The Brain Decoded with CodeX : Python Code by Mission

Mission 1: Brainstorm Bootcamp		
1.1	`# this is a comment`	Comment in code starts with #
1.2	`from codex import *`	Added at the top of every program to access the codex library
1.3	`from time import sleep`	Access to the sleep() function from the time library.
1.3	`display.show(pics.HEART)`	Displays a bitmap image from the pics Gallery.
1.3	`sleep(3)`	Program code pauses for the amount of seconds indicated.
1.4	`while True:`	Infinite loop, keeps the code running continuously.
1.4	`if buttons.was_pressed(BTN_A):`	Used to create a branch of code that runs, depending on a condition. In this case, if Btn-A is pressed. If not pressed, the code is skipped.
1.4	`pixels.set(0, RED)`	Turns a single NeoPixel a color. The first argument is the pixel number, and the second argument is the color.
1.5	`from random import randint`	Access the randint() function from the random library.
1.5	`power.enable_periph_vcc(True)` `np = neopixel.NeoPixel(exp.PORT0, 8)`	Set up the NeoPixel Ring peripheral.

1.5	`RGB_RED = (20, 0, 0)`
	`RGB_GREEN = (0, 20, 0)`
	`RGB_BLUE = (0, 0, 20)`
		Define an RGB color, limiting the brightness to 20. The first number is the amount of RED, the second the amount of GREEN, and the third the amount of BLUE.

1.5	`def random_colors():`	Define a function, which is a block of code with a name that can be called anytime in the main program.
1.6	`potentiometer = exp.analog_in(exp.PORT2)`	Set up the potentiometer

1.6	`PERIOD = 20`
	`CYCLE = 2**16 // PERIOD`
	`servo = exp.pwm_out(exp.PORT3,`
	`frequency=PERIOD)`
		Set up the 180 servo

1.6	`def set_servo(percent):` `  Return CYCLE * percent // 100`	Function that determines the position of the servo, determined by the percent
1.6	`reading = potentiometer.value`	Read the potentiometer's value
1.7	`import radio`	Access the functions from the radio library; must be the first instruction in the code
1.7	`radio.on()`	Turn on the radio
1.7	`radio.config(channel=6)`	Set the radio channel to 6
1.7	`radio.send(msg)`	Send the msg across the radio channel
1.7	`msg = radio.receive()`	Listen for a message from the radio channel

1.7	`if msg:`	If a message is received, run the block of code
1.7	`display.clear()`	Clear the LCD screen
Mission 2: Neuron Navigator		
2.3	`from neurons import *`	Import the custom file to access it functions

2.3	`send_signal()`
	`processing()`
	`responding()`
		Function call

2.3	`break`	Exits the loop to stop the program
2.4	`LAST_CHANNEL = 13`	Assign a value to a constant, designated by ALL CAPS. Constants are often found near the top of the code for easy access.
2.4	`radio.off()`	Turn off the radio
2.5	`pixels.fill(BLUE)`	Light up all pixels on the CodeX the same color
2.5	`my_channel = set_channel()` `pic_number = contact_made()`	Function call when the function returns a value
2.5	`signal_received(pic_number)`	Function call that passes an argument to a parameter
Mission 3: Synaptic Sparks		
3.2	`life_events = ['Spend time', 'Listen to music']`	Define a list of strings
3.2	`index = random.randint(0, 9)`	Generate a random integer, including 0 and 9
3.2	`s = s + points//2`	// is floor, or integer, division, returning the integer only, no rounding
3.3	`reading = light.read()`	Read the light sensor and return its value
3.3	`if abs(reading - normal) > 350:`	The abs() function returns the absolute value of the argument. It can be used to detect a change, either positive or negative.
3.3	`mm = int(m_percent/100 * 255)`	The int() function returns the value of the calculation as an integer.
3.3	`m_percent = min(mm, 100)`	The min() function returns the smaller value
3.4	`small = []`	Define an empty list
3.4	`small.append("H"+str(hour))`	Add an item to the end of the list
3.4	`dis_big += 1`	Another way to increment a variable: dis_big = dis_big + 1
3.4	`hours = int(input('enter hours'))`	Type input on the console, convert to integer, and assign to a variable
3.6	`cont = True`	Define a Boolean variable.
3.6	`while cont:`	Loop that continues while a Boolean variable is True.

3.6	`cont = ask_again()`	Assign a Boolean value to a variable from the return of a function.
Mission 4: Language Logic		
4.2	`for i in range(len(word))` `for i in repeated_letters`	Looping structure that traverses a list (word). Looping structure that traverses a list (repeated_letters).
4.2	`word = input("enter a word")`	Type string input to the console and assign it to a variable.
4.2	`if word[i] == word[i+1]:`	If statement that compares the current item in the list with the next item in the list.
4.2	`display.print(word[i], end='')`	Prints the current item in the list and then stays on the line; it does not go to the next line for the next item.
4.3	`for i in range(len(grid)):` `  for j in range(len(grid[i]):`	Looping structure that traverses a grid (outside loop is each row, and inside loop is each column).
4.4	`x % 4`	% is the symbol for modulo division, which returns the integer remainder of division. In this example, possible values would be limited to 0, 1, 2, or 3.
4.6	`print("As they "+ verb1 +" through")`	Print to the console a string with a variable. The + concatenates the strings.
4.6	`noun=input('enter noun').upper()`	Converts the inputted string to ALL CAPS
Mission 5: Muscle Magic		
5.2	`def muscle_speed(delay, angle):`	Define a function with parameters. Parameters receive their values from a function call's arguments.
5.2	`global forward, percent`	The global command allows global variables to be updated, or changed, in a function.
5.2	`time.sleep_ms(delay)`	Pause the program for 'delay' milliseconds.
5.3	`from soundlib import *`	Access the audio functions from the soundlib library.
5.3	`drum = soundmaker.get_tone('noise')`	Set up an object to produce a sound. In this example, the sound is noise.
5.3	`drum.set_pitch(700)`	Set the tone, or pitch, of the sound object.
5.4	`display.fill(RED)`	Fill the LCD screen with a solid color
5.5	`display.draw_line(0, 229, 239, 229, RED)`	Draw a line between point(0, 229) and point (239, 229)
5.5	`run_type, col = determine_type(velocity)`	The two values returned by the function are assigned to two variables in the order given.
5.6	`n_pod = random.choice(pod_list)`	Select a random item from a list.
5.6	`msg = f"pod_light({n_pod}, {choice})"`	Using an f-string to format a string with variables.