset

A group of items taken from another set.

syntax

The part of code that is focused on the rules followed by text rather than its underlying meaning (the semantics). For example, the syntax for emphasized text requires an tag, the text, and an .

tag

In HTML, the text marking the start and end of an element, usually using angle brackets. For example, and are tags used for emphasizing a piece of text.

template literal

A way to write a string that can span multiple lines and insert the values of other variables.

tuple

A short list of items or values; a 2-tuple has two items, and an n-tuple has n items.

URL (Uniform Resource Locator)

A consistent way to refer to the location of some resource and how to access it via the internet.

variable

The name associated with a value stored in the system’s memory. In computing, a variable can have different values at different times.

version control system

A system that keeps track of files so that users can easily collaborate and access different versions of the same file. Often, but not necessarily, used in software development.

view

In the Angular framework, a set of screen elements that control what the users can see.

virus

A type of malware that inserts its code into other programs, creating more copies of itself.

web page

A document that can be accessed over the internet. It is displayed in a web browser.

website

A set of related resources, such as web pages, images, sounds, and videos that are stored and accessed together over the internet using a web browser.

Index

Page numbers in **bold** refer to main entries.

32-bit processors 96, 97
64-bit processors 96, 97
@media screen 315
\$(document).ready function
 (home.js) 293, 321
.NET Core + ASP.NET Core 217
!important declaration **239**

A

About Menu 303
accessibility (websites) **214–215**
Actor class (Pygame Zero) 179, **183**,
 184, 186, 197–198
Ada 346, 347
Adobe Illustrator 286
agile model 21
Agile Software Development **181**
AI *see* artificial intelligence
AJAX (Asynchronous JavaScript)
 265
alert box 272, 273
algorithms 53, 270
Alice 346
analysis (software development) 20
anchor property (Python) 183
and (logical operator) 270
and block (Scratch) 45
Angular **285**
animate() command (jQuery) 294
animate() function (Pygame Zero)
 197, **198**, 202
animating the web page **288–303**
 adding JavaScript files 290–292
 adding social media 301–302
 exploring fonts and icons
 300–301
 getting started 289–290
 hacks and tweaks 300–303
 managing promotional messages
 296–299
 managing the scroll to top button
 292–295
 page template 303
 program design 288
 project requirements 289
 what the program does 288

animation (CSS) 239
answer blocks (Scratch) 47
APL 345
append() (Python) 113
app.js 290, 291, 308
application layer protocol 207
applications software 17
architecture, computer 96, 97
Arduino 23
area charts 287
arithmetic operators 43, **102**,
 104, 199
arrays 268, 298
 and loops 122, 275, 276
arrow keys 80, 84, 178, 185, 186
artificial intelligence (AI) 345
artists 18
ask blocks (Scratch) 47, 55, 59,
 62, 63
ASP.net and web development
 217
assembly languages 22, 344
Asteroid dodge (Scratch) **80–91**
 code the rocketship 83–86
 create the asteroids 86–89
 hacks and tweaks 90–91
 how the game works 80
 prepare the launch 82–83
 program design 81
Atom (code editor) 208

B

backdrop (Scratch)
 changing 54, 64, 78, 82
 color cycling 90
 designing 66–68
 Sounds tab 79
 switching 85
Backdrop Library (Scratch) 78
 background color
 canvas widget 147
 CSS styling 235, 238, 241, 247,
 307, 339
 promo bar 248
 specifying 312
 subscribe section 259
background images 251, 337
background music (Scratch) 79
background (Pygame Zero)
 checkerboard 199–201
 drawing 181
Backus, John 347
banner 221, 224
 logo 251
 styling 251
bar charts 134, 287
BASIC 346, 347
Basic Input/Output System (BIOS) 17
binary digits 23, 344
BIOS *see* Basic Input/Output
 System
bitmap mode 33, 57, 67, 68, 82
blind people 214
block-based languages 346
Blocks Palette (Scratch) 30, 35, 55
 Add Extension section 35, 37, 39,
 58–59
 Events section 40
 Looks section 38
 Variables section 42, 43, 58, 69,
 70, 72
blocks (Scratch)
 color-coded **34–35**
 defining your own 51
 dragging and dropping 74
Blockly 346
body, styling 307
body tag 247, 310
Boolean expressions **44**, 104, 105
 combining **45**
 using **45**
Boolean operators 104, 270–271
Boolean values 270
Boolean variables (JavaScript)
 266
Bootstrap **285**, 341
 column definition 305, 335, 336
 container 311
 creating a carousel 327–331
 grid system 324
 order of tags 310
 responsive layouts 304, 309
 SASS 341
border styling 239, 245
Boyce, Raymond 347
Brackets (code editor) 208
braille 214, 215
brain teaser (Scratch) **64–79**
 adding more sprites 74–77
 adding a new sprite 71–73
 adding the rules 78–79
 getting started 66–70
 hacks and tweaks 78–79
 program design 65
 the brain teaser 64

branching
 JavaScript **271**
 Python **105**, 119
break command (Python) 110
breakpoints 281
break statement 271
broadcast blocks (Scratch) 48, 85
broadcasts **48–49**
 uses of 49
browsers 217
 CSS features **255**
 Developer Tools 281, 305
 and screen size 240
 transitions 249–250
Brush tool (Scratch) 67, 88
Budget manager **158–175**
 adding a budget 161–164
 converting the code into a class
 169–172
 hacks and tweaks 174–175
 program design 159
 setting up 160
 tracking expenditures 164–168
 tracking multiple budgets 172–173
 what the program does 158–159
building (software development)
 20
build a web page **216–233**
 adding a contact section 230–231
 adding the copyright notice 233
 adding the footer 232
 adding more feature boxes
 226–229
 adding the subscribe section
 232
 feature box control 224–226
 getting started 218–220
 how it works 216
 HTML stage 216
 installing an IDE 217–218
 program requirements 217
 scrolling to the top 230
 structure the home page 220–224
business
 applications 95
 programming languages 346
buttons
 button widgets 147, 148
 call-to-action 220, 224, 258, 329,
 330
 carousel next/previous 329, 331
 clear 155
 creating in Scratch 56, 60–61

buttons *continued*

- hamburger menu 310, 311
- Open project 135, 153
- PayPal “Buy Now” 342
- radio 212
- with rollovers 243, 253, 254
- scroll 257–258
- scroll to top 257, 292–295
- Shop Now 253, 254
- subscribe 260
- templates 333

C

- C 94, 344, 347
- C# 303, 344
- C# MVC 303, 343
- C++ 344, 347
- calculations 43, 102
- calling (functions) 112, 113, 310
 - scheduling 194
- call-to-action button 220, 224, 258, 329, 330
- cameras 17
- canvas widget, Tk 147, **150**
- carousels 327–331
- cars 16
- Cascading Style Sheets *see* CSS
- case sensitivity 99
- casting 103
- catch block 281
- CDN (content delivery network) 290
- central processing unit (CPU) 344
- centring 247, 313
- Chamberlain, Donald 347
- characters 103
 - disallowed *see* entities
- Chart.js **287**
- charts
 - area 287
 - doughnut 287
 - flowcharts 53, 144
 - Gantt charts 134, 135, 149, 151, 152
 - JavaScript 287
 - line 287
 - Python 146–151
- cheat codes 90
- checkboxes 212
- Check Module (IDLE) **131**
- child elements 237, 238
- child objects 282, 283
- child selector 237
- child tags 210, 211, 221
- choice() function (Python) 122
- Chrome *see* Google Chrome
- class attribute 225
- classes
 - HTML tags 211, 246
 - JavaScript 282, **283**
 - Python **156–157**, 158, 159, 168–172, 283
- class inheritance 182
- class selector (CSS) 236
- Clean up Blocks (Scratch) 76
- clear property 246
- click() function (jQuery) 294
- client/server model 206, 207
- client-side scripting **209**, 264, 289
- clock events 179
- clock object **194**
- COBOL 346, 347
- Code Area (Scratch) 30, 35
- code editors **208–209**, 265
- CodePen 209
- coders
 - becoming a coder **16**
 - in the real world **18–19**
- Codeshare 265
- code sharing websites 265
- collision detection 47
- color
 - background 235, 238, 241, 247, 307, 312
 - font 235, 247
 - Pygame Zero 190
 - rows 339
 - styling 238
 - text 262
 - website style sheet 244
- color-matching games 179
- columns
 - adding responsiveness 339–340
 - Bootstrap 305, 335, 336
 - column headers 149
 - column numbers 182
 - column values 137, 138, 160
 - feature box styling 253
- combine id and class selector 237
- comma-separated values *see* CSV
- communication protocols 206, 207
- community sharing 265
- company logo 220, 223, 224, 286, 305, 311, 313

- comparison operators 270
- compilers 23
 - programming languages 344
- complex logic 64
- concatenation 103, 130
- config method 154
- confirmation box 273
- console log 273, 281
- constructor method 282, 283
- Contact Us 220, 230–231, 256, 300, 303, 312
 - styling 258
- container tags 333
- content attribute 240
- Content Delivery Network *see* CDN
- content organization (websites) 214
- continue command (Python) 110, 111
- Control blocks (Scratch) 40
- coordinates 36, 69, 70, 72
 - canvas widgets 150
 - grid and screen **182**
- copyright 220, 233, 305, 331–332, 334
 - styling 263
- corners, styling 308
- costumes (sprites) 30–31, 33, 38, 57, 61, 78, 85, 91
- cross-platform run time engines 289
- CSS 23, **234–239**
 - animating the web page 288, 289, 301, 303
 - building a web page 216, 218, 222, 233
 - debugging 281
 - and graphics design 286, 287
 - meta links 303
 - new features on browsers **255**
 - responsive layouts **240–241**, 304, 306–308, 312–316, 318–321, 325–326, 330, 332, 337–340, 341
 - selectors 236–237
 - styling 238–239
 - styling the web page 242–246, 250
- CSV files 135
 - creating and reading 136–143
 - Python library 136, 138, 139
- csv.reader object 138, 139
- current year Sensing block (Scratch) 45

D

- D3.js (Data-Driven Documents) **287**
- Dabblet 265
- data
 - adding to web page 221
 - programming languages for 345, 347
 - reading from files 135, 138
- databases, programming languages 344, 345
- data centers 16
- Data-Driven Documents
 - see* D3.js
- data types
 - converting 139
 - nonprimitive **268**
 - primitive **266–267**
- datetime module (Python) 116
- deaf people 214
- Deals hyperlink 312
- debugging
 - checklist 133
 - code editors **208–209**
 - debuggers 22, 106, **133**, 281
 - JavaScript 273, **280–281**
 - Python 94, 106, **130–133**
- decisions (Scratch) **44–45**
- declarative programming 24
- default code block 271
- delay, introducing **41**
- Department of Defense (US) 346, 347
- descendant selector 237
- design
 - program 53
 - software development 20
- desktop computers 16, 215, 240, 241, 305
- Developer tools **222**, 280, **281**, 305, 313
- development environment 217, 305
- dictionaries, Python 138, 145, 158, 159, **160**, 174
- DNS (Domain Name Systems) protocol 207
- Document Object Model (DOM) 265
- DOM *see* Document Object Model
- Domain Name Systems *see* DNS
- doughnut charts 287

do while loops (JavaScript) 275
 drag and drop interfaces 25
 Scratch 28, 30, 35, 74, 76, 84
 draw_actors() function
 (Pygame Zero) 184, 188, 191
 draw_background() function
 (Pygame Zero) 199, 200
 draw_chart function (Python)
 149, 151
 draw() function (Pygame Zero)
 186
 draw_game_over() function
 (Pygame Zero) 196, 197
 draw_handler function
 (Pygame Zero) 179
 drawing (Scratch) 67
 draw interface 179
 draw_scenery() function
 (Pygame Zero) 183, 202
 drop-down lists 212
 vertical 311
 dungeon crawl games 178, 179
 duplication
 code 62
 sprites 60

E

ECMA Script 264
 Edge 222
 editing, code **208–209**
 editor window (IDLE) 98, 99
 education, programming languages
 346
 Eich, Brendan 347
 elastic cloud computing 215
 element selector (CSS) 236
 elements (HTML tags) 210, 211,
 234, 235
 elif *see* else-if
 else branch (Python) 105
 else-if branch
 JavaScript 271
 Python 105
 email
 hyperlink 256
 id 259
 embeddable scripts 95
 engineering, programming
 languages 346
 entities **233**
 equality 104
 equals signs (Python) 104

error messages
 code editors 209
 Python 99, 132–133, 162, 163–164
 errors
 in JavaScript 280
 in Python 130–131, 162, 163–164
 see also debugging
 escaping loops (Python) 109,
 110–111
 EvalError (JavaScript) 280
 event-driven programming 24, 40
 event-handler function (Python)
 185, 186
 event handlers (JavaScript) 294
 event loop programs 178, 179
 Events blocks (Scratch) 40, 46
 exceptions (Python) **162**
 expenditures, tracking 159–175
 explosions, painting 88
 Extension blocks (Scratch) 35, 37,
 39, 58–59
 external files 264
 CSS 234
 external hyperlinks 213
 external information 46

F

factories, automation 17
 false value 270–271
 favicon **221**
 definition 221, 309
 images 300
 feature boxes 220, 224–229
 styling 253–257
 feature images 305, 313–315
 banner 314
 fields (objects) 156, 157
 File Explorer 243
 filename labels 154
 file objects 107
 files
 input from 107
 output to 107
 File Transfer Protocol *see* FTP
 Fill tool (Scratch) 57, 68, 82, 88
 financial planning 158–175
 Fintech (financial technology) 95
 firmware 17
 floats (floating point numbers)
 102, 139, 266
 FLOSS (free/libre and open
 source software) 94

flowcharts 53, 144
 folders
 creating 160, 180, 218, 243, 289,
 306, 308
 locating **243**
 fonts
 canvas widget 150
 changing 57, 300, 301
 color 247, 307
 company logo 313
 default 307
 Google Fonts 244, 300, 341
 list of website 244
 prominent 316
 setting/defining 235, 244, 259,
 319, 325
 size 190, 238, 315
 footer section 220, 232, 247, 249,
 331, 333
 styling 261–262
 footers, table 167–168
 forever blocks (Scratch) 41, 84,
 86
 forever loops 79, 86
 for loops
 JavaScript 268, 274
 Python 108, 110, 122, 145
 for in loops (JavaScript) 275
 Format menu (Python) 169
 format strings (Python) 159, **166**
 forms, HTML **212–213**
 Fortran 346, 347
 frame container widget 147
 frameworks (JavaScript) 284–285,
 286, 289
 FTP (File Transfer Protocol) 207
 functions 51
 functions (JavaScript) 269,
 278–279, **282**, 283, 291
 creating 296, 308
 declaring 278
 function statement vs function
 expression 278
 nested 279
 self-executing 279, 291
 functions (Python) **112–115**, 161,
 164–165
 built-in 113
 calling 112, 113, 117, 159
 creating 114–115
 defining 112, 114–115
 importing 117
 methods 169, 170

functions *continued*
 methods for scheduling calls 194
 naming **114**, 117
 functions (Scratch) **50–51**

G

Game Over message 89, 190, 195,
 196
 Game_over variable 189
 games 23
 Asteroid dodge (Scratch) 80–91
 consoles 17
 development 80, 95
 gaming apps 134, 137
 Knight's quest (Python) 178–203
 Gantt charts 134, 135, 149,
 151, 152
 Gap Time 86
 general sibling selector 237
 getters 283
 Gimp 286
 Github Gist 265
 global positioning system *see* GPS
 global variables
 JavaScript 290, 291, 308
 Python 115, 160, 163, 189, 195
 Go 344
 GoDirect Force & Acceleration 59
 Google Chrome 217, 222, 281,
 305, 313
 Google Fonts 244, 300, 341
 Google Maps 258
 Google Material Icons 341
 Google Translate API 52
 Gosling, James 347
 GPS 16
 graphics
 graphical modules (Python) 134,
 146–151
 graphic user interfaces (GUI)
 147, **286–287**
 programming languages for
 345
 graphs and charts 287
 grid
 coordinates **182**, 185–187
 HTML Canvas 287
 moving on the **187**, 192
 width and height 203
 GUI *see* graphic user interfaces
 guidelines, compliance with
 website 215

H

hamburger menu button 310, 311
 hard-coding 100
 hardware 17
 machine and assembly languages 344
 headers
 carousels 327
 defining 245, 249
 styling 307
 tables 167–168
 hex code 150
 hidden fields (web pages) 213
 high-level programming languages 22
 hobbyists 18
 home automation 23
 Homebrew package manager 176–177
 Home() function (home.js) 321
 Home hyperlink 312
 Homeindex() function (home.js) 292, 293
 home.js 290, 292, 317, 321
 Home Menu 300
 home page
 copyright section 331–332
 design 305
 feature image 313–315
 navigation bar 309–313
 primary message 316–317
 quote 317–323
 responsive 304
 slideshow 327–331
 Hopper, Grace 347
 horizontal borders 239
 horizontal layers 220, 247, 305
 horizontal margins 248, 316
 horizontal menu lists 249, 251
 horizontal rule 225, 255, 336
 hosting, web **215**
 household appliances 16
 hover state 249, 252, 254, 258, 260, 262, 313, 326, 330, 332, 338, 339
 HTML (Hypertext Markup Language) 23, 206, 207, 209, 345
 animating the web page 288, 289, 290–292, 296, 300–303
 building a better website **214–215**
 building a web page **216–233**
 color codes 238
 common entities **233**

HTML *continued*
 and CSS 234–239
 debugging 281
 document structure 211
 exploring basic **210–211**
 forms and hyperlinks **212–213**
 and graphics design 286, 287
 and JavaScript 264, 265, 272–273, 286, 288
 responsive layouts 240, 304, 305, 309–314, 316–318, 324–325, 327–329, 331, 333–337
 styling the web page 242, 246, 247–253, 254
 tags and attributes **210–211**, 214, 215, 234, 239
 template file 303
 templates 333–34
 HTML Canvas **287**
 HTTP (Hypertext Transfer Protocol) 206, **207**
 hyperlinks **212–213**
 adding to navigation bar 312
 call-to-action 329, 330
 copyright section 332
 email 256
 footer 262
 home page 220, 304, 309
 list 223, 232
 styling 249–250, 252, 254, 256, 330, 339
 Top Menu 250

I

IBM 347
 Ichbiah, Jean 347
 icons 300, 341
 IDEs (Integrated Development Environment) **23**, 132, 208, **209**, 289, 305
 installing 217–218
 IDLE (Integrated Development and Learning Environment) 95, 96, 97, 113, 120, 121, 136, 160, 180, 182
 Check Module **131**
 colors in the code **98**, 132
 creating a CSV file in 136
 debugging 130, 131, 133
 editor window 98
 shell window 98
 using **98–99**
 id selector (CSS) 237
 if branch (Python) 105, 118
 if-then block (Scratch) 44, 59, 62, 63, 72, 73, 74, 75, 77–79, 84–87
 if-then branch (JavaScript) 271
 if-then-else block (Scratch) 44, 73, 75, 77
 if-then-else branch (JavaScript) 271
 images
 adding to web page 221, 228, 229, 305
 background 227, 251
 centring 314
 feature 255, 257, 305, 313–315
 folder 180, 217, 218, 242, 306
 middle 256, 257
 styling 257
 image tile grids 178, 179
 imperative programming 24
 importing (Python) 117
 indentation
 errors 130
 Python 99, 108, 109, 130, 169
 tags 211
 indexes (Python) 122
 index file 218, 219, 306
 indexing (search engines) 214
 index strings 179
 infinite loops (Python) 109
 infographics 287
 information processing 46
 initializers (Python) 169
 inline CSS 234
 input
 JavaScript **272–273**
 Python **106–107**
 Scratch 34, **46–47**
 types of 46
 input blocks (Scratch) 34
 input events 179
 input fields 212
 styling 259
 input() function (Python) 106, 113
 input validation (web pages) 213
 instance variables 170
 instructions, computer programs as 17
 integers **102**, 139, 140, 141, 266
 Integrated Development Environment *see* IDEs
 Integrated Development and Learning Environment *see* IDLE

integrity attribute (Bootstrap) 309
 interactive behaviors 216, 272, 273, 288
 internet 206
 Internet Explorer 217, 222
 interpreters 23
 int function (Python) 140
 IP (Internet Protocol) 207
 IP address 207
 IP routing **206**
 issubset set method **145**
 iterations 108–111, 118, 181, 182, 197, 276–277
 iterative model 21

J

Java 25, **264**, 344, 347
 JavaScript 23, 24, 25, **264–287**, 347
 animating the web page **288–303**
 building a web page 216, 218, 221, 233
 debugging **280–281**
 features of **265**
 functions in **278–279**, 310
 graphic user interfaces **286–287**
 input and output **272–273**
 interactive functionality 289
 libraries and frameworks **284–285**, 289
 logic and branching **270–271**
 loops in **274–277**
 object-oriented **282–283**
 order of tags 310
 responsive layouts 240, 304, 308, 321–323
 using online 264
 variables and data types **266–269**
 what it is **264–265**
 JavaScript Engine 264, 277, 284, 291, 293, 308, 310, 321, 328
 JQuery 265, **284**, 289, 294, 298, 305
 adding 290
 order of tags 310
 responsive layouts 304, 309
 JSFiddle 265
 Json data format 269

K

Kemeny, John 347
 keyboard control 46, 186, 214
 keys 187, **188**
 dictionary 138, 140, 160
 keywords 210
 class 156, 283
 Pygame Zero keyword arguments 198
 Knight's quest **178–203**
 adding the keys 187–188
 adding messages 196–197
 animating the actors 197–198
 creating the guard actors 191
 creating the player 184
 hacks and tweaks 202–203
 how to play 178–179
 moving the guards 192–194
 moving the player 185–187
 setting up 180–184
 tracking the result 194–197
 Kurtz, Thomas 347
 Kyma 346

L

labels 212
 label widget 154
 labyrinthine environments 178
 languages
 foreign 52–63
 programming *see* programming languages
 laptops 240
 layers, horizontal 247, 305
 layout styles, different 241
 left column elements 225
 LEGO BOOST 59
 LEGO Education WeDo 2.0 59
 LEGO Mindstorms EV3 59
 len() function (Python) **103**, 113
 Lerdorf, Rasmus 347
 less than 104
 libraries
 Google fonts and icons 341
 graphics 287
 JavaScript 265, 284–285, 286, 287
 Python 94, 95, 114, 116–117, 136, 138, 139, 146, 159, 173
 Scratch 39, 54
 Lifelong Kindergarten group (MIT) 29
 lightweight editors **208**

line charts 287
 link layer protocol 207
 Linux 29
 LISP 345
 lists
 of lists 111
 loops with 108
 Python **103**, 111, 113, 118, 119, 121, 122, 129
 Scratch **43**
 sets compared with 140
 shuffling 121
 splitting **122**
 tuples compared with 137
 within lists 111
 list widget 147
 local variables 115, 269
 location (Contact Us section) 258
 logic
 errors 131
 JavaScript **270**
 Python **104–105**, 140
 Scratch **44–45**
 task ordering 144–145
 logical operators 104, 270–271
 logic programming 345
 logo
 company 220, 223, 224, 286, 305, 311, 313
 styling banner 251
 styling Top Menu 250, 313
 loops
 continuous 135, 144, 145
 do while loops 275
 escaping 109, 110–111, 277
 infinite **109**
 JavaScript **274–277**
 with a list 108
 loop conditions 108
 for loops 108, 110, 122, 145, 268, 274
 loop variables 108, 111
 nested **110–111**, 179, 276
 Pygame Zero game loop **179**
 Python **108–111**, 118, 119, 122, 125
 Scratch **41**, 45, 65, 73, 75, 79, 81
 while loops 108, 109, 110, 125, 274
 Lovelace, Ada 347
 low-level programming languages 22

M

Mac computers
 locating folders 243
 Pygame Zero 176–177
 Python on 97
 Scratch on 29
 machine code 23, 264, 347
 machine languages 344
 main loop function 147
 Makey Makey blocks (Scratch) 59
 managers 18
 map() function (Python) 140
 Maple 345
 maps
 Contact Us section 258
 virtual worlds 287
 margins **245**
 setting 247, 307, 311, 319
 Mathematica 345
 mathematics, symbolic 345
 MathJS **285**
 MATLAB 345
 Max 346
 media queries 241
 messages
 displaying (Scratch) 38
 naming 49
 primary (websites) 309, 316–317
 sending (Scratch) **48–49**
 metadata 221, 240
 meta links 303
 methods
 calling (Python) 113
 JavaScript 282, 293, 297, 298
 Python 113, 157, 169, 170, 174
 micro:bit 59
 microprocessors 23
 Microsoft Basic 347
 Microsoft Visual Studio
 Community 2019 *see* Visual Studio Community 2019
 Microsoft Windows 17
 locating folders 243
 Pygame Zero 176–177
 Python on 96
 Scratch on 29
 military, programming languages
 for 346
 modal windows 272, 273
 modules
 adding 309–310
 built-in 116
 modules *continued*
 importing and using 117
 Python **116–117**
 modulo (remainder) operator (Python) **199**
 Moment.js **285**
 money, Budget manager project **158–175**
 Monty Python's Flying Circus 96
 Motion blocks (Scratch) 36–37
 mouse 214
 mouse-over state 230, 249, 252, 254, 258, 260, 262
 move_guard() function (Pygame Zero) 193, 194, 197
 movement
 illusion of 81, 87
 logic 185
 move_player() function (Pygame Zero) 186, 188, 189, 193, 195, 198, 202
 multimedia 206
 multiple classes selector 237
 multiview function 208
 Music blocks (Scratch) **39**, 59
 music, programming languages 346

N

named tuples (Python) 135, 142–143
 name errors 131
 NASA, Mission Control Center 95
 navigation bar 305
 creating 309–313
 templates 333
 nested functions (JavaScript) 279
 nested loops
 JavaScript 276
 Python 110–111, 179
 NetBeans 209
 network protocols 207
 new projects (Scratch) 54, 66
 next sibling selector 237
 Node.js 264, **284**, 285, 289
 Node Package Manager
 see NPM
 noise reduction 345
 nonprimitive data types **268**
 normal state 254
 not (logical operator) 271
 not block (Scratch) 45
 not equal 104

Notepad 208, 218

NPM **284**

numbers

computation 345

in JavaScript **266**

list of 129

in Python **102**, 129, 139

random 43, **121**, **201**

in Scratch **43**

O

Objective-C 344

object-oriented programming

25, 156–157, 264, 347

JavaScript **282–283**

objects

JavaScript 269, 282–283, 297

Python **156–157**, 188

office applications, programming
languages 344

office workers 18

on document ready() function

(home.js) 292, 321, 323

on_key_down() function (Python)
186

opcode 23

Opera 222

operands 23

operating systems 17

installing Python 96–97

programming languages 344

updates 177

Operator blocks (Scratch) 42–43,
44, 64

or (logical operator) 270

or block (Scratch) 45

outline color (Pygame Zero) 190

output

JavaScript **272–273**

Python **106–107**

Scratch **35**

output blocks (Scratch) 35

P

package managers 176–177

packets **206–207**

packet sequence 207

packet switched networks 206

padding 235, 245, **245**, 247, 248,
249, 252, 307, 311

page elements, styling 244–253

Paint Editor (Scratch) 33, 57, 61, 64,
66–68, 78, 88

pandas 95

parent elements 237, 238

parent function 279

parent objects 282

parent tags 210, 211, 239

parent widgets 147

Pascal 346

PayPal 342

pen blocks (Scratch) 37, 59

Perl 344

pet shop website project

animating the web page

288–303

Build a web page **216–233**

styling the web page **242–263**

PHP 209, 344, 347

pip package managers 176–177

pixels 150, 182, 215, 238

placeholder text 261

place markers 327

planning (software development)

20

plugins 209

pos property (Python) 193

PostScript 345

prerequisites

as sets of numbers 140

tasks 134, 141, 144, 145

preview windows 208

primary messages 305, 316–317

primitive data types **266–267**, 268,
269

print() function (Python) 106, 113

printing (code editing) 208

probability **201**

problem-solving 65

procedural programming 25

processing blocks (Scratch) 34–35

program flow 34–35, 50

managing 40–41

programming **16–17**

programming languages 16, **22–25**,
344–347

choosing **25**

for data **345**

early **346**

families of **344**

popular **347**

for specific purposes **345**

types of **24–25**

visual **346**

programming languages *continued*

website construction 216

see also JavaScript; Python;

Scratch

programs 17

Project planner **134–155**

creating and reading the CSV file

136–143

drawing the chart 146–151

hacks and tweaks 152–155

how it works 134

ordering the tasks 144–146

program design 135

Prolog 345

promo bar 247–248, 287, 288, 296,
299

promotional messages 220, 221,

296–299

cycling through 298–299

initializing 297–298

styling 247–248

prompts 272

protocols **207**

prototype-based languages 283

prototypes (JavaScript) **282**, 283

push() method (JavaScript) 268

puzzles 64–79

PyBrain 95

Pygame **117**

Pygame Zero 95, 117, **176–177**

Actor class 178, **183**

animations **198**

drawing text with **190**

game loop **179**

initializing 180

library 117

updates **177**

PySoy 95

Python 22, 24, 25, **92–203**, 347

applications **95**

Budget manager project **158–175**

common errors 99

data in **102–103**

debugging **130–133**

dictionaries **160**, 174

exceptions **162**

features 94–95

format strings **166**

functions **112–115**, 159, 161,

164–165

how it works 95

input and output **106–107**

installing **96–97**

Python *continued*

Knight's quest project **178–203**

libraries 94, 95, 114, **116–117**, 136,
138, 139, 146, 159, 173

logical operators and branching
104–105

loops in **108–109**

on a Mac 97

objects and classes **156–157**

Project planner project **134–155**

Pygame Zero **176–177**

sets **140**

Team allocator project **118–129**

tuples **137**

using IDLE **98–99**

variables **100–101**, 115, 121, 125,
160, 163, 169, 170, 189, 195

versions 96

on Windows 96

Python Django 303, 343

Python Standard Library 116

Q

quotation marks 103, 320–321

quotes 305, 317–323

animating 322–323

applying properties to 321

initializing 322

R

R 345

radio buttons 212

randint() function (Python) 201

random allocation 118, 119

random blocks (Scratch) 87

random module (Python) 116, 118,
119, 120, 121, 200

random numbers 43, **121**, **201**

RangeError (JavaScript) 280

range function (Python) 108

Raspberry Pi 23

ReactJS **284**

readability, code 94

readable content (websites) 214

read_tasks() function (Python)
141, 143

ReferenceError (JavaScript) 280

reference variables 268

repeat blocks (Scratch) 41, 45, 75

repeat until blocks (Scratch) 45

RequireJS **285**

researchers 19
 response variable 125
 responsive layouts (websites) 215, **240–241**, 304
 responsive website **304–343**
 adding the copyright 331–332
 adding a feature image 313–315
 adding last minute details 327–331
 adding a message 316–317
 adding the modules 309–310
 adding popular destinations 324–325
 adding a quote 317–323
 adding the title and favicon 309
 creating the navigation bar 309–313
 creating a new page 335–340
 creating a template 332–333
 getting started 305–308
 hacks and tweaks 341–343
 how it works 304
 program design 304
 project requirements 305
 result, tracking 195
 right column elements 226
 Ritchie, Dennis 347
 robotics 23, 179
 rollover effect 253
 root folder 218, 219
 root window widget 147
 rounded corners 308
 round() function (Python) 113, 186
 routers 206–207
 rows
 adding responsiveness 338
 changing colors 339
 row numbers 182
 styling 338
 Ruby 344
 run time environment 284
 runtime errors 130

S

Safari 217, 222
 SASS variables 341
 Scalable Vector Graphics *see* SVG
 scenery (Python) 182–183
 Scheme 345
 science, programming languages for 346

scientific computing 95
 scientists 19
 scope, variables **269**
 Scratch 25, **26–91**, 346
 Asteroid dodge project **80–91**
 backdrops 55
 brain teaser project **64–79**
 colored blocks and scripts **34–35**
 features **28–29**
 getting Scratch 29
 hardware support 29
 input **46–47**
 interface **30–31**
 logic and decisions **44–45**
 loops **41**, 45, 65, 73, 75
 managing program flow **40–41**
 manipulating data **42–43**
 output using looks and sounds **38–39**
 output using movement **36–37**
 sending messages **48–49**
 sprites **32–33**
 travel translator project **52–63**
 using functions **50–51**
 variables **42**, 44, 58, 69, 70, 72, 74–77, 79, 83
 versions of **30**
 screen coordinates **182**, 183
 screen_coords() function (Python) 186
 screen.draw.text() function (Python) 190
 screen size 215, 240–241, 304, 305, 309, 315, 316, 319, 325, 326, 338, 339
 adding responsiveness 315, 316, 319, 325, 330, 338, 339–340
 centring contents 313, 314
 scripting, client- and server-side 209
 scripting languages 344
 scripts
 creating 35, 344
 JavaScript 264, 265
 Python 95
 Scratch 34, 35
 scripts file 308
 scroll button, styling 257–258
 scrolling 214
 scroll to top 230, 257, 288, 292–295
 search engines 214

selectors, CSS 235, **236–237**
 complex 237
 grouping 237
 self-executing functions (JavaScript) 279
 semantics 215
 sensor input 46
 servers 206–207
 dedicated 215
 server-side scripting **209**, 264, 289
 template options 303, 343
 set() constructor function (Python) 140
 sets (Python) 134, **140**
 issubset set method **145**
 setters 283
 setup_game() function (Python) 187, 189, 191, 195, 196
 shape-matching games 179
 shared hosting 215
 shell window (IDLE) 98, 99, 120, 161, 163
 Shop Menu 303
 Shop Now link 253, 254
 Shopping Cart 300
 shuffle() function (Python) 121
 shuffling 118, 119, 121
 side-scrolling games 80, 81
 simulations 65
 slideshows 327, 328, 331
 smartphones 179, 215, 240, 241, 305
 social media, adding to websites 301–302
 socket module (Python) 116
 software 16, 18
 developers **20–21**
 engineers 19
 software systems, programming languages 344
 Solution Explorer 243, 289, 313, 321, 335
 solution file 218, 219
 Solution Folder 243, 289
 sound, Scratch 39
 space applications 95
 spacers 246, 307
 spaces, vertical 224, 246
 spacing settings 235, **245**
 special effects (Scratch) 38
 specificity 239
 split method (Python) 140
 sport, team allocator project **118–129**

spreadsheet applications 135, 136
 spread syntax **268**
 Sprite library (Scratch) 54, 68, 71, 74, 76, 78, 82
 Sprite List (Scratch) 30, 54, 83
 sprites (Scratch) **32–33**
 adding buttons 56, 60–61
 adding new 54, 71–77, 82
 broadcasts 48–49
 changing appearance 38
 collision detection 47
 costumes 32, 33, 57, 61, 78, 85, 91
 creating 32, 54, 68
 deleting the cat 54, 66
 drawing with pen blocks 37
 duplicating 60
 moving using coordinates 36
 moving using directions 37
 naming/renaming 56, 68
 speech/thought bubbles 38
 SQL 345, 347
 Stage (Scratch) 30
 changing backdrop 55
 coordinates 36
 designing backdrop 66–68
 Stage Info (Scratch) 30
 statistical computing 345
 steering controls 84
 strings
 combining 103
 concatenating 267
 format **166**
 JavaScript 266, **267**
 Python 100, **103**, 138, 139, 140, 141, 166, 190, 198
 Scratch **42**
 Stroustrup, Bjarne 347
 style definitions, CSS 235, **238–239**, 250, 252, 255, 319, 325
 styles folder 243
 style sheets 242, 243, 306
 styling
 body elements 307
 carousel 330
 Contact Us section 258
 copyright 263, 332
 corners 308
 CSS **238–239**
 and ease of navigation 243
 email hyperlink 256
 encourages interaction 243

styling *continued*

- feature box 253–257
- footer 261–262
- headers 307
- hyperlinks 252, 256, 262, 330, 339
- image and subheading 326
- image text 314
- input field 259
- logo 313
- margin and padding **245**
- nonpicture elements 254
- page elements 245–253
- primary message 316
- quotation marks 320–321
- quote 318
- rows 338
- scroll button 257–258
- standardization 244
- styling the web page project **242–263**
 - Subscribe section 259
 - text 306–308, 314, 330
 - Tk canvas widget 150
- styling the web page **242–263**
 - feature box styling 253–257
 - program requirements 242
 - setting up 243–244
 - styling the page elements 245–253
 - styling the remaining elements 257–263
 - what the program does 242
- subelements 210
- subheadings 325–326
- Sublime Text (code editor) 208
- subscribe button 260
- subscription link 220, 232, 247
 - styling 259–260
- subsets 122, 145
- substrings 267
- SVG **286**
- Swift 344
- switch statement **271**
- symbolic mathematics 345
- synchronization 49
- syntax 94
 - errors 130, 131, 132
 - highlighting 208
 - spread **268**
- SyntaxError (JavaScript) 280
- SynthEdit 346
- system languages 344

T

- tables
 - footers 167
 - headers 167
- tablets 215, 240, 305
- tabs 209
- tags, HTML **210–211**, 234
 - attributes 211
 - container 333
 - indenting 211
 - order of **310**
- TCP (Transmission Control Protocol) 207
- Team allocator **118–129**
 - create a team 120–121
 - hacks and tweaks 127–129
 - how it works 118
 - pick new teams 125–126
 - pick teams 122–124
 - program design 119
- templates
 - creating new pages from 305, 333–334, 335
 - literals 267
 - master page 303, 304
 - renaming 335
 - web page **343**
- testing (software development) 20
- TeX 345
- text
 - adding to web page 221
 - drawing with Pygame Zero **190**
 - feature images 314
 - placeholder 261
 - positioning 319
 - quote 318
 - styling 253–254, 306–308, 314, 330
- text alternatives (website) 214
- text area (web pages) 212, 213
- text-based languages 94
- TextEdit 218
- text editors 208, 305
 - IDLE 95
- text fields (web pages) 212, 213
- text files 305
- text input, users 47
- Text tool (Scratch) 57, 61, 88
- Text to Speech blocks (Scratch) 59, 63
- throw statement 281
- ticker tape 215
- tiles 179, 199, 200
- time management, Project planner **134–155**

- time module (Python) 116
- TK Canvas widget 147, **150**
- Tk Frame widget 153
- Tk GUI **147**
- tkinter (Python) 116, 146, 148, 150
- TK module (Python) 146–151
- top-down 2D view 178
- top-down coding 114
- top-level window widget 147
- Top Menu 220, 222, 223, 303, 310, 311
 - styling 249, 250–251
- touching mouse-pointer block (Scratch) 47
- tournaments, sporting 128
- traceback error messages (Python) 130
- transitions **250**
 - transition instruction 249–250, 252, 254
- Translate blocks (Scratch) 56, 59, 60, 63
- translation apps, travel translator project **52–63**
- transport layer protocol 207
- travel translator (Scratch) **52–63**
 - adding a language 56–60
 - adding more languages 60–62
 - hacks and tweaks 63
 - how the app works 52
 - program design 53
 - setting the scene 54–56
- travel website project **304–343**
- trend analysis 345
- true value 270–271
- try block 281
- tuples (Python) 134, **137**, 138, 142–143
 - calling 143
- turtle module (Python) 116
- tween keyword arguments **198**
- Twitter 301–302
- two-dimensional games 178, 179
- TypeError (JavaScript) 280
- type errors 130
- TypeScript **285**
- UDP (User Datagram Protocol) 207
- underscores 114
- Uniform Resource Locator *see* URL
- Unix operating system 94

- update game state (Pygame Zero) 179
- update handler function (Pygame Zero) 179
- updates, Pygame Zero **177**
- upper() method (Python) 113
- URIError (JavaScript) 280
- URL (Uniform Resource Locator) 206
 - hyperlinks and 213
- User Datagram Protocol
 - see* UDP
- user input
 - JavaScript 265, **272**, 284
 - Python **106**
 - Scratch 40
- user interactions 46

V

- value
 - Boolean 270
 - comparing **270**
 - functions producing 115
 - sets 140
 - of variables 100, 101, 109, 268, 269
- van Rossum, Guido 94, 96, 347
- variables
 - arrays 268
 - and Boolean operators 104, 266
 - class assigned to 283
 - declaring **101**, 267, 269
 - global **115**, 160, 163, 189, 195, 269
 - initializing 267
 - instance 170
 - JavaScript **266–269**, 278, 279, 298, 308
 - local **115**, 269
 - naming 101, 188
 - nonprimitive data types 268
 - primitive data types 266–267
 - Python **100–101**, 115, 121, 125, 160, 163, 169, 170, 189, 195
 - SASS 341
 - scope of **269**
 - Scratch **42**, 44, 58, 69, 70, 72, 74–77, 79, 83
 - and sets 140
 - using **101**
 - values 100, 101, 109, 268, 269

VBA 344
vector images 286
vector mode 33, 57, 61
vertical alignment 321
vertical borders 239
vertical lists 311, 312, 326
vertical spaces 224, 225, 230, 231, 246, 307, 319, 329, 333
video sensing blocks (Scratch) 59
viewport 240
viewport meta definition 309
Virtual Private Server *see* VPS
virtual worlds, mapping 287
Visual Basic 347
visual programming languages 25, 348
Visual Studio Code 208
Visual Studio Community 2019 209, 217–220, 242, 243, 289, 305, 306, 333
voice, choice of (Scratch) 63
VPS (Virtual Private Server) hosting 215

W

wait blocks (Scratch) 41, 45
wait until blocks (Scratch) 45
warp speed slider 80, 89
washing machines 16
waterfall model 21
web apps 23
webbrowser module (Python) 116
web browsers 206, 207, 208, 209, 214, 234, 250, 255, 265
Web Content Accessibility Guidelines 215
web development 19, 95
web hosting 215
web pages 215
 animating the web page project **288–303**
 building **216–233**
 creating new 335–340
 hyperlinks and forms **212–213**
 responsive 240, 241
 styling the web page project **242–263**
 templates 303, 333–334, 335, **343**

web protocol 207
web server 206, 207, 208, 209
websites 23
 accessibility 214–215
 animating **288–303**
 basic HTML **210–211**
 building better **214–215**
 building a web page **216–233**
 connecting to **206–207**
 HTML forms and hyperlinks **212–213**
 icons 300
 interactive 288
 payments on 342–343
 responsive website project **304–343**
 social media 301–302
 styling **242–263**
WebStorm 209
welcome message 120
when I receive block (Scratch) 48, 73, 75, 77, 86, 88
while loops
 JavaScript 274
 Python 108, 109, 110, 125

widgets
 frame container 147
 label 154
 list 147
 parent 147
 root window 147
 Tk canvas widget 147, **150**
 Tk Frame 153
 Tk GUI **147**
 top-level window 147
 Twitter 301
Windows *see* Microsoft Windows
windows, resizing 152
Wolfram Language 24
work schedules 134
World Wide Web, working of **206–207**
wrap class 248

XYZ

x coordinates 182
XML 265
y coordinates 182
zoom function 209

Acknowledgments

DK would like to thank the following people for help in preparing this book: Anjali Sachar, Mridushmita Bose, and George Thomas for design assistance; Deepak Negi for picture research assistance; Nayan Keshan and Kanika Praharaj for code testing; Helen Peters for indexing; Jamie Ambrose for proofreading; and Harish Aggarwal (Senior DTP Designer), Surabhi Wadhwa-Gandhi (Jacket Designer), Priyanka Sharma (Jackets Editorial Coordinator), and Saloni Singh (Managing Jackets Editor).

Scratch is developed by the Lifelong Kindergarten Group at MIT Media Lab. See <http://scratch.mit.edu>

Python is copyright © 2001–2019 Python Software Foundation. All Rights Reserved.

Microsoft Visual Studio 2019

Blockly is a library from Google for building beginner-friendly block-based programming languages.