


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

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

instream.InStream

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

outstream.OutStream

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

picture.Picture

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

<code>p.get(x, y)</code>	returns the color of <code>p</code> at the location <code>(x, y)</code>
<code>p.set(x, y, c)</code>	sets the color of <code>p</code> at the location <code>(x, y)</code> to <code>c</code>

≡ stdarray

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

≡ stdaudio

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

≡ stddraw

<code>BLACK</code>	represents black
<code>BLUE</code>	represents blue
<code>CYAN</code>	represents cyan
<code>DARK_BLUE</code>	represents dark blue
<code>DARK_GRAY</code>	represents dark gray
<code>DARK_GREEN</code>	represents dark green
<code>DARK_RED</code>	represents dark red
<code>GRAY</code>	represents gray
<code>GREEN</code>	represents green
<code>LIGHT_GRAY</code>	represents light gray
<code>MAGENTA</code>	represents magenta
<code>ORANGE</code>	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 <i>x</i> -scale of canvas to the interval [min , max]
setYscale(min=0.0, max=1.0)	sets the <i>y</i> -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
writeln(fmt, *args)	writes each element of args to standard output according to the format specified by the string fmt

<code>isEmpty()</code>	returns True if standard input is empty, and False otherwise
<code>readInt()</code>	returns a token from standard input as an integer
<code>readAllInts()</code>	returns the remaining tokens from standard input as a list of integers
<code>readFloat()</code>	returns a token from standard input as a float
<code>readAllFloats()</code>	returns the remaining tokens from standard input as a list of floats
<code>readBool()</code>	returns a token from standard input as a boolean
<code>readAllBools()</code>	returns the remaining tokens from standard input as a list of booleans
<code>readString()</code>	returns a token from standard input as a string
<code>readAllStrings()</code>	returns the remaining tokens from standard input as a list of strings
<code>hasNextLine()</code>	returns True if standard input has a next line, and False otherwise
<code>readLine()</code>	returns a line of tokens from standard input as a string
<code>readAllLines()</code>	returns the remaining lines of tokens from standard input as a list of strings
<code>readAll()</code>	returns the remaining tokens from standard input as a string

☰ stdrandom

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

☰ stdstats

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