Appendix B

Quick Reference

Table 1 Python input, output, variables

Leaving reminders by using comments	Using constants
`# whole-line comment` `print("hello") # end-of-line comment`	`SIZE = 100 # upper case name`
Printing to the screen	**Reading from the keyboard (Python V2)**

`print(10)  # print a number`
`print("hello") # print a string`
`print(a)  # print a variable`
	`name = raw_input("your name?")`
	`age  = int(raw_input("how old?"))`
	`nextYearAge = age + 1`

Storing values in variables	**Using Boolean variables**

`a = 10  # an integer (whole) number`
`b = 20.2 # a decimal number`
`message = "hello" # a string`
`finished = True  # a Boolean`
`high_score = None # no value`
	`anotherGo = True`
	`while anotherGo:`
	`  choice = int(raw_input("choice?"))`
	`  if choice == 8:`
	`  anotherGo = False`

Converting strings to (integer) numbers	**Converting numbers to strings**
`size = int(size)`	`print("your age is:" + str(age))`
Calculating with number variables	**Calculating absolute values and differences**

`b = a + 1 # addition`
`c = a - 1 # subtraction`
`d = a * 2 # multiplication`
`e = a / 2 # division`
`f = a % 2 # remainder after division`
	`absolute  = abs(-1) # absolute is 1`
	`difference = abs(x1-x2)`
	`smallest  = min(x1, x2)`
	`largest  = max(x1, x2)`

Table 2 Python loops

Looping	Using counted loops

`a = 0`
`while a<100:`
`  print(a)`
`  a = a + 1`
	`for a in range(10):`
	`  print(a) # 0..9`
	`for a in range(5, 50, 5):`
	`  print(a) # 5..45 in steps of 5`

Looping through all characters in a string	**Splitting comma-separated strings**

`name = "Charlotte"`
`for ch in name:`
`  print(ch)`
	`line = "one,two,three"`
	`data = line.split(",")`
	`for d in data:`
	`  print(d)`

Table 3 Python conditions

Using if statements	Using if/else statements

`if a==42:`
`  print("the ultimate answer")`
	`if a==5:`
	`  print("correct")`
	`else:`
	`  print("wrong")`

Using elif/else for multiple choices	**Checking multiple conditions with *and, or***

`if a==1:`
`  print("option A")`
`elif a==2:`
`  print("option B")`
`else:`
`  print("anything else")`
	`if choice>0 and choice<=8:`
	`  print("in range")`
	`if choice<1 or choice>8:`
	`  print("out of range")`

Checking for *different* or *same* with if	**Checking for *less* with if**

`if a!=1:`
`  print("not equal")`
`if a==1:`
`  print("equal")`
	`if a<1:`
	`  print("less than")`
	`if a<=1:`
	`  print("less than or equal")`

Checking opposites with *not*	**Checking for *greater* with if**

`playing = True`
`if not playing:`
`  print("finished")`
	`if a>1:`
	`  print("greater than")`
	`if a>=1:`
	`  print("greater than or equal")`

Table 4 Python functions

Defining and calling functions	Passing parameters to functions

`def myname():`
`  print("my name is Laura")`
`myname()`
`myname()`
	`def hello(name):`
	`  print("hello " + name)`
	`hello("Andrew")`
	`hello("Geraldine")`

Returning values from functions	**Writing to global variables inside functions**

`def smallest(a, b):`
`  if a<b:`
`  return a`
`  return b`
`print(smallest(10, 20))`
	`treasure_x = 0`
	
	`def build():`
	`  global treasure_x`
	`  treasure_x = 10`

Table 5 Python lists

Creating lists	Adding to the end of a list
`a = [] # an empty list` `a = [1, 2, 3]#an initialised list`	`a.append("hello")`
Printing the contents of a list	**Working out the size of a list**
`print(a)`	`print(len(a))`
Accessing items in a list by their index	**Accessing the last item in a list**
`print(a[0]) # 0=first item, 1=second`	`print(a[-1])`
Removing the last item from a list	**Looping through all items in a list**
`word = a.pop() # remove last item` `print(word)  # item just removed`	`for name in ["David","Gail","Janet"]:` `  print(name)`

Table 6 Other Python modules

Delaying for a short amount of time	Generating random numbers
`import time` `time.sleep(1) # wait 1 second`	`import random` `print(random.randint(1,100))`
Getting the current date/time	**Using maths functions**

`import datetime`
`dateNow = datetime.datetime.now()`
`import time`
`timeNow = time.time()`
	`import math`
	`radians  = math.radians(angle)`
	`sin  = math.sin(radians)`
	`cos  = math.cos(radians)`
	`squareRoot = math.sqrt(number)`

Table 7 File processing

Reading lines from a file

```
f = open("data.txt", "r")
tips = f.readlines()
f.close()
for t in tips:
  print(t)
```

Writing lines to a file

```
f = open("scores.txt", "w")
f.write("Victoria:26000\n")
f.write("Amanda:10000\n")
f.write("Ria:32768\n")
f.close()
```

Getting a list of matching filenames

```
import glob
names = glob.glob("*.csv")
for n in names:
  print(n)
```

Stripping unwanted white space from strings

```
a = "\n\n  hello  \n\n"
a = a.strip()
print(a)
```

Table 8 Accessing hardware

Configuring GPIO pins

```
import RPi.GPIO as GPIO #RaspberryPi
import anyio.GPIO as GPIO #Arduino
GPIO.setmode(GPIO.BCM)
GPIO.setup(5, GPIO.OUT) #output
GPIO.setup(6, GPIO.IN)  #input
```

Reading and writing GPIO pins

```
GPIO.output(5, True)  # on (3.3V)
GPIO.output(5, False) # off (0V)
if GPIO.input(6) == False:
  print("pressed")
```

Safely cleaning up after using the GPIO

```
try:
  do_something() # put code here
finally:
  GPIO.cleanup()
```

Using the 7-segment display module

```
import anyio.seg7 as display
LED_PINS = [10,22,25,8,7,9,11,15]
ON = False # Common-Anode
ON = True # Common-Cathode
display.setup(GPIO, LED_PINS, ON)
```

Writing characters to the 7-segment display

```
display.write("3")
display.write("A")
display.write("up")
```

Other 7-segment display functions

```
display.setdp(True) # point on
display.setdp(False) # point off
display.clear()
display.pattern([1,1,0,1,1,0,1,0])
```

Table 9 Minecraft API

These are the key functions of the Minecraft Python API. For a complete list of functions, visit: www.stuffaboutcode.com/p/minecraft-api-reference.html.

Importing the Minecraft API	Importing and using the block name constants
`import mcpi.minecraft as minecraft`	`import mcpi.block as block` `b = block.DIRT.id`
Creating a connection to Minecraft	**Posting a message to the Minecraft chat**
`mc = minecraft.Minecraft.create()`	`mc.postToChat("Hello Minecraft")`
Getting the player's tile position	**Setting the player's tile position**

`# what are the coordinates of tile`
`# that player is standing on?`
`pos = mc.player.getTilePos()`
`x = pos.x`
`y = pos.y`
`z = pos.z`
	`x = 5`
	`y = 3`
	`z = 7`
	`mc.player.setTilePos(x, y, z)`

Getting the player's position	**Setting the player's position**

`# get precise position of player`
`# in the world (e.g. x=2.33)`
`pos = mc.player.getPos()`
`x = pos.x`
`y = pos.y`
`z = pos.z`
	`# set precise position of player`
	`x = 2.33`
	`y = 3.95`
	`z = 1.23`
	`mc.setPos(x, y, z)`

Getting the height of the world	**Getting the block type at a position**
`# y position of first non-AIR block` `height = mc.getHeight(5, 10)`	`# id of block (e.g. block.DIRT.id)` `blockId = mc.getBlock(10, 5, 2)`
Setting/changing a block at a position	**Setting/changing lots of blocks in one go**
`mc.setBlock(5, 3, 2, block.DIRT.id)`	`mc.setBlocks(0,0,0, 5,5,5, block.AIR.id)`
Finding out which blocks have been hit	**Clearing any block hits**

`# which blocks hit since last time?`
`events = mc.events.pollBlockHits()`
`for e in events:`
`  pos = e.pos`
`  print(pos.x)`
	`# clear(ignore) hits since last time`
	`mc.events.clearAll()`

Table 10 Minecraft API block types

Here is a sample of the block types available in Minecraft. Where appropriate, the data values that can be used are included, and the Pi and PC/Mac columns show whether this block can be used on those platforms. For a complete list of block IDs and block data values, visit: www.stuffaboutcode.com/p/minecraft-api-reference.html.

ID	Constant	Data ID	Sub-type	Pi	PC/MAC
0	AIR	-	-	Y	Y
1	STONE	-	-	Y	Y
2	GRASS	-	-	Y	Y
3	DIRT	-	-	Y	Y
4	COBBLESTONE	-	-	Y	Y
5	WOOD_PLANKS	0	Oak	Y	Y
		1	Spruce	N	Y
		2	Birch	N	Y
		3	Jungle	N	Y
7	BEDROCK	-	-	Y	Y
8	WATER	-	-	Y	Y
9	WATER_STATIONARY	0	High	Y	Y
		..7	Low	Y	Y
10	LAVA	-	-	Y	Y
11	LAVA_STATIONARY	0	High	Y	Y
		..7	Low	Y	Y
12	SAND	-	-	Y	Y
13	GRAVEL	-	-	Y	Y
14	GOLD_ORE	-	-	Y	Y
15	IRON_ORE	-	-	Y	Y
16	COAL_ORE	-	-	Y	Y
17	WOOD	0	Oak (up/down)	Y	Y
		1	Spruce (up/down)	Y	Y
		2	Birch (up/down)	Y	Y
		3	Jungle (up/down)	N	Y
		4	Oak (east/west)	N	Y
		5	Spruce (east/west)	N	Y
		6	Birch (east/west)	N	Y
		7	Jungle (east/west)	N	Y

ID	Constant	Data ID	Sub-type	Pi	PC/MAC
		8	Oak (north/ south)	N	Y
		9	Spruce (north/ south)	N	Y
		10	Birch (north/ south)	N	Y
		11	Jungle (north/ south)	N	Y
		12	Oak (only bark)	N	Y
		13	Spruce (only bark)	N	Y
		14	Birch (only bark)	N	Y
		15	Jungle (only bark)	N	Y
18	LEAVES	1	Oak leaves	Y	Y
		2	Spruce leaves	Y	Y
		3	Birch leaves	Y	Y
20	GLASS	-	-	Y	Y
24	SANDSTONE	0	Sandstone	Y	Y
		1	Chiselled Sandstone	Y	Y
		2	Smooth Sandstone	Y	Y
35	WOOL	0	White	Y	Y
		1	Orange	Y	Y
		2	Magenta	Y	Y
		3	Light Blue	Y	Y
		4	Yellow	Y	Y
		5	Lime	Y	Y
		6	Pink	Y	Y
		7	Grey	Y	Y
		8	Light grey	Y	Y
		9	Cyan	Y	Y
		10	Purple	Y	Y
		11	Blue	Y	Y
		12	Brown	Y	Y
		13	Green	Y	Y
		14	Red	Y	Y
		15	Black	Y	Y

continued

Table 10 continued

ID	Constant	Data ID	Sub-type	Pi	PC/MAC
37	FLOWER_YELLOW	-	-	Y	Y
38	FLOWER_CYAN	-	-	Y	Y
41	GOLD_BLOCK	-	-	Y	Y
42	IRON_BLOCK	-	-	Y	Y
45	BRICK_BLOCK	-	-	Y	Y
46	TNT	0	Inactive	Y	Y
		1	Ready to explode	Y	Y
49	OBSIDIAN	-	-	Y	Y
50	TORCH	0	Standing on the floor	Y	Y
		1	Pointing east	Y	Y
		2	Pointing west	Y	Y
		3	Pointing south	Y	Y
		4	Pointing north	Y	Y
53	STAIRS_WOOD	0	Ascending east	Y	Y
		1	Ascending west	Y	Y
		2	Ascending south	Y	Y
		3	Ascending north	Y	Y
		4	Ascending east (upside down)	Y	Y
		5	Ascending west (upside down)	Y	Y
		6	Ascending south (upside down)	Y	Y
		7	Ascending north (upside down)	Y	Y
57	DIAMOND_BLOCK	-	-	Y	Y
80	SNOW_BLOCK	-	-	Y	Y
89	GLOWSTONE_BLOCK	-	-	Y	Y
246	GLOWING_OBSIDIAN	-	-	Y	N
247	NETHER_REACTOR	0	Unused	Y	N
		1	Active	Y	N
		2	Stopped/used up	Y	N

Table 11 MinecraftStuff API (MinecraftDrawing)

Importing the MinecraftStuff API	Creating the MinecraftDrawing Object

`import mcpi.minecraftstuff as ↵`
`minecraftstuff`
	`mc = minecraft.Minecraft.create()`
	`mcdrawing =`
	`  minecraftstuff.MinecraftDrawing(mc)`

Drawing a line between two points	**Getting all the block positions of a line, as a list**

`mcdrawing.drawLine(0, 0, 0,`
`  10, 10, 10, block.DIRT.id)`
	`line = mcdrawing.getLine(`
	`0,0,0,10,10,10)`
	`pos1 = line[0]`
	`print(pos1.x)`

Drawing a sphere	**Drawing a circle**
`mcdrawing.drawSphere(0, 0, 0,` `  radius, block.DIRT.id)`	`mcdrawing.drawCircle(0, 0, 0,` `  radius, block.DIRT.id)`
Drawing a flat polygon(e.g. a triangle)	

`tri = []`
`filled = True`
`tri.append(minecraft.Vec3(0,0,0))`
`tri.append(minecraft.Vec3(10,0,0))`
`tri.append(minecraft.Vec3(5,10,0))`
`mcdrawing.drawFace(tri, filled,`
`  block.DIRT.id)`	

Table 12 MinecraftStuff API (MinecraftShape)

Creating a MinecraftShape	Drawing and clearing a shape

`mc = minecraftmMinecraft.create()`
`blocks =`
`  [minecraftstuff.ShapeBlock(0,0,0,`
`  block.DIRT.id),`
`  minecraftstuff.ShapeBlock(1,0,0,`
`  block.DIRT.id)]`
`pos = Vec3(0,0,0)`
`shape =`
`  minecraftstuff.MinecraftShape(mc,`
`  pos, blocks)`
	`shape.draw()`
	
	`shape.clear()`

Moving a shape to a position	**Moving a shape by a number of blocks**
`shape.move(10, 10, 10)`	`ymove = 1` `shape.moveBy(0, ymove, 0)`