APPENDIX A

Python Library Reference

THE following tables provide a convenient reference for the most common Python functions and methods used in this book. You can find a complete reference for the Python standard library at `https://docs.python.org/3/library/` .

A.1 MATH MODULE

This table lists commonly used functions and constants in the `math` module. The variable name `x` represents a generic numerical argument. Arguments in square brackets are optional.

`acos(x)`	returns the arccosine of `x` ($\cos^{-1} x$)
`asin(x)`	returns the arcsine of `x` ($\sin^{-1} x$)
`atan(x)`	returns the arctangent of `x` ($\tan^{-1} x$)
`atan2(y, x)`	returns the arctangent of `y/x` ($\tan^{-1}(y/x)$)
`cos(x)`	returns the cosine of `x` radians ($\cos x$)
`degrees(x)`	returns the number of degrees in `x` radians
`exp(x)`	returns $e^{\mathtt{x}}$
`log(x, [b])`	returns the logarithm base `b` of `x` ($\log_{\mathtt{b}} \mathtt{x}$); if `b` is omitted, returns the natural logarithm of `x` ($\ln \mathtt{x}$)
`radians(x)`	returns the number of radians in `x` degrees
`sin(x)`	returns the sine of `x` radians ($\sin x$)
`sqrt(x)`	returns the square root of `x` ($\sqrt{x}$)
`tan(x)`	returns the tangent of `x` radians ($\tan x$)
`e`	the value of e (Euler's number), the base of the natural logarithm
`inf`	a value representing ∞
`pi`	the value of π

A.2 TURTLE METHODS

This table lists commonly used methods of the `Turtle` class (in the `turtle` module). The following descriptions assume default settings: angles are in degrees and right turns are clockwise. Arguments in square brackets are optional.

`backward(distance)`	moves turtle `distance` opposite to its current direction
`begin_fill()`	marks the beginning of a shape to be filled
`circle(radius, [extent, steps])`	draws a circle with given `radius`; if `extent` is given, draws an arc of `extent` degrees; if `steps` is given, draw a regular polygon with `steps` sides
`dot([size, color])`	draws a dot with diameter `size` in given `color` (defaults `1, 'black'`)
`down()`	puts the turtle's tail down, enabling drawing
`end_fill()`	fills the shape drawn since the last call to `begin_fill()`
`fillcolor(color)`	sets the turtle's fill color to `color` (see Tangent 2.1)
`forward(distance)`	moves turtle `distance` forward in its current direction
`getscreen()`	returns the `Screen` object in which the turtle is drawing
`goto(x, y)`	moves turtle to position (`x`, `y`) without changing heading
`heading()`	returns the turtle's heading
`hideturtle()`	hides the turtle while drawing
`home()`	moves turtle to the origin and resets to original heading
`left(angle)`	turns turtle `angle` degrees counterclockwise
`pencolor(color)`	sets the turtle's pen color to `color` (see Tangent 2.1)
`pensize(width)`	sets the pen to the given width
`position()`	returns the turtle's current position as a tuple
`right(angle)`	turns turtle `angle` degrees clockwise
`setheading(angle)`	sets turtle's heading to `angle` degrees
`speed(s)`	sets turtle's speed to `s`, a number 0 to 10; 1 is slowest, 10 is fast, 0 is fastest
`up()`	puts the turtle's tail up, disabling drawing
`write(message)`	writes `message` at the current turtle position
`xcor()`	returns the turtle's x coordinate
`ycor()`	returns the turtle's y coordinate

A.3 SCREEN METHODS

This table lists commonly used methods of the `Screen` class (in the `turtle` module). Arguments in square brackets are optional.

`bgcolor(color)`	sets the color of the background to `color`
`bgpic(filename)`	sets the background to contain the named GIF image
`colormode(mode)`	if `mode` is 1.0, RGB colors are specified by numbers between 0 and 1.0; if `mode` is 255, they are specified by numbers between 0 and 255
`exitonclick()`	causes the drawing window to close when clicked
`mainloop()`	must be called at the end of any program handling mouse clicks or other events
`onclick(function)`	call `function` when there is a mouse click; `function` must take `x` and `y`, the location of the click, as parameters
`setup(width, height, startx, starty)`	sets the size and location of the drawing window
`setworldcoordinates (x1, y1, x2, y2)`	sets the coordinates of the window with (`x1`, `y1`) at the lower left and (`x2`, `y2`) at the upper right
`tracer(n)`	only perform every `n`th screen update
`update()`	updates the screen to reflect all drawing so far
`window_height()`	returns the height of the drawing window
`window_width()`	returns the width of the drawing window

A.4 MATPLOTLIB.PYPLOT MODULE

This table lists commonly used functions in the `matplotlib.pyplot` module. The parameters in square brackets are optional. For a complete reference, see `http://matplotlib.org/api/pyplot_summary.html` .

`bar(x, y)`	creates a bar graph with the given `x` and `y` values
`hist(values, [bins])`	creates a histogram of `values` using the given number of bins (default is 10)
`legend()`	creates a legend using labels from the plotting calls
`plot(x, y, [options])`	creates a line graph with the given lists of `x` and `y` values; common optional keyword arguments are `color = 'blue'` or another color string `linewidth = 2` or another width `linestyle = 'dashed'` or `'solid'` or `'dotted'` `label = 'mylabel'` (used by `legend()`)
`scatter(x, y, [options])`	creates a scatter plot with the given lists of `x` and `y` values; common options are `color` and `label`
`title(titlestring)`	sets the title of the graph to be `titlestring`
`xlabel(xstring)`	labels the x axis of the current graph with `xstring`
`xticks(range, [labels], [options])`	set the locations (and optionally, labels) of the ticks on the x-axis
`ylabel(ystring)`	labels the y axis of the current graph with `ystring`
`yticks(range, [labels], [options])`	set the locations (and optionally, labels) of the ticks on the y-axis

A.5 RANDOM MODULE

The following table lists commonly used functions in the `random` module.

`gauss(mean, stdDev)`	returns a value according to the Gaussian (i.e., normal) distribution with the given mean and standard deviation
`random()`	returns a pseudorandom number in [0,1)
`randrange(start, stop, step)`	returns a randomly selected integer value from `range(start, stop, step)`
`seed(s)`	sets the seed for the PRNG; default is the current time
`uniform(a, b)`	returns a pseudorandom number in [`a`,`b`]

A.6 STRING METHODS

The following table lists commonly used methods of the `str` class. Optional parameters are denoted in square brackets.

`count(substring)`	returns number of times `substring` appears in the string
`endswith(substring)`	returns `True` if the string ends with `substring` and `False` otherwise
`find(substring)`	returns the index of the first instance of `substring` in the string, or -1 if `substring` is not found
`lower()`	returns a copy of the string with all letters in lowercase
`lstrip([chars])`	returns a copy of the string with all instances of the characters in the string `chars` removed from its beginning; if `chars` is omitted, whitespace characters are removed
`replace(old, new)`	returns a copy of the the string with all instances of the string `old` replaced with the string `new`
`rstrip([chars])`	returns a copy of the string with all instances of the characters in the string `chars` removed from its end; if `chars` is omitted, whitespace characters are removed
`split([sep])`	returns a list of "words" in the string that are separated by the delimiter string `sep`; if `sep` is omitted, the string is split at runs of whitespace characters
`startswith(substring)`	returns `True` if the string starts with `substring` and `False` otherwise
`strip([chars])`	returns a copy of the string with all leading and trailing instances of the characters in the string `chars` removed; if `chars` is omitted, whitespace characters are removed
`upper()`	returns a copy of the string with all letters in uppercase

A.7 LIST METHODS

The following table lists commonly used methods of the `list` class and three list functions from the `random` module. Optional parameters are in square brackets.

`append(item)`	appends `item` to the end of the list; returns `None`
`clear()`	clears the contents of the list
`copy()`	returns a shallow copy of the list
`count(item)`	returns number of times `item` appears in the list
`extend(items)`	appends all of the values in the list named `items` to the end of the list; returns `None`
`index(item)`	returns the index of the first occurrence of `item` in the list; raises a `ValueError` if `item` is not found
`insert(index, item)`	inserts `item` in the list at `index`; returns `None`
`pop([index])`	deletes the item in position `index` from the list and returns it; if `index` is omitted, deletes and returns the last item in the list
`remove(item)`	removes the first instance of `item` from the list; returns `None`; raises `ValueError` if `item` is not found
`reverse()`	reverses the items in the list in place; returns `None`
`sort([key, reverse])`	sorts the list in place using a stable sort; if provided, `key` is a function that returns a key to be used for a list item in the sort; if `reverse` is `True`, the list is sorted in reverse order; returns `None`
`random.choice(data)`	returns a random element from the list `data`
`random.sample(data, k)`	returns a list of `k` unique elements from list `data`
`random.shuffle(data)`	shuffles the list `data` in place; returns `None`

A.8 IMAGE MODULE

The module `image.py` which contains the `Image` class is available on the book website. The first table lists the functions in the `image` module.

`Image(width, height, [title = 'Title'])`	returns a new empty `Image` object with the given `width` and `height`; optionally sets the `title` of the image window displayed by `show`
`Image(file = 'file.gif', [title = 'Title'])`	returns a new `Image` object containing the image in the given GIF `file`; optionally sets the `title` of the image window displayed by `show`
`mainloop()`	waits until all image windows have been closed, then quits the program

The second table lists the methods of the `Image` class.

`get(x, y)`	returns a tuple representing the RGB color of the pixel at coordinates `(x,y)` of the image
`height()`	returns the height of the image
`save(fileName)`	saves the image as a GIF file with the given `fileName`
`set(x, y, color)`	sets the color of the pixel at coordinates `(x,y)` of the image to `color` (a RGB tuple)
`show()`	displays the image in its own window
`width()`	returns the width of the image
`update()`	updates the image in its existing window

A.9 SPECIAL METHODS

The following table lists commonly used special methods that may be overridden in new classes.

Method	Called by	Comments
`__init__(self)`		class constructor
`__str__(self)`	`str(self)`	string representing `self`
`__lt__(self, other)`	`self < other`	
`__le__(self, other)`	`self <= other`	
`__gt__(self, other)`	`self > other`	
`__ge__(self, other)`	`self >= other`	
`__eq__(self, other)`	`self == other`	
`__ne__(self, other)`	`self != other`	
`__len__(self)`	`len(self)`	length of `self`
`__getitem__(self, index)`	`self[index]`	returns the item in `self` at `index`
`__setitem__(self, index, value)`	`self[index] = value`	assigns `value` to the item in `self` at `index`
`__delitem__(self, index)`	`del self[index]`	deletes the item in `self` at `index`
`__contains__(self, item)`	`item in self`	returns whether `item` is in `self`
`__add__(self, other)`	`self + other`	
`__sub__(self, other)`	`self - other`	
`__mul__(self, other)`	`self * other`	
`__truediv__(self, other)`	`self / other`	true division
`__floordiv__(self, other)`	`self // other`	floor division