Python Code By Mission

Mission 2 – Introducing CodeX	
Import codex	`from codex import *`
Display a built-in image	`display.show(pics.HEART)`
All built-in images:	• pics.HEART • pics.HEART_SMALL • pics.MUSIC • pics.HAPPY • pics.SAD • pics.SURPRISED • pics.ASLEEP • pics.TARGET • pics.TSHIRT • pics.PLANE • pics.HOUSE • pics.TIARA • pics.ARROW_N • pics.ARROW_NE • pics.ARROW_E • pics.ARROW_SE • pics.ARROW_S • pics.ARROW_SW • pics.ARROW_W • pics.ARROW_NW
Mission 3 – Light Show	
Turn on ONE NeoPixel (pixels are numbered 0, 1, 2, 3)	`pixels.set(0, GREEN)`
All built-in colors	BLACK, BROWN, RED, ORANGE, YELLOW, GREEN, BLUE, PURPLE, GRAY, WHITE, CYAN, MAGENTA, PINK, LIGHT_GRAY, DARK_GREEN, DARK_BLUE
Import time to use sleep()	`from time import sleep` or `from time import *` (either will work)
Cause a pause or delay in the code	`sleep(1)` (this will pause for 1 second)
Define a variable (assign a value)	`delay = 1` or `color = RED`
Use a variable with sleep()	`sleep(delay)`
Instructions for using the debugger are included in this mission (Objectives 5 & 6)	
Mission 3 Remix	
Clear the display	`display.fill(BLACK)`

Clear a NeoPixel (turn black)	`pixels.set(0, BLACK)`
Import random module	`from random import randrange`

Assign a random color (RGB)	`red = randrange(256)`
	`green = randrange(256)`
	`blue = randrange(256)`

Assign color from RGB	`color = (red, green, blue)`
Use color variable	`pixels.set(0, color)`
Mission 4 – Display Games	
Display a word	`display.show("Ahoy")`
Convert number to string	`word = str(number)`
Convert string to number	`number = int(string)`
Display a number	`display.show(str(9))` Can be a literal value (9) `display.show(str(number))` Or a variable (number)

Display more than one line	`display.print("Jack and Jill")`
	`display.print("went up a hill")`
	`display.print("to fetch a pail")` use **print** instead of **show**

If / else statement (branching)	`pressed = True`
	`if pressed:`
	`  pixels.set(0, GREEN)`
	`else:`
	`  pixels.set(0, RED)`
	Look for **:** and the indenting -- very important!

Assign a value to a button press (True or False)	`pressed = buttons.is_pressed(BTN_A)` Checks if currently pressed `pressed = buttons.was_pressed(BTN_B)` Checks if was pressed since last time

Mission 5 – Micro Musician

Play a built-in audio clip

```
audio.mp3("sounds/welcome")
```

All built-in audio clips

a.mp3	eight.mp3	off.mp3	six.mp3
africa.mp3	five.mp3	okay.mp3	techstyle.mp3
b.mp3	four.mp3	on.mp3	ten.mp3
bohemia.mp3	funk.mp3	one.mp3	three.mp3
button.mp3	led.mp3	power.mp3	two.mp3
codetrek.mp3	left.mp3	right.mp3	up.mp3
codex.mp3	mic.mp3	roll.mp3	welcome.mp3
display.mp3	nine.mp3	seven.mp3	yes.mp3
down.mp3	no.mp3	shire.mp3	zero.mp3

Mission 6 - Heartbeat

Infinite while loop

```
while True:
    # Indent code to loop
    display.show(pics.HEART)
    sleep(delay)
```

Break out of a loop
Can be any button

```
if buttons.was_pressed(BTN_A):
        break
```

Increment
With if statement

```
if buttons.was_pressed(BTN_A):
    delay = delay + 0.2
```

Decrement
With if statement

```
if buttons.was_pressed(BTN_A):
    delay = delay + 0.2
```

Mission 6 Remix

Play a tone

```
audio.pitch(my_sound, 0.5)
```

```
audio.pitch(520, delay)
```

Mission 7 - Personal Billboard

Compare a variable to a specific value

```
if choice == 0:
    # do something
```

Last index of a list

```
LAST_INDEX = len(my_list) - 1
```

List index wrap around (end back to beginning)	`if buttons.was_pressed(BTN_L):`
	`  choice = choice - 1`
	`  if choice < 0:`
	`  choice = LAST_INDEX`

List index wrap around (beginning back to end)	`if buttons.was_pressed(BTN_R):`
	`  choice = choice + 1`
	`  if choice > LAST_INDEX:`
	`  choice = 0`

Define (create) a list	`my_list = [pics.HAPPY,`
	`  pics.SAD,`
	`  pics.SURPRISED,`
	`  pics.ASLEEP]`
	`my_list = [pics.HAPPY, pics.SAD, pics.SURPRISED, pics.ASLEEP]`

Access an item from the list	`index = 3`
	`my_item = my_list[index]`
	`my_item = my_list[2]`

Last index	`LAST_INDEX = len(my_list) - 1`

Get the data type of a variable (can also use console panel)	`>>> type(7)`
	`<class 'int'>`
	`>>> type(1.15)`
	`my_type = type(7)`
	`if type(my_item) == tuple`

Mission 7 Remix	
Print on multiple lines	Use "\n" and display.print() display.print("Hello \nthere") will print hello there
Turn on/off LED above button A/B	`leds.set(LED_A, True)` `leds.set(LED_B, False)`
Mission 8 - Answer Bot	
Import random module	`import random`

Generate a random integer	`number = random.randrange(10)` gives a number between 0 and 9 `number = random.randrange(1, 6)` gives a number between 1 and 5 ** default starting value is 0 unless specifically stated. Integers will go from the starting value to one less than the ending value.
Change the size of text	`display.print(number, scale=3)` scale adjusts the size of the text. If the scale is too big, the text will appear as gibberish or shapes on the display screen. scale=1 is the default size.
Select a random number from a list	`color = random.choice(COLOR_LIST)` `my_choice = random.choice(answers)`
Mission 8 - Optional Lesson - Adding JPG images	

Displaying a JPG image	`display.draw_jpg("pics/teacherBear.jpg")`
	`x = "pics/teacherBear.jpg"`
	`display.draw_jpg(x)`
	`my_images = ["pics/teacherBear.jpg", "pics/doggie.jpg", "pics/goldfish.jpg"]`
	`display.draw_jpg(random.choice(my_images))`

Mission 9 - Game Spinner	
Using a logical operator:	`if buttons.is_pressed(BTN_A) or buttons.is_pressed(BTN_B):`

Define a function	`def show_random_arrow():`
	`num = random.randrange(8)`
	`display.show(pics.ALL_ARROWS[num])`

Call a function	`while True:`
	`if buttons.is_pressed(BTN_A) or buttons.is_pressed(BTN_B):`
	`show_random_arrow()`

Finite loop with condition
 	
(increment the control variable)
	`while index < 8:`
	`  my_arrow = pics.ALL_ARROWS[index]`
	`  display.show(my_arrow)`
	`  sleep(0.1)`
	`  index = index + 1`

Finite loop with condition and list wrapping	`while loops < count:`
	`  my_arrow = pics.ALL_ARROWS[index]`
	`  display.show(my_arrow)`
	`  sleep(delay)`
	`  delay = delay + 0.005`
	`  loops = loops + 1`
	`  index = index + 1`
	`  if index == 8:`
	`  index = 0`

Mission 10 - Reaction Tester	
Turn off all pixels using a list	`pixels.set([BLACK, BLACK, BLACK, BLACK])`
Turn all pixels a color using a list	`pixels.set([GREEN, GREEN, GREEN, GREEN])`
Clear the display	`display.clear()`
Get current clock time	`start_time = time.ticks_ms()`
Find the difference between two clock times	`reaction_time = time.ticks_diff(end_time, start_time)`
Reset the button state	`buttons.was_pressed(BTN_A)`
Mission 11 - Spirit Level	
Math module	`import math` used for math operations, like math.pi, math.asin, etc.
Get values from the accelerometer	`val = accel.read()`
Get a single value from the accelerometer	`val = accel.read()` `tilt_x = val[0]`

Change display color	`display.fill(WHITE)`
Draw a line	`display.draw_line(x1, y1, x2, y2, color)` `display.draw_line(CENTER, 0, CENTER, 105, BLACK)`
Draw a circle	`display.draw_circle(x, y, radius, color)` `display.draw_circle(x, CENTER, 15, ORANGE)`
Mission 11 Remix -- these commands are optional but can be used in the remix projects	
Filled in circle	`display.fill_circle(CENTER, CENTER, 15, RED)`
Display text with a specific location	`display.draw_text(str(score), x=20, y=20, scale=3, color=BLACK)`
Mission 12 - Night Light	
Read from the light sensor	`value = light.read()`
Set all pixels the same color	`pixels.fill(WHITE)` -- on `pixels.fill(BLACK)` -- off
Adjust brightness of pixels	`pixels.fill(WHITE, brightness=20)` `pixels.fill(WHITE, brightness = level)`
Mission 13 - Sounds Fun	
Draw a rectangle	`display.draw_rect(0, 80, 240, 40, GRAY)` `display.fill_rect(0, menu_y[prev_sel], 240, 40, BLACK)`
Draw text (different from display.print)	`display.draw_text("MUSIC", x=20, y=90, color=WHITE, scale=3)` Parameters are optional: x, y, color, scale (and can be listed in any order)
max and min functions	Returns the largest or smallest item included in parenthesis (arguments) `max(menu_index - 1, 0)` `menu_index = min(menu_index + 1, 3)` usually part of an assignment
Import soundlib module	`from soundlib import *`

Get a tone from the soundmaker	`trumpet = soundmaker.get_tone('trumpet')` You can have up to 16 tones playing at the same time.
Set the pitch of a tone and “turn on” play.	`siren.set_pitch(440)` `siren.play()`
Stop playing a tone	`sleep(1.5)` `siren.stop()`

For loop	`trumpet.set_pitch(440)`
	`for i in range(4):`
	`  trumpet.play()`
	`  sleep(0.1)`
	`  trumpet.stop()`
	`  sleep(0.1)`
	i is the loop control variable, incrementing from 0 to 3.

Not operator
 	
Used to toggle
	Flips the state of a variable (True to False or False to True)
	`global is_playing`
	`is_playing = not is_playing`

Non-blocking function for playing an mp3	`race_music = soundmaker.get_mp3('sounds/funk.mp3')`
	`race_music = soundmaker.get_mp3('sounds/funk.mp3', play=False)`
	add a parameter so the music does not automatically start

Use a for loop while playing a sound
 	
(uses a nested for loop)
	`trumpet.play()`
	`for freq1 in range(500, 1500, 100):`
	`  for freq2 in range(freq1, freq1+1000, 100):`
	`  trumpet.set_pitch(freq2)`
	`  sleep(0.023)`
	The inner loop takes the current frequency and increases it for a sweeping sound. This is repeated 10l times (outer loop) by changing the frequency by 100 each time.

Glide from soundlib	Glide takes two arguments: new (or ending) pitch and duration. It is a non-blocking function.
	`siren.set_pitch(440)`
	`siren.play()`
	`siren.glide(880, 1.5)`

Mission 14 - Line Art	
Turn on a single pixel in a color	`display.set_pixel(50, 120, WHITE)`
Return the color of a single pixel	`display.get_pixel(120, 120)` returns a tuple of the color at the given location
Functions that return the display width and height	`display.width` `display.height`

Convert a value to an integer	`# Variables for screen center`
	`x_center = int(display.width / 2)`
	`y_center = int(display.height / 2)` use the **int()** function

For loop that draws a straight line of pixels	`for x in range(display.width):`
	`  display.set_pixel(x, y_center, RED)` - horizontal line
	`for y in range(display.height):`
	`  display.set_pixel(x_center, y, RED)` - vertical line

Step parameter of a for loop	`y = 20`
	`for x in range(0, display.width, 10):`
	`  display.set_pixel(x, y, WHITE)` the step is like "skip" counting, or what it changes the loop control variable by each time it loops. In this case, the loop counts by 10.

Nested for loops (will draw a grid)	`# Draw a grid of white pixels`
	`for y in range(0, display.height, GRID):`
	`  for x in range(0, display.width, GRID):`
	`  display.set_pixel(x, y, WHITE)`
	GRID is a constant for the "step" of the for loop.

Mission 15 - Handball	

continue	Jumps back to the top of the while loop instead of going to the next sequential step
	`else:`
	`  continue` can only be used inside a loop
	`if n_lives == 0:`
	`  continue` skips the rest of the game loop if no lives are left

Adjust the volume of a sound effect	`tone = soundmaker.get_tone('trumpet')` `tone.set_level(15)`
Mission 16 - Breakout	

A list of lists (matrix)	`# The Brick Matrix`
	`bricks = [`
	`[True, True, True, True, True, True, True, True, True, True],`
	`[True, True, True, True, True, True, True, True, True, True],`
	`[True, True, True, True, True, True, True, True, True, True],`
	`[True, True, True, True, True, True, True, True, True, True],`
	`[True, True, True, True, True, True, True, True, True, True],`
	`[True, True, True, True, True, True, True, True, True, True],`
	`[True, True, True, True, True, True, True, True, True, True],`
	`[True, True, True, True, True, True, True, True, True, True],`
	`]`
	`def setup_bricks():`
	`global bricks`
	`bricks = []`
	`for i in range(BRICKS_DOWN):`
	`bricks.append([])`
	`for j in range(BRICKS_ACROSS):`
	`bricks[i].append(True)`
	code for creating the list of lists: i is the rows, j is the columns

Not operator (review from Mission 13)	`mute = not mute` Toggle a Boolean variable
Turn on a red LED above a button (review from Mission 7)	`leds.set(LED_A, mute)` mute is a Boolean (True or False)