This document describes the Application Programming Interface (API) for the libraries used throughout the book *Introduction to Programming in Python: An Interdisciplinary Approach* by Robert Sedgewick, Kevin Wayne, and Robert Dondero.

color.Color

`Color(r=0, g=0, b=0)`	constructs a color `c` given its red, green, and blue components
`c.getRed()`	returns the red component of `c`
`c.getGreen()`	returns the green component of `c`
`c.getBlue()`	returns the blue component of `c`
`c.luminance()`	returns the luminance of `c`
`c.toGray()`	returns the grayscale equivalent of `c`
`c.isCompatible(d)`	returns `True` if `c` is compatible with `d`, and `False` otherwise
`str(c)`	returns a string representation of `c`

instream.InStream

`InStream(fileOrUrl=None)`	constructs an input stream `i` from a file/URL or standard input if the argument is empty
`i.isEmpty()`	returns `True` if `i` is empty, and `False` otherwise
`i.readInt()`	returns a token from `i` as an integer
`i.readAllInts()`	returns the remaining tokens from `i` as a list of integers
`i.readFloat()`	returns a token from `i` as a float
`i.readAllFloats()`	returns the remaining tokens from `i` as a list of floats
`i.readBool()`	returns a token from `i` as a boolean
`i.readAllBools()`	returns the remaining tokens from `i` as a list of booleans
`i.readString()`	returns a token from `i` as a string
`i.readAllStrings()`	returns the remaining tokens from `i` as a list of strings
`i.hasNextLine()`	returns `True` if `i` has a next line, and `False` otherwise
`i.readLine()`	returns a line of tokens from `i` as a string
`i.readAllLines()`	returns the remaining lines of tokens from `i` as a list of strings
`i.readAll()`	returns the remaining tokens from `i` as a string

outstream.OutStream

`OutStream(file=None)`	constructs an output stream `o` from a file or standard output if the argument is empty
`o.writeln(x="")`	writes `x` followed by newline to `o`
`o.write(x="")`	writes `x` to `o`
`o.writef(fmt, *args)`	writes each element of `args` to `o` according to the format specified by the string `fmt`

picture.Picture

`Picture(file)`	constructs a picture `p` from an image (`.jpg` or `.png`) file
`Picture(width=512, height=512)`	constructs a picture `p` given its dimensions in pixels
`p.save(file)`	saves `p` to the file with the given name

`p.width()`	returns the width of `p` in pixels
`p.height()`	returns the height of `p` in pixels
`p.get(x, y)`	returns the color of `p` at the location `(x, y)`
`p.set(x, y, c)`	sets the color of `p` at the location `(x, y)` to `c`

`stdarray`

`create1D(n, value=None)`	creates and returns a 1D list of size `n`, with each element initialized to `value`
`create2D(m, n, value=None)`	creates and returns a 2D list of size `m x n`, with each element initialized to `value`
`readInt1D()`	reads an integer n from standard input; then reads n integers also from standard input and returns them as a 1D list of size n
`readInt2D()`	reads integers m and n from standard input, and then reads mn integers also from standard input and returns them as a 2D list of size $m \times n$
`readFloat1D()`	reads an integer n from standard input, and then reads n floats also from standard input and returns them as a 1D list of size n
`readFloat2D()`	reads integers m and n from standard input, and then reads mn floats also from standard input and returns them as an 2D list of size $m \times n$
`readBool1D()`	reads an integer n from standard input, and then reads n booleans also from standard input and returns them as a 1D list of size n
`readBool2D()`	reads integers m and n from standard input, and then reads mn booleans also from standard input and returns them as a 2D list of size $m \times n$
`write1D(a)`	writes the size and elements of the 1D list `a` to standard output
`write2D(a)`	writes the size and elements of the 2D list `a` to standard output

`stdaudio`

`playSample(s)`	plays sound sample `s`
`playSamples(a)`	plays all sound samples in the list `a`
`playFile(file)`	plays all sound samples in the file whose name is `file.wav`
`save(file, a)`	saves all sound samples in the list `a` to the WAVE file whose name is `file.wav`
`read(file)`	reads and returns a list of all sound samples from the WAVE file whose name is `file.wav`
`wait()`	waits for the currently playing sound to finish

`stddraw`

`BLACK`	represents black
`BLUE`	represents blue
`CYAN`	represents cyan
`DARK_BLUE`	represents dark blue
`DARK_GRAY`	represents dark gray
`DARK_GREEN`	represents dark green
`DARK_RED`	represents dark red
`GRAY`	represents gray

`GREEN`	represents green
`LIGHT_GRAY`	represents light gray
`MAGENTA`	represents magenta
`ORANGE`	represents orange
`PINK`	represents pink
`RED`	represents red
`VIOLET`	represents violet
`WHITE`	represents white
`YELLOW`	represents yellow
`setCanvasSize(w=512, h=512)`	sets the width and height of the canvas to `w` and `h` pixels
`setXscale(min=0.0, max=1.0)`	sets the x-scale of canvas to the interval `[min, max]`
`setYscale(min=0.0, max=1.0)`	sets the y-scale of canvas to the interval `[min, max]`
`setPenRadius(r=0.005)`	sets the pen radius to `r`
`setPenColor(c=BLACK)`	sets the pen color to `c`
`setFontFamily(f="Helvetica")`	sets the font family to `f`
`setFontSize(s=12)`	sets the font size to `s`
`point(x, y)`	draws on the canvas a point at `(x, y)`
`line(x0, y0, x1, y1)`	draws on the canvas a line from `(x0, y0)` to `(x1, y1)`
`circle(x, y, r)`	draws on the canvas a circle of radius `r` centered at `(x, y)`
`filledCircle(x, y, r)`	draws on the canvas a filled circle of radius `r` centered at `(x, y)`
`rectangle(x, y, w, h)`	draws on the canvas a rectangle of width `w` and height `h` whose lower left point is `(x, y)`
`filledRectangle(x, y, w, h)`	draws on the canvas a filled rectangle of width `w` and height `h` whose lower left point is `(x, y)`
`square(x, y, r)`	draws on the canvas a square of side length `2r` centered at `(x, y)`
`filledSquare(x, y, r)`	draws on the canvas a filled square of side length `2r` centered at `(x, y)`
`polygon(x, y)`	draws on the canvas a polygon with coordinates `(x[i], y[i])`
`filledPolygon(x, y)`	draws on the canvas a filled polygon with coordinates `(x[i], y[i])`
`text(x, y, s)`	draw on canvas the string `s` centered at `(x, y)`
`picture(pic, x=None, y=None)`	draws on the canvas the picture `pic` centered at `(x, y)` or middle of the screen
`clear(c=WHITE)`	clears the canvas to color `c`
`save(f)`	saves the canvas to file `f`
`show(msec=float("inf"))`	shows the canvas and waits for `msec` milliseconds
`hasNextKeyTyped()`	returns `True` if the queue of keys the user typed is not empty, and `False` otherwise
`nextKeyTyped()`	removes and returns the first key from the queue of keys that the the user type
`mousePressed()`	return `True` if the mouse has been left-clicked, and `False` otherwise
`mouseX()`	returns the `x` coordinate of the location at which the mouse was most recently left-clicked
`mouseY()`	returns the `y` coordinate of the location at which the mouse was most recently left-clicked

`stdio`

`writeln(x="")`	writes `x` followed by newline to standard output
`write(x="")`	writes `x` to standard output
`writef(fmt, *args)`	writes each element of `args` to standard output according to the format specified by the string `fmt`
`isEmpty()`	returns `True` if standard input is empty, and `False` otherwise
`readInt()`	returns a token from standard input as an integer
`readAllInts()`	returns the remaining tokens from standard input as a list of integers
`readFloat()`	returns a token from standard input as a float
`readAllFloats()`	returns the remaining tokens from standard input as a list of floats
`readBool()`	returns a token from standard input as a boolean
`readAllBools()`	returns the remaining tokens from standard input as a list of booleans
`readString()`	returns a token from standard input as a string
`readAllStrings()`	returns the remaining tokens from standard input as a list of strings
`hasNextLine()`	returns `True` if standard input has a next line, and `False` otherwise
`readLine()`	returns a line of tokens from standard input as a string
`readAllLines()`	returns the remaining lines of tokens from standard input as a list of strings
`readAll()`	returns the remaining tokens from standard input as a string

`stdrandom`

`seed(i=None)`	seeds the random number generator using integer `i` or the current time
`uniformInt(lo, hi)`	returns an integer chosen uniformly at random from the interval `[lo, hi)`
`uniformFloat(lo, hi)`	returns a float chosen uniformly at random from the interval `[lo, hi)`
`bernoulli(p=0.5)`	returns `True` with probability `p` and `False` with probability `1 - p`
`binomial(n, p=0.5)`	returns the number of heads in `n` coin flips, each of which is heads with probability `p`
`gaussian(mu=0.0, sigma=1.0)`	returns a float from a Gaussian distribution with mean `mu` and standard deviation `sigma`
`discrete(a)`	returns an integer `i` with probability `a[i]`
`exp(lambd)`	returns a float from an exponential distribution with rate `lambd`
`choice(a)`	returns a random element from the list `a`
`sample(a, k)`	returns `k` unique random elements from the list `a`
`shuffle(a)`	shuffles the list `a`

`stdstats`

`mean(a)`	returns the average of the elements in list `a`
`var(a)`	returns the sample variance of the elements in list `a`
`stddev(a)`	returns the standard deviation of the elements in list `a`
`median(a)`	returns the median of the elements in list `a`
`plotPoints(a)`	plots the elements in list `a` as points
`plotLines(a)`	plots the elements in list `a` as line end-points
`plotBars(a)`	plots the elements in list `a` as bars