abacusAn early computing device that allowed users to perform simple calculations by moving beads along wires.

absolute pathnameA pathname that begins with the file system's root directory. *See also* **pathname**.

abstractionA simplified view of a task or data structure that ignores complex detail.

accessorsA method used to examine an attribute of an object without changing it.

algorithmsA finite sequence of instructions that, when applied to a problem, will solve it.

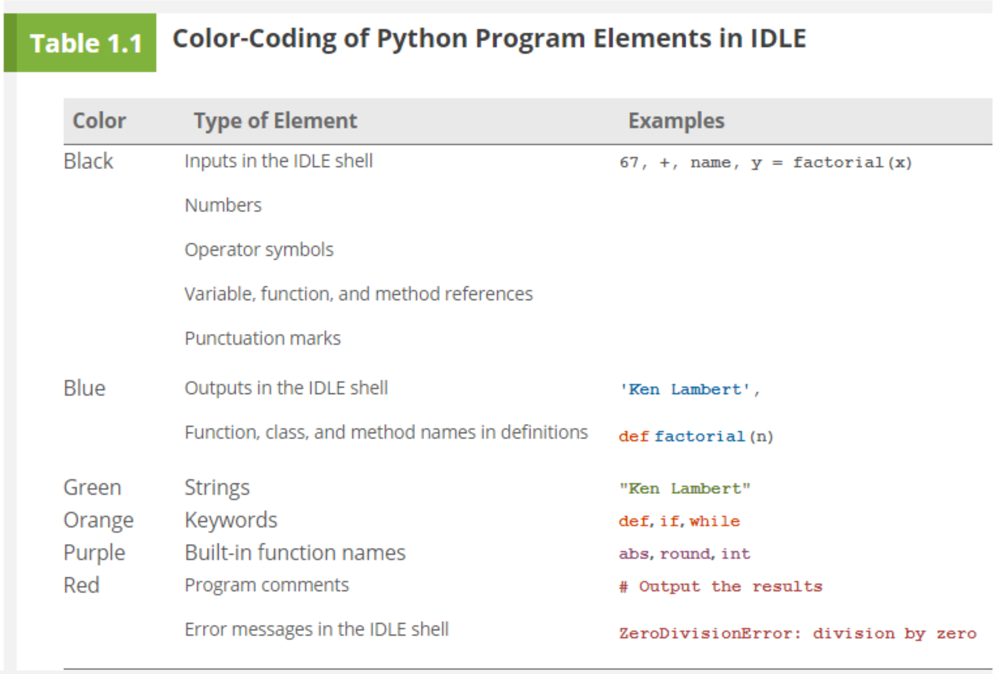
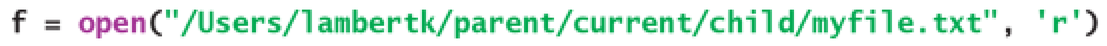


Table 1.1 Color-Coding of Python Program Elements in IDLE

Color	Type of Element	Examples
Black	Inputs in the IDLE shell	`67, +, name, y = factorial(x)`
	Numbers	
	Operator symbols	
	Variable, function, and method references	
	Punctuation marks	
Blue	Outputs in the IDLE shell	`'Ken Lambert',`
	Function, class, and method names in definitions	`def factorial(n)`
Green	Strings	`"Ken Lambert"`
Orange	Keywords	`def, if, while`
Purple	Built-in function names	`abs, round, int`
Red	Program comments	`# Output the results`
	Error messages in the IDLE shell	`ZeroDivisionError: division by zero`



```
f = open("/Users/lambertk/parent/current/child/myfile.txt", 'r')
```

AnalysisThe phase of the software life cycle in which the programmer describes what the program will do.

ancestorAny class that is either a parent of a class or lies on a path in the class hierarchy above that parent.

anonymous functionA function without a name, constructed in Python using **lambda**.

applications software Programs that allow human users to accomplish specialized tasks, such as word processing or database management. Also called applications or apps.

argumentA value or expression passed as data by the caller to a function or method.

arithmetic expressionA sequence of operands and operators that computes a value

artificial intelligenceA field of computer science whose goal is to build machines that can perform tasks that require human intelligence.

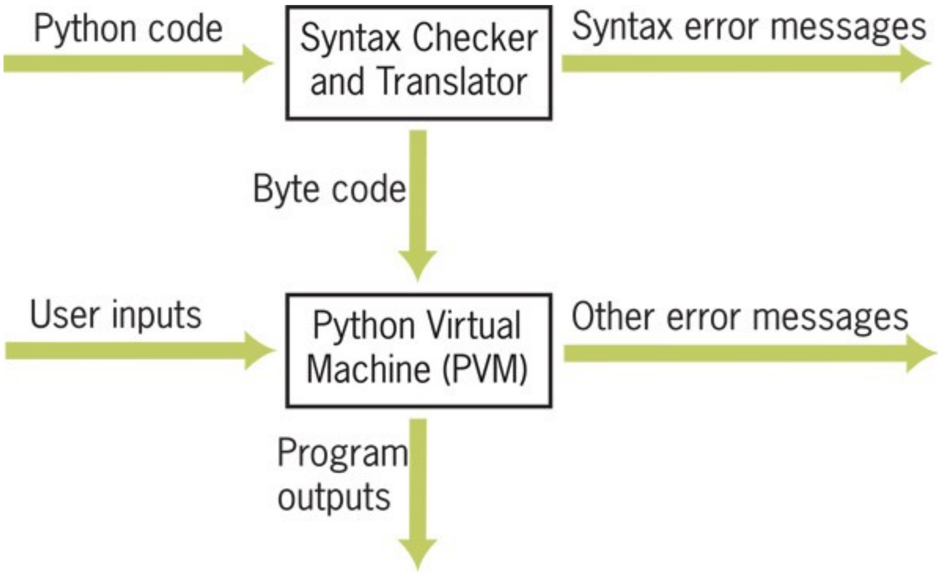
ASCII set The American Standard Code for Information Interchange ordering for a character set.

assemblerA program that translates an assembly language program to machine code.

assembly languagesA computer language that allows the programmer to express operations and memory addresses with mnemonic symbols.

accessor methodsA method used to examine an attribute of an object without changing it.

assignment statementA method of giving values to variables.



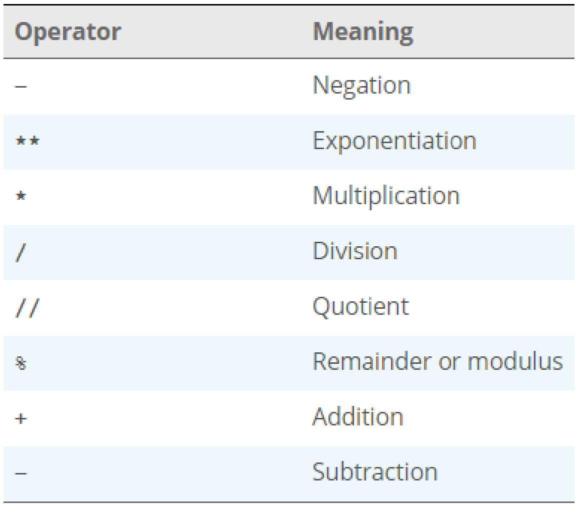
associationA pair of items consisting of a key and a value.

association lists*See* **dictionary**.

augmented assignment operations An assignment operation that performs a designated operation, such as addition, before storing the result in a variable.

base case The condition in a recursive algorithm that is tested to halt the recursive process.

batch processing The scheduling of



Operator	Meaning
–	Negation
**	Exponentiation
*	Multiplication
/	Division
//	Quotient
%	Remainder or modulus
+	Addition
–	Subtraction

multiple programs so that they run in sequence on the same computer.

big data The gathering and analysis of massive amounts of data.

binary digitsA digit, either 0 or 1, in the binary number system. Program instructions are stored in memory using a sequence of binary digits. *See also* **bit**.

bit-mapped display screenA type of display screen that supports the display of graphics and images.

bitsA binary digit

bodyThe code segment nested within a loop, selection statement, function definition, method definition, or class definition.

Boolean data typeA data type whose values are **True** and **False**

Boolean functionA function, also called a predicate, that returns the Boolean value true or false.

byte code The kind of object code generated by a Python compiler and interpreted by a Python virtual machine. Byte code is platform independent.

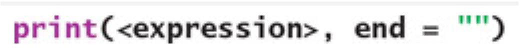
call stack The area of computer memory reserved for managing data associated with function and method calls.

canvasA rectangular area of a window within which geometric shapes, images, and text can be drawn

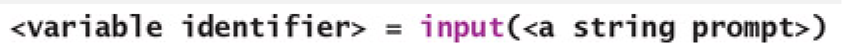
central processing unit (CPU)A major hardware component that consists of the arithmetic/logic unit and the control unit. Also sometimes called a **processor**.

character sets The list of characters available for data and program statements.

check buttonA window component with a label and a control that the user can select or deselect, and which can be selected concurrently with other check buttons in the window.



```
print(<expression>, end = "")
```



```
<variable identifier> = input(<a string prompt>)
```

class diagramA graphical notation that describes the relationships among the classes in a software system.

class hierarchy An arrangement of classes that shows the subclass/superclass/inheritance relationships among them.

Client An agent that requests and receives some service.

Client/server applications A type of application that allows many agents to receive service from one provider.

command buttonA window component that allows the user to execute a command by pressing or clicking it with the mouse.

compilerA computer program that automatically converts instructions in a high-level language to machine language.

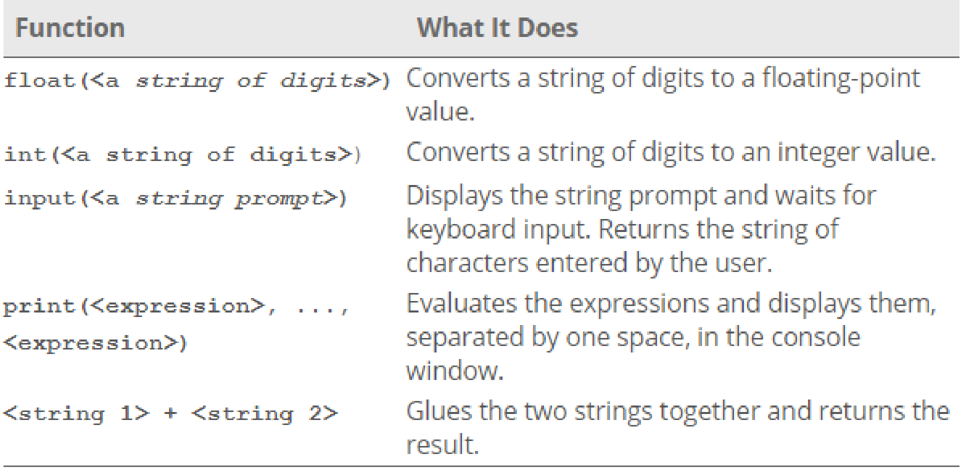
computing agent The entity that executes instructions in an algorithm.

concurrent processing The simultaneous performance of two or more tasks.

Condition A Boolean expression used to control the flow of a computation

condition-controlled loopA type of loop whose continuation depends on the value of a Boolean expression.

conditional iteration A type of loop that continues



Function	What It Does
`float(<a string of digits>)`	Converts a string of digits to a floating-point value.
`int(<a string of digits>)`	Converts a string of digits to an integer value.
`input(<a string prompt>)`	Displays the string prompt and waits for keyboard input. Returns the string of characters entered by the user.
`print(<expression>, ..., <expression>)`	Evaluates the expressions and displays them, separated by one space, in the console window.
`<string 1> + <string 2>`	Glues the two strings together and returns the result.

as long as a condition is true.

continuation conditionA Boolean expression that is checked to determine whether or not to continue iterating within a loop. If this expression is true, iteration continues.

constructorA method that is run when an object is instantiated, usually to initialize that object’s instance variables. This method is named `___init___` in Python.

conversion functions An operation that transforms one type of data into another type of data.

correct programA program whose outputs match those expected for the corresponding inputs.

CRT (Cathode Ray Tube) screen The first type of display device used to show computer output to users.

current working directory The directory to which a running program is attached, in which a file can be accessed directly by its name.

Customer requestA description of the functions of a program for its intended users, as provided by the party purchasing the software.

dataThe symbols that are used to represent information in a form suitable for storage, processing, and communication.

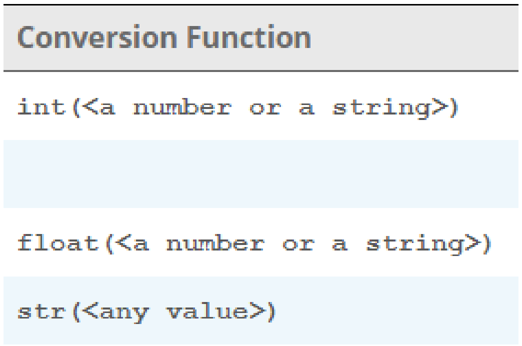
data science The discipline of gathering and analyzing massive amounts of data.

data typeA set of values and operations on those values.

definingThe process whereby a variable receives its initial value.

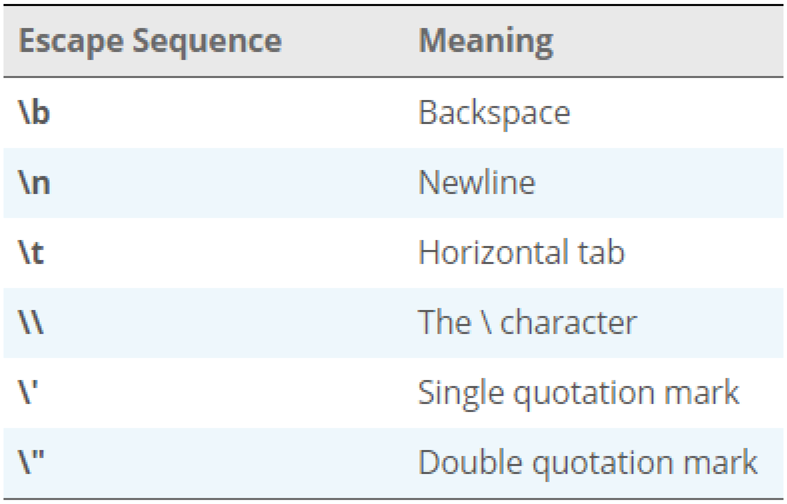
DesignThe phase of the software life cycle in which the programmer describes how the program will accomplish its tasks.

design error An error such that a program runs but produces unexpected results. Also referred to as a logic error. *See also* **syntax error**.

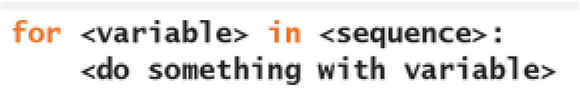


Conversion Function
`int(<a number or a string>)`
`float(<a number or a string>)`
`str(<any value>)`

decimal notation The use of the decimal digits 0..9 and a decimal point in representing real numbers in a program.



Escape Sequence	Meaning
\b	Backspace
\n	Newline
\t	Horizontal tab
\\	The \ character
\'	Single quotation mark
\"	Double quotation mark



```
for <variable> in <sequence>:
    <do something with variable>
```

default argumentsA special type of argument that is automatically provided if the caller does not supply one.

default behavior Behavior that is expected and provided under normal circumstances.

docstringA sequence of characters enclosed in triple quotation marks (""") that Python uses to document program components such as modules, classes, methods, and functions.

empty string A string that contains no characters.

end-of-line comments Part of a single line of text in a program that is not executed, but that serves as documentation for readers.

entries*See* **association**.

entry-control loopA type of loop whose continuation condition is tested at the beginning of the loop.

entry fieldsA rectangular box that supports the input and output of a single line of text.

escape sequenceA sequence of two characters in a string, the first of which is /. The sequence stands for another character, such as the tab or newline.

event-driven programming The programming of operations that handle events.

event handlersA method that is triggered when an event occurs.

ExpressionsA description of a computation that produces a value.

extendsThe process whereby a given class becomes a subclass of another class, thereby inheriting its attributes and behavior.

externalAlso called **secondary memory**, a device such as a hard drive or flash stick where data can be backed up or stored permanently.

Fibonacci numberA series of numbers generated by taking the sum of the previous two numbers in the series. The series begins with the numbers 1, 1, and 2.

field width The number of columns used for the output of text.

file dialogsA type of dialog that allows the user to browse the file system to open or save a file.

file system Software that organizes data on secondary storage media.

filteringThe successive application of a Boolean function to a sequence of arguments that returns a sequence of the arguments that make this function return True.

first-class data objects Data objects that can be passed as arguments to functions and returned as their values.

floating-pointA data type that represents real numbers in a computer program.

format operator %The operator %, when used with a format string and a set of one or more data values, returns a string with the given format.

format stringA special syntax within a string that allows the programmer to specify the number of columns within which data are placed in a string.

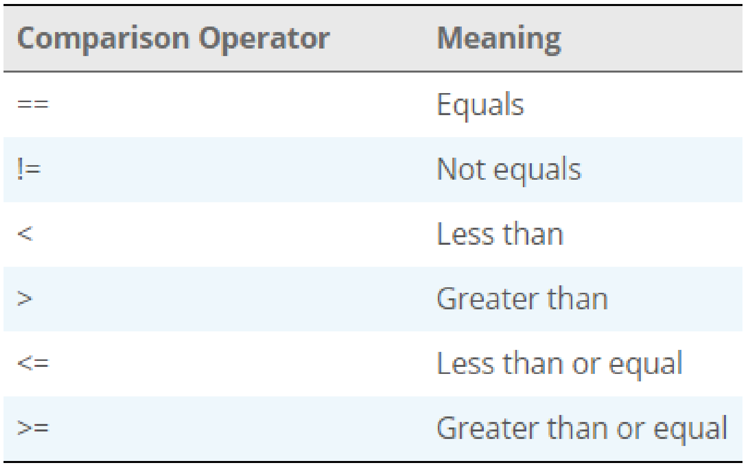
functionA chunk of code that can be treated as a unit and called to perform a task.

garbage collection The automatic process of reclaiming memory when the data of a program no longer need it.

general methodA method that solves a class of problems, not just one individual problem.

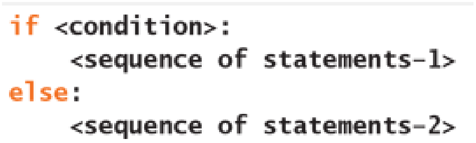
grammarThe set of rules for constructing sentences in a language.

graphical user interface (GUI)A means of communication between human beings and computers that uses a pointing device for input and a bitmapped screen for output. The bitmap displays images of windows and window objects such as buttons, text fields, and drop-down menus. The user interacts with the interface by using the mouse to directly manipulate the window objects. *See also***window object**.



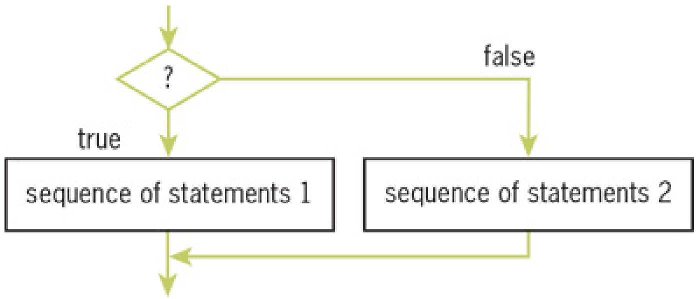
Comparison Operator	Meaning
==	Equals
!=	Not equals
<	Less than
>	Greater than
<=	Less than or equal
>=	Greater than or equal

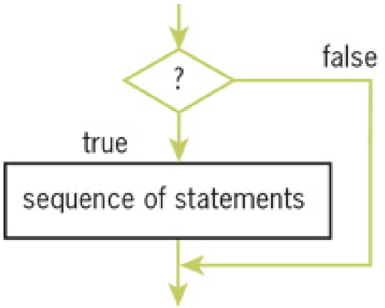
headerThe first line in a loop, selection statement, function definition, method definition, or class definition.



```
if <condition>:
    <sequence of statements-1>
else:
    <sequence of statements-2>
```

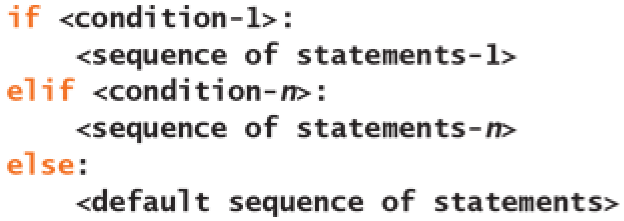
higher-order functionA function that expects another function as an argument and/or returns another function as a value.



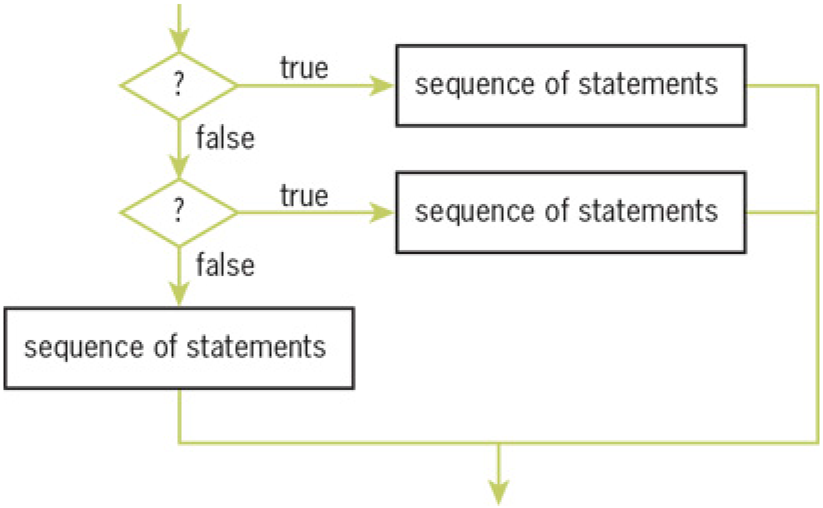


hex stringA string with the format *#RRGGBB*, where each letter is a hexadecimal digit, to represent information about an RGB color value in Python.

high-level programming languages Programming languages whose vocabulary and sentence structure are fairly close to those of English.



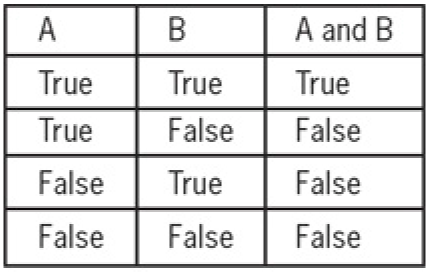
```
if <condition-1>:
    <sequence of statements-1>
elif <condition-n>:
    <sequence of statements-n>
else:
    <default sequence of statements>
```



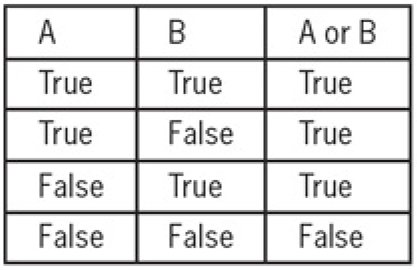
hypermediaA data structure that allows the user to access different kinds of information (text, images, sound, video, applications) by traversing links.

if statementA type of control statement that prevents a program from performing an action if the condition is false.

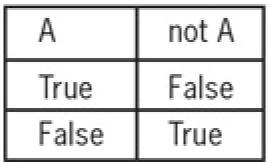
if-else statementA selection statement that allows a program to perform alternative actions based on a condition.



A	B	A and B
True	True	True
True	False	False
False	True	False
False	False	False

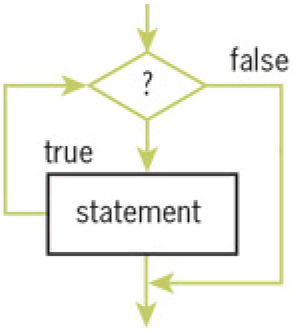


A	B	A or B
True	True	True
True	False	True
False	True	True
False	False	False



A	not A
True	False
False	True

ImplementationThe phase of the software life cycle in which the program is coded in a programming language.



incrementalThe process of developing software by gradually filling in an outline or sketch of the code, starting with minimal functionality, until the completed functionality is achieved.

indirect recursionA recursive process that results when one function calls another, which results at some point in a second call to the first function

infinite loopA loop in which the controlling condition is not changed in such a manner to allow the loop to terminate.

infinite precision The property of a real number, in which its fractional part consists of an infinite number of digits.

infinite recursion In a running program, the state that occurs when a recursive method cannot reach a stopping state.

information processing The transformation of one piece of information into another piece of information.

inheritanceThe process by which a subclass can reuse attributes and behavior defined in a superclass. *See also* **subclass** and **superclass**.

initializing*See* **defining**.

inputData obtained by a program from the external world during execution.

input/output devices Devices that allow information to be transmitted between the central processing unit of a computer and the external world.

instance variables Storage for data in an instance of a class.

integersA positive or negative whole number, or the number 0. The magnitude of an integer is limited by a computer's memory.

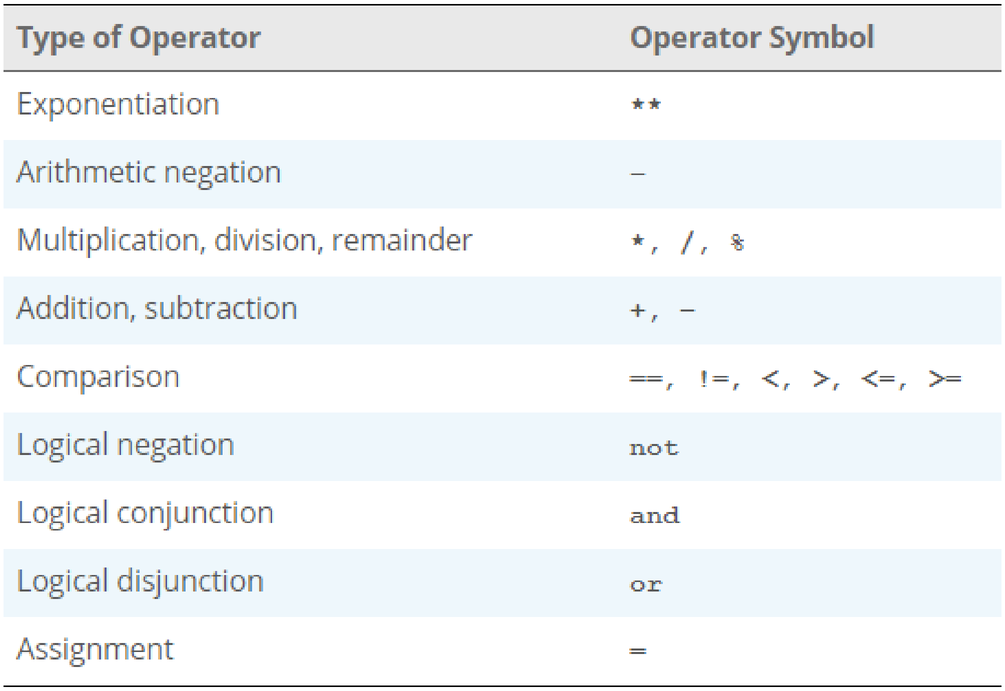
integrated circuit The arrangement of computer hardware components in a single miniaturized unit.

IntegrationThe phase of the software development life cycle during which program components are brought together and tested.

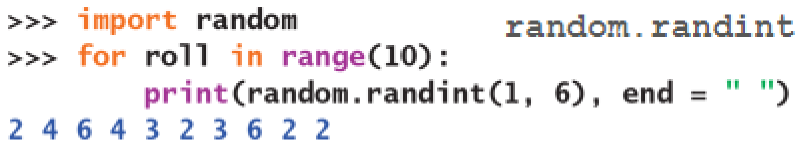
internalAlso called primary memory, a device that provides temporary storage for data and programs for fast access by a computer's central processing unit. _See also_**random access memory**.

interpreterA program that translates and executes another program.

iterativeThe process of moving forward through the phases of software development and returning to earlier phases to make improvements or corrections.



Type of Operator	Operator Symbol
Exponentiation	`**`
Arithmetic negation	`-`
Multiplication, division, remainder	`*, /, %`
Addition, subtraction	`+, -`
Comparison	`==, !=, <, >, <=, >=`
Logical negation	`not`
Logical conjunction	`and`
Logical disjunction	`or`
Assignment	`=`



random.randint

```
>>> import random
>>> for roll in range(10):
        print(random.randint(1, 6), end = " ")
2 4 6 4 3 2 3 6 2 2
```

jump tableA dictionary that associates command names with functions that are invoked when those functions are looked up in the table.

hardwareThe physical computing machine and its support devices.

keypunch machine An early input device that allowed the user to enter programs and data onto punched cards.

keysAn item that is associated with a value and which is used to locate that value in a collection.

label objectA window object that displays text or an image, usually to describe the roles of other window objects.

lifetimeThe time during which a data object or method call exists.

listA sequence of items ordered by position.

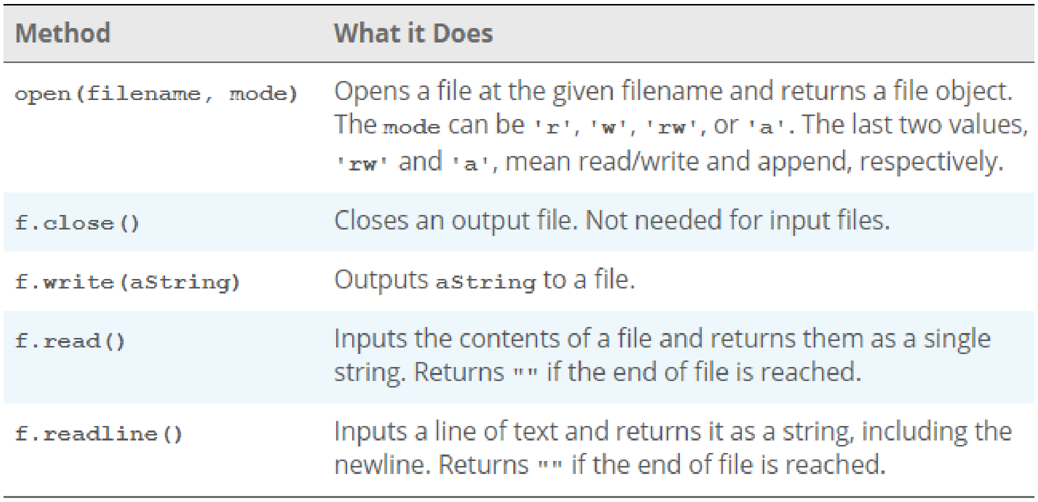
literalAn element of a language that evaluates to itself, such as 34 or "hi there."

loaderA software program that copies program code and data from secondary memory into primary memory before program execution begins.

logical negation The use of the logical operator `not` with a Boolean expression, returning `True` if the expression is false, and `False` if the expression is true.

logical operator or Any of the logical connective operators and, `or`, or `not`.

loop control variableA variable that is checked within the continuation condition of a loop



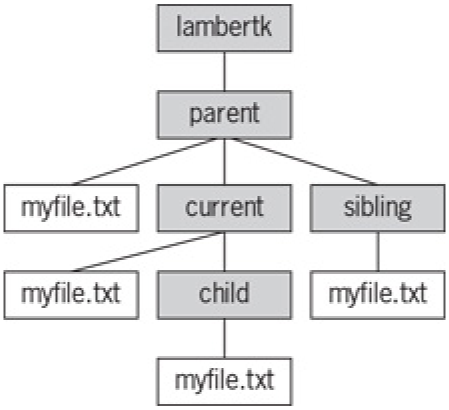
Method	What it Does
`open(filename, mode)`	Opens a file at the given filename and returns a file object. The `mode` can be `'r'`, `'w'`, `'rw'`, or `'a'`. The last two values, `'rw'` and `'a'`, mean read/write and append, respectively.
`f.close()`	Closes an output file. Not needed for input files.
`f.write(aString)`	Outputs `aString` to a file.
`f.read()`	Inputs the contents of a file and returns them as a single string. Returns `""` if the end of file is reached.
`f.readline()`	Inputs a line of text and returns it as a string, including the newline. Returns `""` if the end of file is reached.

machine code The language used directly by the computer in all its calculations and processing.

main moduleA Python module containing code that serves as the starting point of program execution.

mainframe computers Large computers typically used by major companies and universities

Magnetic storage media Any media that allow data to be stored



as patterns in a magnetic field.

mappingThe successive appliction of a function to a sequence of arguments that returns a sequence of results.

memoryThe ordered sequence of storage cells that can be accessed by address. Instructions and variables of an executing program are temporarily held here. *See also* **main memory** and **secondary memory**.

microprocessorA processor that incorporates the entire central processing unit on a single integrated chip.

mixed-mode arithmetic Expressions containing data of different types; the values of these expressions will be of either type, depending on the rules for evaluating them.

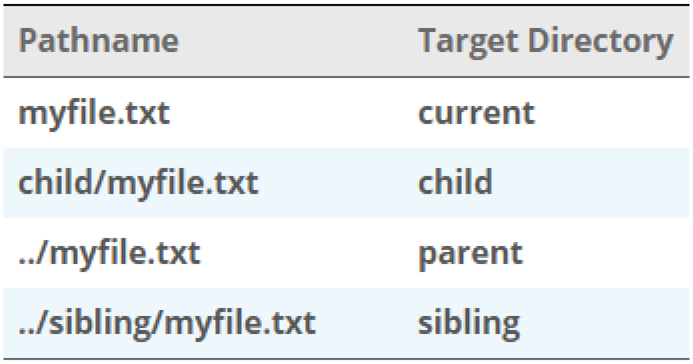
modeThe value that appears most frequently in a set of data values.

modulesAn independent program component that can contain variables, functions, and classes.

Moore's LawA hypothesis that states that the processing speed and storage capacity of computers will increase by a factor of two every 18 months.

multi-way selection statementA type of control statement that includes two or more conditions and possible courses of action.

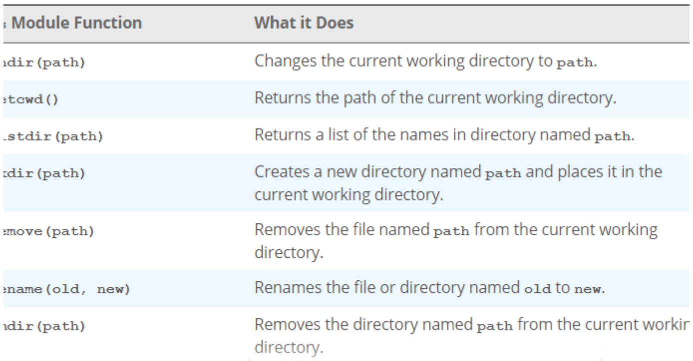
mutator methodsA method used to change the value of an attribute of an object.



Pathname	Target Directory
myfile.txt	current
child/myfile.txt	child
../myfile.txt	parent
../sibling/myfile.txt	sibling

namespaceThe set of all of a program's variables and their values.

networksA collection of resources that are linked together for communication.



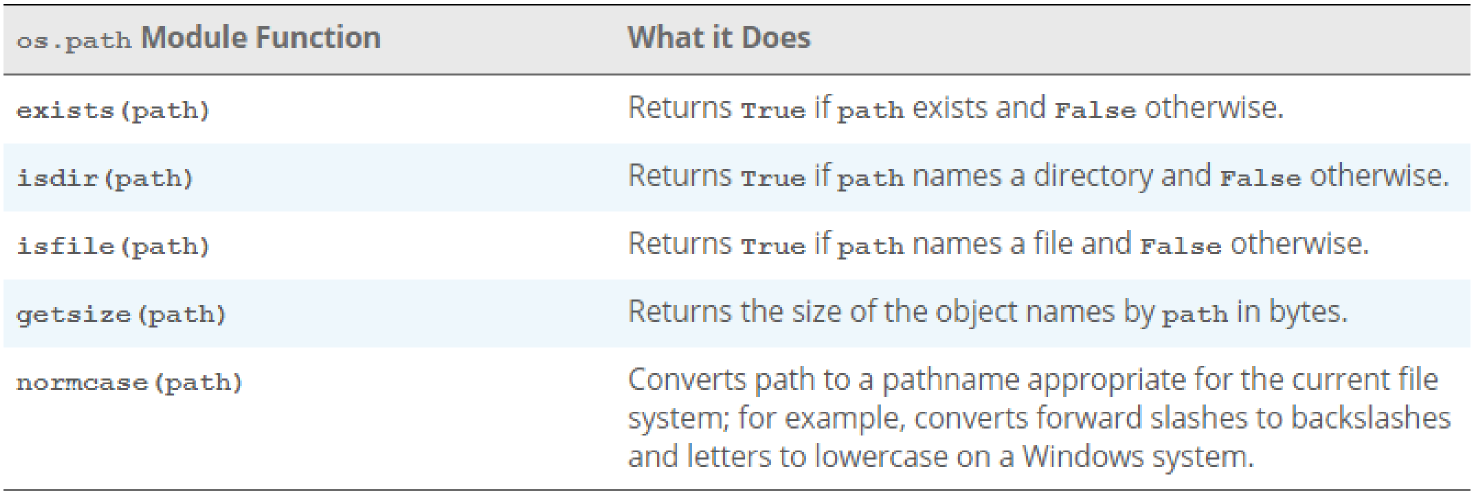
Module Function	What it Does
`dir(path)`	Changes the current working directory to `path`.
`tcwd()`	Returns the path of the current working directory.
`stdir(path)`	Returns a list of the names in directory named `path`.
`dir(path)`	Creates a new directory named `path` and places it in the current working directory.
`move(path)`	Removes the file named `path` from the current working directory.
`name(old, new)`	Renames the file or directory named `old` to `new`.
`dir(path)`	Removes the directory named `path` from the current workir directory.

newline characterA special character ('\n') used to indicate the end of a line of characters in a string or a file.

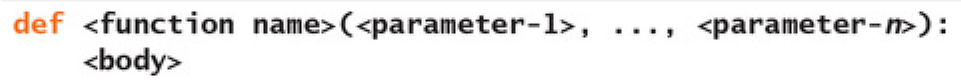
numeric data types Sets of values that represent integers or real numbers.

augmented assignment operations An assignment operation that performs a designated operation, such as addition, before storing the result in a variable.

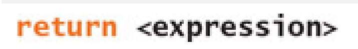
one-way selection statement*See* **`if statement`**



`os.path` Module Function	What it Does
`exists(path)`	Returns `True` if `path` exists and `False` otherwise.
`isdir(path)`	Returns `True` if `path` names a directory and `False` otherwise.
`isfile(path)`	Returns `True` if `path` names a file and `False` otherwise.
`getsize(path)`	Returns the size of the object names by `path` in bytes.
`normcase(path)`	Converts path to a pathname appropriate for the current file system; for example, converts forward slashes to backslashes and letters to lowercase on a Windows system.



```
def <function name>(<parameter-1>, ..., <parameter-n>):
    <body>
```

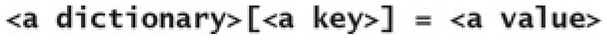


```
return <expression>
```

operating systemA large program that allows the user to communicate with the hardware and performs various management tasks.

optical storage media Devices such as CDs and DVDs that store data permanently and from which the data are accessed by using laser technology.

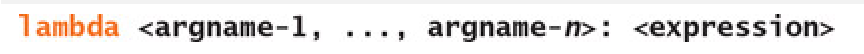
optional arguments Arguments to a function or method that may be omitted.



```
<a dictionary>[<a key>] = <a value>
```

outputInformation that is produced by a program and sent to the external world.

panelsA rectangular window component with its own grid that is useful for organizing other window components.



```
lambda <argname-1, ..., argname-n>: <expression>
```

parameters*See* **argument**

parent class The immediate superclass of a class.

pathnameA chain of directory names that allows the computer to access a file on a file system.

personal digital assistants (PDAs)A handheld device that allows the user to perform some simple tasks.

pattern matching The use of a data structure containing variables to access data within another structure.

pixelsA picture element or dot of color used to display images on a computer screen.

portsA channel through which several clients can exchange data with the same server.

precedence rules Rules that govern the order in which operators are applied in expressions.

predicateA function that returns a Boolean value.

problem instances An individual problem that belongs to a class of problems.

problem decomposition The process of breaking a problem into subproblems.

processorThe hardware components that perform computation and control the flow of execution.

prototypeA rough draft or outline of a program, which runs but without its full functionality

program comments Text in a program that is not program code, but is intended to document its structure or behavior for the human reader.

programming languagesA formal language that computer scientists use to give instructions to the computer.

program libraries Software tools or resources used in applications.

programsA set of instructions that tells the machine (the hardware) what to do.

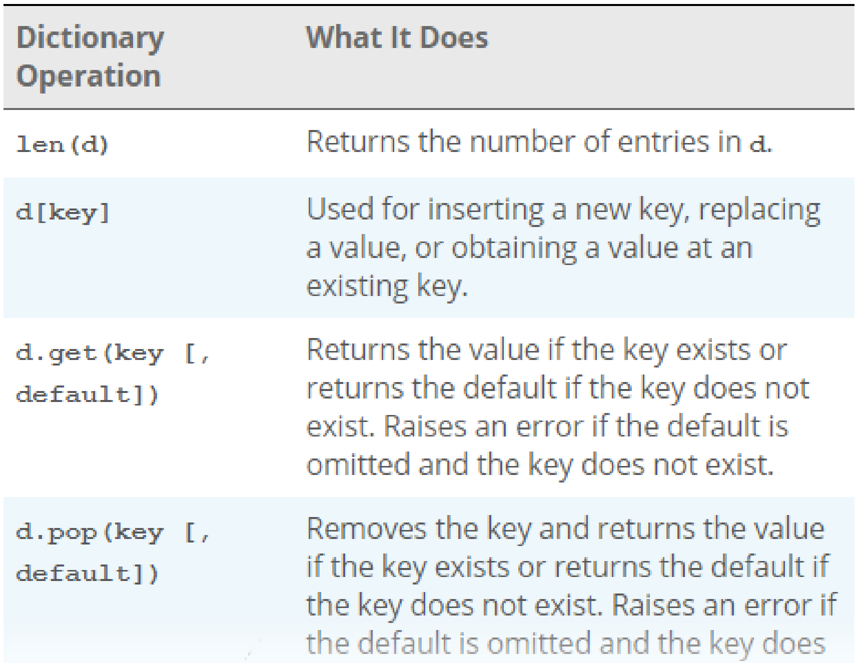
prompter boxA popup dialog box that accepts input from the user.

pseudocodeA stylized half-English, half-code language written in English but suggesting program code.

Python virtual machine (PVM)A program that interprets Python byte codes and executes them.

Python Shell An interactive program that allows the programmer to enter Python code and receive immediate feedback.

radio buttonsA window component with a label and a control that a user can select, which has the effect of deselecting the other radio buttons in the same radio



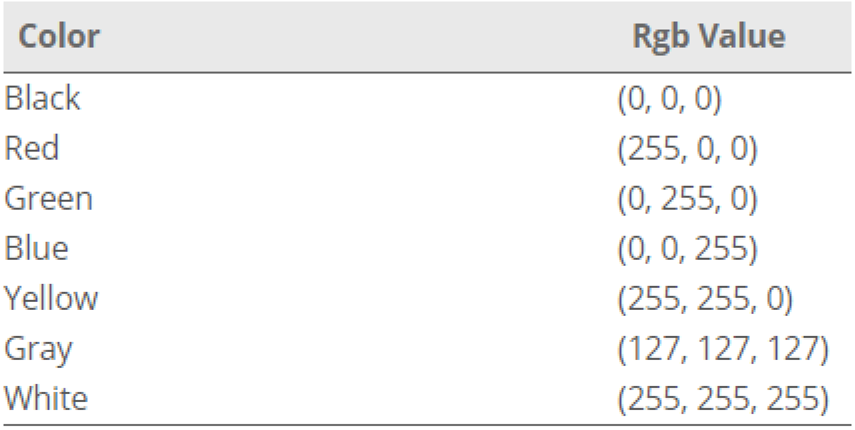
Dictionary Operation	What It Does
`len(d)`	Returns the number of entries in `d`.
`d[key]`	Used for inserting a new key, replacing a value, or obtaining a value at an existing key.
`d.get(key [, default])`	Returns the value if the key exists or returns the default if the key does not exist. Raises an error if the default is omitted and the key does not exist.
`d.pop(key [, default])`	Removes the key and returns the value if the key exists or returns the default if the key does not exist. Raises an error if the default is omitted and the key does

button group.

random access memory (RAM) Memory where a program and data are loaded for execution. Same as **primary memory**.

random numbers Numbers chosen from a given sequence to simulate randomness in a computer application.

RGB system The representation of color values using red, green, and blue components.

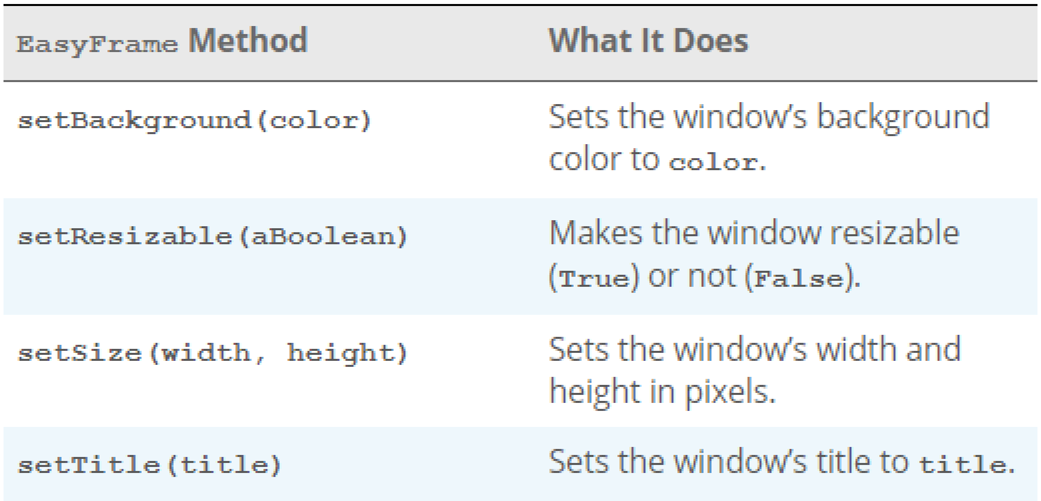


Color	Rgb Value
Black	(0, 0, 0)
Red	(255, 0, 0)
Green	(0, 255, 0)
Blue	(0, 0, 255)
Yellow	(255, 255, 0)
Gray	(127, 127, 127)
White	(255, 255, 255)

recursive call The call of a function that already has a call waiting in the current chain of function calls.

recursive definitionA set of statements in which at least one statement is defined in terms of itself.

recursive design The process of decomposing a problem into subproblems of exactly the same form that can be solved by the same algorithm.



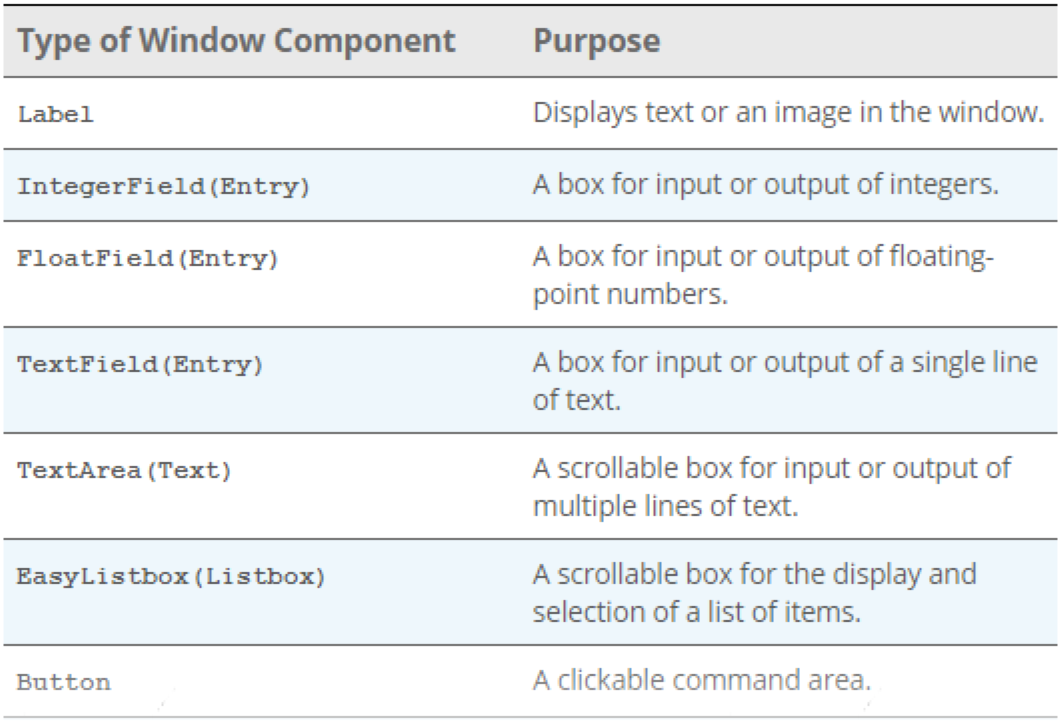
`EasyFrame` Method	What It Does
`setBackground(color)`	Sets the window's background color to `color`.
`setResizable(aBoolean)`	Makes the window resizable (`True`) or not (`False`).
`setSize(width, height)`	Sets the window's width and height in pixels.
`setTitle(title)`	Sets the window's title to `title`.

recursive functionsA function that calls itself.

recursive stepA step in the recursive process that solves a similar problem of smaller size and eventually leads to a termination of the process.

reducingThe application of a function to a sequence of its arguments to produce a single value.

regular polygonA figure of three or more sides, each of which is the same length.



Type of Window Component	Purpose
`Label`	Displays text or an image in the window.
`IntegerField(Entry)`	A box for input or output of integers.
`FloatField(Entry)`	A box for input or output of floating-point numbers.
`TextField(Entry)`	A box for input or output of a single line of text.
`TextArea(Text)`	A scrollable box for input or output of multiple lines of text.
`EasyListbox(Listbox)`	A scrollable box for the display and selection of a list of items.
`Button`	A clickable command area.

responsibility-driven design The assignment of roles and responsibilities to different actors in a program.

returning a value The process whereby a function or method makes the value that it computes available to its caller.

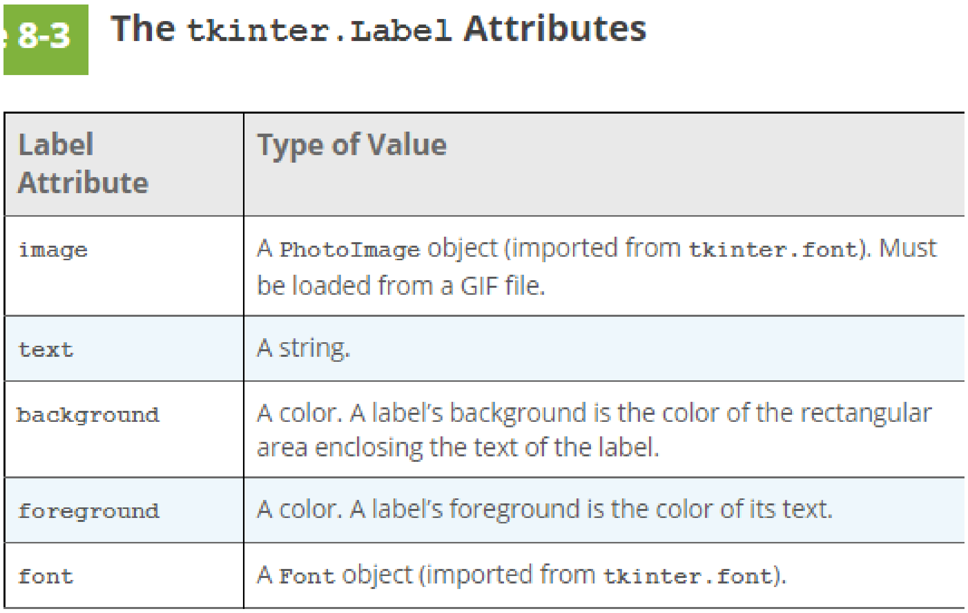
```
def someMethod(self):

anObject.someMethod()
```

root directory The directory at the top or beginning of a file system.

run-time system Software that supports the execution of a program.

scientific notation The representation of a floating-point number that uses a decimal point and an exponent to express its value.



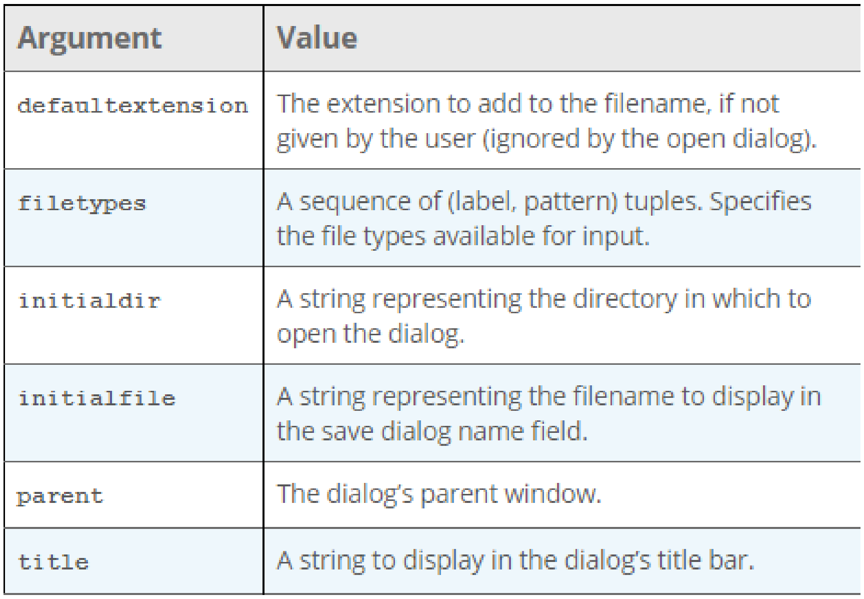
8-3 The `tkinter.Label` Attributes

Label Attribute	Type of Value
`image`	A `PhotoImage` object (imported from `tkinter.font`). Must be loaded from a GIF file.
`text`	A string.
`background`	A color. A label's background is the color of the rectangular area enclosing the text of the label.
`foreground`	A color. A label's foreground is the color of its text.
`font`	A `Font` object (imported from `tkinter.font`).

scopeThe area of program text in which the value of a variable is visible.

secondary memory An auxiliary device for memory, usually a disk or magnetic tape. *See also* **main memory** and **memory**.

selection statementsA control statement that selects some particular logical path based on the value of an expression. Also referred to as a **conditional statement**.



Argument	Value
`defaultextension`	The extension to add to the filename, if not given by the user (ignored by the open dialog).
`filetypes`	A sequence of (label, pattern) tuples. Specifies the file types available for input.
`initialdir`	A string representing the directory in which to open the dialog.
`initialfile`	A string representing the filename to display in the save dialog name field.
`parent`	The dialog's parent window.
`title`	A string to display in the dialog's title bar.

semantic errorA type of error that occurs when the computer cannot carry out the instruction specified.

semanticsThe rules for interpreting the meaning of a program in a language.

Semiconductor storage media Devices, such as flash sticks, that use solid state circuitry to store data permanently.

sentinelA special value that indicates the end of a set of data or of a process.

serverA computational object that provides a service from another computational object, usually over a network.

shellA program that allows users to enter and run Python program expressions and statements interactively.

short-circuit evaluation The process by which a compound Boolean expression halts evaluation and returns the value of the first subexpression that evaluates to true, in the case of **`or`**, or false, in the case of **`and`**.

solid-state device An electronic device, typically based on a transistor, and which has no moving parts.

softwarePrograms that make the machine (the hardware) do something, such as word processing, database management, or games.

software development The planning and organizing of a program.

software development life cycle The process of development, maintenance, and demise of a software system. Phases include analysis, design, coding, testing/ verification, maintenance, and obsolescence

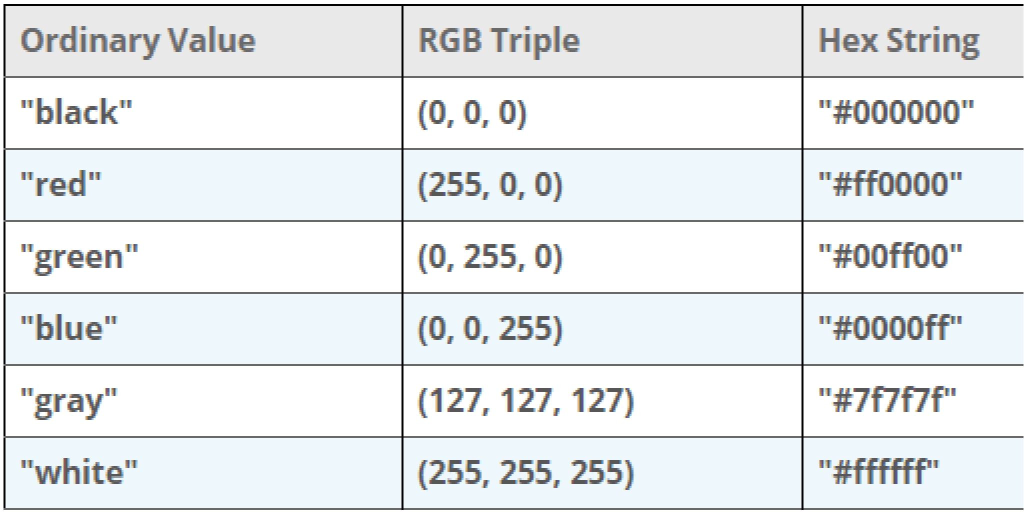
source code The program text as viewed by the human being who creates or reads it, prior to compilation.

stack frame An area of computer memory that keeps track of a function or method call's parameters, local values, return value, and the caller's return address.

step value The amount by which a counter is incremented or decremented in a countcontrolled loop.

stringA sequence of zero or more characters enclosed in quote marks.

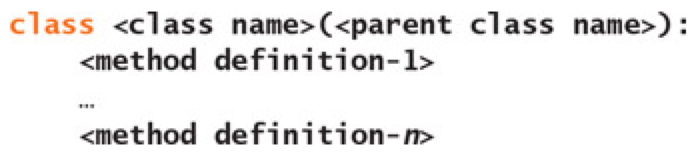
stepwise refinement The process of repeatedly subdividing tasks into subtasks until each subtask is easily accomplished. *See also* **top-down design**.



Ordinary Value	RGB Triple	Hex String
"black"	(0, 0, 0)	"#000000"
"red"	(255, 0, 0)	"#ff0000"
"green"	(0, 255, 0)	"#00ff00"
"blue"	(0, 0, 255)	"#0000ff"
"gray"	(127, 127, 127)	"#7f7f7f"
"white"	(255, 255, 255)	"#ffffff"

strongly typed programming languageA language in which the types of operands are checked prior to applying an operator to them, and which disallows such applications, either at run time or at compile time, when operands are not of the appropriate type.

subclassA class that inherits attributes and behaviors from another class.



```
class <class name>(<parent class name>):
    <method definition-1>
    …
    <method definition-n>
```

subclassingThe process of making a new class a subclass of an existing class

symbolic constantsA name that receives a value at program start-up and whose value cannot be changed.

syntaxThe form or structure of a sentence in a programming language.

syntax errors An error in spelling, punctuation, or placement of certain key symbols in a program. *See also***design error**.

system software The programs that allow users to write and execute other programs, including operating systems such as Windows and macOS.

tables*See* **dictionary**.

tabular format The presentation of output in columns of data that are either left-aligned or right-aligned.

terminal-based interfaceA user interface that allows the user to enter input from a keyboard and view output as text in a window. Also called a **terminal-based interface**.

termination conditionA Boolean expression that is checked to determine whether or not to stop iterating within a loop. If this expression is true, iteration stops.

test suiteA set of test cases that exercise the capabilities of a software component.

text editorA program that allows the user to enter text, such as a program, and save it in a file.

title bar The top border of a window that can contain a title and can be dragged with a mouse.

top-down designA method for coding by which the programmer starts with a top-level task and implements subtasks. Each subtask is then subdivided into smaller subtasks. This process is repeated until each remaining subtask is easily coded. *See also* **stepwise refinement**.

touchscreen interfaceA user interface that allows the user to enter input by tapping or gesturing while touching its screen.

transistorA device with no moving parts that can hold an electromagnetic signal and that is used to build computer circuitry for memory and a processor.

translatorA program that converts a program written in one language to an equivalent program in another language.

true color The use of enough color values that the human eye cannot distinguish adjacent colors on the scale.

truth tableA means of listing all of the possible values of a Boolean expression.

type conversion functionA function that takes one type of data as an argument and returns the same data represented in another type.

two-way selection statement*See* **`if-else` statement**.

Unicode setA character set that uses 16 bits to represent over 65,000 possible characters. These include the ASCII character set as well as symbols and ideograms in many international languages. *See also***ASCII character set**.

user interfaces Software and hardware devices that present information to human users and receive input data or commands from them.

valuesAn item that is associated with a key and is located by a key in a collection.

variable identifierA memory location, referenced by an identifier, whose value can be changed during execution of a program

variable references The process whereby the computer looks up and returns the value of a variable.

vector graphics The drawing of simple two-dimenional shapes.

virtual machineA software tool that behaves like a high-level computer.

virtual realityA technology that allows a user to interact with a computer-generated environment, usually simulating movement in three dimensions.

vocabularyThe set of words in a language.

waterfall modelA series of steps in which a software system trickles down from analysis to design to implementation. *See also***software development life cycle**.

Web applicationsA program that runs on a remote server but uses clients' Web browsers to deliver them services.

Web client Software on a computer that makes requests for resources and receives them from the Web.

Web servers Software on a computer that responds to requests for resources and makes them available on the Web.

while loopA pretest loop that examines a Boolean expression before causing a statement to be executed.

widgetsA computational object that displays an image, such as a button or a text field, in a window and supports interaction with the user.

windowA rectangular area of a computer screen that can contain window objects. Windows typically can be resized, minimized, maximized, zoomed, or closed.

The names in this code fall into four categories, depending on where they are introduced:

1. **Module variables**. The names **replacements** and **changePerson**are introduced at the level of the module. Although **replacements**names a dictionary and **changePerson** names a function, they are both considered variables. You can see the module variables of the **doctor** module by importing it and entering **dir(doctor)** at a shell prompt. When module variables are introduced in a program, they are immediately given a value.
2. **Parameters**. The name **sentence** is a parameter of the function **changePerson**. A parameter name behaves like a variable and is introduced in a function or method header. The parameter does not receive a value until the function is called.
3. **Temporary variables**. The names **words**, **replyWords**, and **word**are introduced in the body of the function **changePerson**. Like module variables, temporary variables receive their values as soon as they are introduced.
1. **Method names**. The names **split** and **join** are introduced or defined in the **str** type. As mentioned earlier, a method reference always uses an object, in this case, a string, followed by a dot and the method name.

A window has several attributes. The most important ones are its

- title (an empty string by default)
- width and height in pixels
- resizability (true by default)
- background color (white by default)

Rules of Thumb for Defining a Simple Class

We conclude this section by listing several rules of thumb for designing and implementing a simple class:

1. Before writing a line of code, think about the behavior and attributes of the objects of the new class. What actions does an object perform, and how, from the external perspective of a user, do these actions access or modify the object's state?
2. Choose an appropriate class name, and develop a short list of the methods available to users. This interface should include appropriate method names and parameter names, as well as brief descriptions of what the methods do. Avoid describing how the methods perform their tasks.
3. Write a short script that appears to use the new class in an appropriate way. The script should instantiate the class and run all of its methods. Of course, you will not be able to execute this script until you have completed the next few steps, but it will help to clarify the interface of your class and serve as an initial test bed for it.
4. Choose the appropriate data structures to represent the attributes of the class. These will be either built-in types such as integers, strings, and lists, or other programmer-defined classes.
5. Fill in the class template with a constructor (an **__init__** method) and an **__str__** method. Remember that the constructor initializes an object's instance variables, whereas **__str__** builds a string from this information. As soon as you have defined these two methods, you can test your class by instantiating it and printing the resulting object.
6. Complete and test the remaining methods incrementally, working in a bottom-up manner. If one method depends on another, complete the second method first.
7. Remember to document your code. Include a docstring for the module, the class, and each method. Do not add docstrings as an afterthought. Write them as soon as you write a class header or a method header. Be sure to examine the results by running **help**with the class name.