## **Python Summary**

The textbook has many useful tables describing various functions and what they do. Here I have tried to bring everything together so that you can have it as an easy reference. While I have included everything that was in the tables in the book, I have rearranged some parts to avoid duplication and added some extra commands where I thought they may be useful.

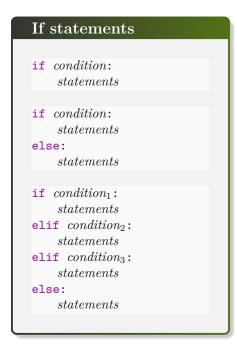


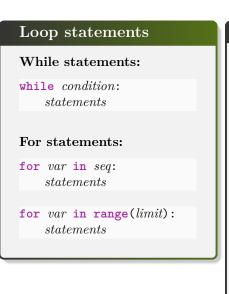
#### Contents

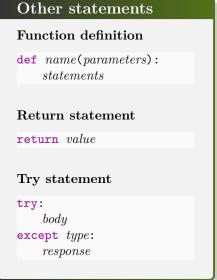
1	Common Syntax	1
2	Built-in Functions	2
3	Common Libraries	2
4	Portable Graphics Library	9
5	String Methods	7
6	List Methods	g
7	Dictionary Methods	10
8	Set Methods	10

### **───**

## 1 Common Syntax







### 2 Built-in Functions

#### **Built-in Python Functions Built-in functions** abs(x) Returns the absolute value of x. $\max(x, y, \ldots)$ Returns the largest of the arguments. Returns the smallest of the arguments. $\min(x, y, \ldots)$ round(x) Returns the closest integer to x. int(x) Converts x to an integer. float(x) Converts x to a floating-point number. len(s)Returns the length of the string argument s. str(x) Converts x to a string. str(value) Converts x to a string.

## 3 Common Libraries

pi	The mathematical constant $\pi$ .		
e 	The mathematical constant $e$ (the base for natural logarithm).		
eneral mathematical functions			
sqrt(x)	Returns the square root of $x$ .		
floor(x)	Returns the largest integer less than or equal to $x$ .		
ceil(x)	Returns the smallest integer greater than or equal to $x$ .		
copysign(x,y)	Returns $x$ with the sign of $y$ .		
	exponential functions  Returns the exponential function of $x$ ( $e^x$ )		
$\frac{\text{ogarithmic and}}{\text{exp}(x)}$	Returns the exponential function of $x$ ( $e^x$ ).		
exp(x) log(x)	Returns the exponential function of $x$ ( $e^x$ ). Returns the natural logarithm (base $e$ ) of $x$ .		
exp(x)	Returns the exponential function of $x$ ( $e^x$ ).		
$\exp(x)$ $\log(x)$ $\log 10(x)$ rigonometric for	Returns the exponential function of $x$ ( $e^x$ ). Returns the natural logarithm (base $e$ ) of $x$ . Returns the common logarithm (base $e$ ) of $x$ .		
$\exp(x)$ $\log(x)$ $\log(0)$ $\exp(x)$ $\exp(x)$ $\exp(x)$ $\exp(x)$ $\exp(x)$	Returns the exponential function of $x$ ( $e^x$ ). Returns the natural logarithm (base $e$ ) of $x$ . Returns the common logarithm (base $e$ ) of $x$ .  Inctions  Returns the cosine of the radian angle $e$		
$\exp(x)$ $\log(x)$ $\log(x)$ $\exp(x)$	Returns the exponential function of $x$ ( $e^x$ ). Returns the natural logarithm (base $e$ ) of $x$ . Returns the common logarithm (base $10$ ) of $x$ .  Inctions  Returns the cosine of the radian angle $theta$ . Returns the sine of the radian angle $theta$ .		
exp(x) log(x) log10(x)  rigonometric fu  cos(theta) sin(theta) tan(theta)	Returns the exponential function of $x$ ( $e^x$ ). Returns the natural logarithm (base $e$ ) of $x$ . Returns the common logarithm (base $10$ ) of $x$ .  Inctions  Returns the cosine of the radian angle $theta$ . Returns the sine of the radian angle $theta$ . Returns the tangent of the radian angle $theta$ .		
exp(x) log(x)	Returns the exponential function of $x$ ( $e^x$ ). Returns the natural logarithm (base $e$ ) of $x$ . Returns the common logarithm (base 10) of $x$ .  Inctions  Returns the cosine of the radian angle $theta$ . Returns the sine of the radian angle $theta$ . Returns the tangent of the radian angle $theta$ . Returns the principal arctangent of $x$ , which lies between $-\pi/2$ and $+\pi/2$ .		
$\exp(x)$ $\log(x)$ $\log(x)$ $\exp(x)$	Returns the exponential function of $x$ ( $e^x$ ). Returns the natural logarithm (base $e$ ) of $x$ . Returns the common logarithm (base $10$ ) of $x$ .  Inctions  Returns the cosine of the radian angle $theta$ . Returns the sine of the radian angle $theta$ . Returns the tangent of the radian angle $theta$ .		

Selections from the random library		
Random integers		
randint(min, max)	Returns a random integer between $min$ and $max$ , inclusive.	
randrange(limit)	Returns a random integer from 0 up to but not including <i>limit</i> .	
<pre>randrange(start, limit)</pre>	Returns a random integer from $start$ up to but not including $limit$ .	
Random floating-point numbers		
1 ()		
random()	Returns a random floating-point number in the range between 0 and 1.	
uniform(min, max)	Returns a random floating-point number in the range between 0 and 1.  Returns a random floating-point number between <i>min</i> and <i>max</i> .	
· · · · · · · · · · · · · · · · · · ·	Returns a random floating-point number between <i>min</i> and <i>max</i> .	
uniform(min, max)  Random functions on lis	Returns a random floating-point number between $min$ and $max$ .	
uniform(min, max)  Random functions on list choice(list)	Returns a random floating-point number between min and max.  ts  Returns a random element from the specified list.	
uniform(min, max)  Random functions on list choice(list) sample(list, k)	Returns a random floating-point number between $min$ and $max$ .  ts  Returns a random element from the specified list.  Returns a list with $k$ elements randomly chosen from $list$ .	
uniform(min, max)  Random functions on list choice(list) sample(list, k) shuffle(list)	Returns a random floating-point number between $min$ and $max$ .  ts  Returns a random element from the specified list.  Returns a list with $k$ elements randomly chosen from $list$ .	

# 4 Portable Graphics Library

Methods for GWindow objects		
GWindow(width, height)	Creates a new GWindow objects of the specified size.	
$gw.\mathtt{get\_width()}$	Returns the width of the graphics window.	
$gw.\mathtt{get\_height()}$	Returns the height of the graphics window.	
gw.add(obj)	Adds the object to the graphics window.	
gw.add(obj, x, y)	Repositions the object to $(x, y)$ and adds it to the window.	
$gw.\mathtt{remove}(\mathit{obj})$	Removes the object from the graphics window.	
gw.clear()	Removes all objects from the graphics window.	
<pre>gw.get_element_at(x, y)</pre>	Returns the topmost graphical object covering the point $(x,y)$ . If no such object exists then None is returned.	
<pre>gw.add_event_listener(type, func)</pre>	Prepares the window to respond to events of the specified $type$ by calling $func$ .	
<pre>gw.set_interval(func, delay)</pre>	Prepares the window to $\underline{\text{repeatedly}}$ call $func$ every $delay$ milliseconds.	
$gw.\mathtt{set\_timeout}(func,\ delay)$	Prepares the window to call $func$ once after waiting $delay$ milliseconds.	

## Creating Graphical Objects

## ${\bf Creating\ graphical\ objects}$

GRect(x, y, width, height)	Creates a GRect object with the specified dimensions.
$\mathtt{GRect}(\mathit{width}, \mathit{height})$	Creates a GRect object an $(0,0)$ with the specified size.
GOval(x, y, width, height)	Creates a GOval object that fits inside the corresponding rectangle of the specified size.
GOval(width, height)	Creates a GOval object in which the oval ifts inside a rectangle of the specified size. The origin of the GOval is at $(0,0)$ .
GLine $(x_1, y_1, x_2, y_2)$	Creates a GLine object connection $(x_1, y_1)$ and $(x_2, y_2)$ .
GLabel(str, x, y)	Creates a GLabel object containing the specified string with its baseline origin at the point $(x, y)$ .
${ t GLabel}(str)$	Creates a GLabel object containing the specified string with its baseline origin at the point $(0,0)$ .
GArc(x, y, width, height, start, sweep)	Creates a GArc object at the specified point and dimensions which starts at <i>start</i> degrees and extends counterclockwise <i>sweep</i> degrees.
GPolygon()	Creates an empty GPolygon object, which then needs vertices to be added.
GCompound()	Creates an empty GCompound object, into which other objects can then be added.

## Methods available to all objects

#### Methods that control the location

$object.\mathtt{set\_location}(x,\ y)$	Sets the location of this object to $(x, y)$ .
$object.\mathtt{move}(\mathit{dx},\;\mathit{dy})$	Moves the object using the displacements $dx$ and $dy$ .
$object.{\tt move\_polar}(r,\ theta)$	Moves the object $r$ pixels in the direction $theta$ .

#### Methods that control the appearance

object.set_color(color)	Sets the color used to display this object.
$object.\mathtt{set\_line\_width}(\mathit{width})$	Sets the width of the lines (in pixels) used to draw the object.
$object.\mathtt{set\_visible}(\mathit{flag})$	Sets whether the object is visible, where $flag$ is a boolean.
$object. { t rotate(} theta{ t )}$	Rotates the object theta degrees about its origin.
$object.\mathtt{scale}(\mathit{sf})$	Scales the object by $sf$ both horizontally and vertically.

#### Methods that control the stacking order

$object.\mathtt{send\_backward}()$	Moves this object one step backward in the stacking order.
$object.\mathtt{send\_forward}()$	Moves this object one step forward in the stacking order.
$object.\mathtt{send\_to\_back()}$	Moves this object to the back of the stacking order.
object.send_to_front()	Moves this object to the front of the stacking order.

#### Methods that return properties

object.get_x()	Returns the $x$ coordinate of the object.
object.get_y()	Returns the $y$ coordinate of the object.
$object.\mathtt{get\_width()}$	Returns the width of the object.
$object.\mathtt{get\_height()}$	Returns the height of the object.
object.get_color()	Returns the color used to display this object.
$object.\mathtt{get\_line\_width()}$	Returns the width of the lines used to draw the object.
$object. {\tt is\_visible()}$	Returns a boolean indicating whether the object is currently visible.
$object.\mathtt{contains}(x, y)$	Check to see whether the point $(x, y)$ is inside the object.

## Methods only available to GFillableObject objects

#### GFillableObjects include GRects, GOvals, GArcs, and GPolygons

$object.\mathtt{set\_filled}(bool)$	Sets whether the object is filled.
object.set_fill_color(color)	Sets the color used to fill the interior of the object.
$object.\mathtt{get\_fill\_color()}$	Returns the color used to display the interior of the object.
object.is_filled()	Returns a boolean indicating whether the object is currently filled.

### Methods only available to GLabel objects

$object.\mathtt{set\_font}(str)$	Sets the font for the label. The format of the font specification is a CSS string
	as described in the text.
$object.\mathtt{set\_label}(\mathit{str})$	Sets the text of the label to the provided string.
object.get_label()	Returns the text of the label as a string.
$object.\mathtt{get\_ascent}()$	Returns the font ascent (the maximum distance above the baseline).
<pre>object.get_descent()</pre>	Returns the font descent (the maximum distance below the baseline).

## Methods only available to GLine objects

object.set_start_point(x, y)	Changes the starting point of the line to $(x, y)$ without changing the end.
$object.\mathtt{set\_end\_point}(x,\ y)$	Changes the end point of the line to $(x, y)$ without changing the start.
<pre>object.get_start_point()</pre>	Returns the starting point of the line.
object.get_end_point()	Returns the end point of the line.

### Methods only available to GRect or GOval objects

$object.\mathtt{set\_size}(\mathit{width},\mathit{height})$	Sets the size of the object to the specified width and height.
object.set_bounds(x, y, width, height)	Sets both the location of the object and the size of the object.

#### Methods only available to GArc objects

<pre>arc.set_start_angle(start)</pre>	Sets the start angle to <i>start</i> degrees.
<pre>arc.get_start_angle()</pre>	Returns the start angle.
<pre>arc.set_sweep_angle(sweep)</pre>	Sets the sweep angle to sweep.
<pre>arc.get_sweep_angle()</pre>	Returns the sweep angle.
<pre>arc.get_start_point()</pre>	Returns the coordinate of the starting point of the arc.
<pre>arc.get_end_point()</pre>	Returns the coordinate of the ending point of the arc.

## Methods only available to GPolygon objects

$poly.add_vertex(x, y)$	Adds a vertex at the point $(x, y)$ .
$poly.$ add_edge( $dx, dy$ )	Adds a vertex shifted by $dx$ and $dy$ from the preceding vertex.
$poly.add_polar_edge(r, theta)$	Adds a vertex shifted by $r$ units in direction $theta$ .
$poly.\mathtt{get\_bounds()}$	Returns a GRect object of the bounding rectangle of the polygon.

## Methods in the GImage class

### Creating GImages

GImage(filename)	Creates a new GImage by reading the image data from the specified file.
${\tt GImage}(array)$	Creates a new GImage from the provided pixel array.

#### Method to read pixels

image.get\_pixel\_array() Returns the pixel array for this image.

## Static methods relating to GImages

GImage.get_red(pixel)	Returns the red component of the pixel as an integer between 0 and 255.
GImage.get_green(pixel)	Returns the green component of the pixel as an integer between 0 and 255.
<pre>GImage.get_blue(pixel)</pre>	Returns the blue component of the pixel as an integer between 0 and 255.
${\tt GImage.create\_rgb\_pixel}(r,\ g,\ b)$	Creates a pixel value from the specified $r$ , $g$ , and $b$ components, each of which is between 0 and 255.
<pre>GImage.create_rgb_pixel(a, r, g, b)</pre>	Creates a pixel value from the specified $r$ , $g$ , and $b$ components and with opacity $a$ , each of which is between 0 and 255.

# 5 String Methods

## Common methods in Python's string class

Finding patterns	
str.find(pattern)	Searches the string $str$ for the string $pattern$ , starting at the beginning of $str$ . Returns the first index at which the pattern appears, or $-1$ if not found.
str. find(pattern, k)	Same as above, but starts the search at index $k$ .
$str. ext{rfind}(pattern)$	Searches backward in $str$ for the last instance of $pattern$ , starting at the end of $str$ . Returns the last index at which the pattern appears, or $-1$ if not found.
str.rfind(pattern, k)	Same as above, but starts at index $k$ .
str.startswith( $prefix$ )	Returns True if $str$ starts with the characters in $prefix$ .
$str.\mathtt{endswith}(\mathit{suffix})$	Returns True if $str$ ends with the characters in $suffix$ .

#### Creating transformed strings

str.lower()	Returns a copy of $str$ will all letters converted to lowercase.
str.upper()	Returns a copy of $str$ with all letters converted to uppercase.
str.capitalize()	Returns a lowercase copy of $str$ but with the first letter capitalized.
str.lstrip()	Returns a copy of $str$ after removing any whitespace characters from the left side.
str.rstrip()	Returns a copy of $str$ after removing any whitespace characters from the right side.
$str.\mathtt{strip()}$	Returns a copy of $str$ after removing any whitespace characters from both sides.
str.replace(old, new)	Returns a copy of $str$ after replacing all instances of the string $old$ with the string $new$ .

#### Testing for character properties

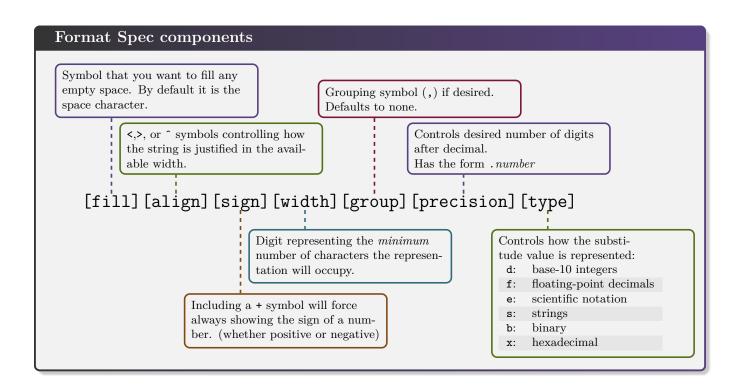
str.isalpha()	Returns True if $str$ is nonempty and contains only letters.
str.isdigit()	Returns True if $str$ is nonempty and contains only numeric digits.
str.isalnum()	Returns True if $str$ is nonempty and contains only letters or digits.
str.islower()	Returns True if $str$ has at least one letter and all letters are lowercase.
<pre>str.isupper()</pre>	Returns True if $str$ has at least one letter and all letters are uppercase.
str.isspace()	Returns True if $str$ is nonempty and contains only whitespace characters (eg. space, tab,
	or newline).

### Formatting strings

str.center(width)	Returns a copy of $str$ centered in a field of the specified $width$ .
str.ljust(width)	Returns a copy of $str$ flush to the left in a field of the specified $width$ .
$str.\mathtt{rjust}(\mathit{width})$	Returns a copy of $str$ flush to the right in a field of the specified $width$ .

### Splitting and joining strings

<pre>str.split(pattern)</pre>	Splits the $str$ into a list of strings by dividing it at $pattern$ .
$str.\mathtt{splitlines}()$	Splits a multiline string into a list of the individual lines.
sep.join(list)	Joins the elements of —list— into a string using $sep$ to separate the elements.



## 6 List Methods

## Common methods in Python's list class

### Methods that leave the original list unchanged

list.index(value)	Returns the first index matching value. This method raises a ValueError ex-
	ception if no match is found.
<pre>list.index(value, start)</pre>	Starting from <i>start</i> , returns the first index matching <i>value</i> . This method raises
	a ValueError exception if no match is found.
list.count(value)	Returns the number of times that <i>value</i> appears in the list.
list.copy()	Returns a shallow copy of the original list.

#### Methods that add or remove elements

$list. {\tt append}  (\mathit{value})$	Adds value to the end of the list.
$list.\mathtt{extend}(\mathit{list}_2)$	Adds the elements in $list_2$ to the end of the list.
list.insert( $index, value$ )	Inserts value before the specified index position.
$list.\mathtt{remove}(\mathit{value})$	Removes the first instance of value from the list, raising a ValueError excep-
	tion if it does not appear.
<pre>list.pop()</pre>	Removes and returns the last element of the list.
list.pop(index)	Removes and returns the element at the specified <i>index</i> position.
<pre>list.clear()</pre>	Removes all elements from a list.

#### Methods that reorder the elements of a list

<pre>list.reverse()</pre>	Reverses the order of the elements in the list.
<pre>list.sort()</pre>	Sort the elements of the list in ascending order.
list.sort(key)	Sort the elements of the list in ascending order according to a particular $key$ .
<pre>list.sort(key, reverse=True)</pre>	Sort the elements of the list in descending order according to a particular $key$ .

#### String methods that involve lists

str.split()	Splits the string $str$ into a list of substrings by dividing it at each occurrence of a whitespace character.
str.split(separator)	Splits the string $str$ into a list of substrings by dividing it at each instance of the string $separator$ .
$str.\mathtt{splitlines}()$	Splits $str$ into a list of substrings by dividing it at each line break.
sep.join(list)	Joins the elements of $list$ into a string using $sep$ to separate the elements.

# 7 Dictionary Methods

len(dict)	Returns the number of key-value pairs in the dictionary.
<pre>dict.clear()</pre>	Removes all key-value pairs from the dictionary, leaving it empty.
<pre>dict.copy()</pre>	Creates and returns a shallow copy of this dictionary.
dict.get(key, value)	Returns the value associated with $key$ in the dictionary, or the specified value if $key$ is not found. The default value for $value$ is None.
<pre>dict.setdefault(key, value)</pre>	If key is not in the dictionary, setdefault creates a new entry for the key value pair. The method then returns the value, which is either its previous value or the newly created one.
dict.pop(key)	Removes and returns the key-value pair corresponding to <i>key</i> . If the key is not found, then a KeyError exception is raised.
<pre>dict.items()</pre>	Returns an iterable object that returns successive tuples consisting of a key-value pair.

# 8 Set Methods

set()	Creates an empty set. The expression { } creates an empty did
	tionary.
set(iterable)	Creates a set from the elements of the iterable object. (Which could be a list, string, etc. Anything you could loop over.)
len(set)	Returns the number of elements in the set.
$set_1.\mathtt{union}(set_2)$	Returns the union of $set_1$ and $set_2$ .
$set_1 \mid set_2$	
$set_1$ .intersection( $set_2$ )	Returns the intersection of $set_1$ and $set_2$ .
$set_1 \& set_2$	
$set_1$ .difference( $set_2$ )	Returns the difference of $set_1$ and $set_2$ .
$set_1$ - $set_2$	
$set_1$ .symmetric_difference( $set_2$ )	Returns the symmetric difference of $set_1$ and $set_2$ .
$set_1$ $\hat{\ }$ $set_2$	
$set_1 == set_2$	Returns True if $set_1$ and $set_2$ contain the same elements.
$set_1 \leftarrow set_2$	Returns True if $set_1$ is a subset of $set_2$ .
$set_1 < set_2$	Returns True if $set_1$ is a proper subset of $set_2$ .
element in set	Returns True if <i>element</i> is in the <i>set</i> .
$set_1.\mathtt{isdisjoint}(set_2)$	Returns True if $set_1$ and $set_2$ contain no elements in common.
set.clear()	Removes all elements from the set, leaving it empty.
set.copy()	Creates and returns a shallow copy of the set.
set.add( $element$ )	Adds the specified element to the set.
$set.\mathtt{remove}(element)$	Removes the element from the set, raising a ValueError if it is miss
	ing. Removes the element from the set, doing nothing if it is missing.