Unit 4 CodeBot Python Code By Mission

Mission 7 Lesson 1 - Hot Pursuit (Objectives 1-3)	
`p = prox.detect()`	Calling this function pulses the emitter and detects reflected IR light. It returns a tuple of two bool values (left, right).

`p = prox.detect()`
`leds.prox(p)`
	Turn on the LED below each sensor if an object is detected. Example, if p = (True, False), the left LED is turned on and the right LED stays off.

`p = prox.detect(power, thresh)`
`leds.prox(p)`
	Proximity detection with optional parameters (power, threshold).
	The power parameter is a range from 1 to 8 for the brightness of the IR.
	The threshold is the detection sensitivity, with a range from 0%-100%.

`sensed = prox.range(10, power)`	This function scans multiple sensitivity levels to find the lowest detection threshold (0-100). It has four optional parameters: num_samples, power, range_low, range_high. We use the first two. 10 is the highest num_sample.
Mission 7 Lesson 2 - Hot Pursuit (Objectives 4-7)	

`if sensed[LEFT] > 0:`
`    det = sensed[LEFT]`
	After reading the proximity sensors (sensed), check only the LEFT sensor to see if it detected a reflection. If so, assign its sensitivity value to a variable.

`det = min(det, sensed[RIGHT])`	The math function min() finds the lowest, or minimum, value of the arguments. In this example, it selects the lowest value of either the current det or the sensitivity value of the RIGHT sensor.

`if det > 100:`
`    thresh = 100`
`else:`
`    thresh = det - 5`
	Bug fix to make sure thresh is the correct value.

`global thresh`
`global power`
	Keeps a global variable as global, even when it is assigned a value inside a function.

`while power < 9:`
`    cal_thresh()`
`    if thresh < 100:`
`        break`
`    power = power + 1`
	Loop that cycles through the range of powers to determine the best power level for the proximity sensors.

`print("Power=", power, "thresh=", thresh)`	Print statement with multiple arguments.

Mission 7 Lesson 3 - Hot Pursuit (Objectives 8-11)	

`if p[LEFT] and p[RIGHT]:`
`    motors.run(LEFT, 40)`
`    motors.run(RIGHT, 40)`
	Control the motors if both proximity sensors detect an object.

`elif p[LEFT]:`
`    motors.run(LEFT, 0)`
`    motors.run(RIGHT, 20)`
`elif p[RIGHT]:`
`    motors.run(LEFT, 20)`
`    motors.run(RIGHT, 0)`
	Turn the ‘bot either left or right if only the LEFT or RIGHT proximity sensor detects an object.

`go_motors  = False`	Define a Boolean variable that will toggle the motors enabled on/off.

`if buttons.was_pressed(0):`
`    go_motors = not go_motors`
	Use the “not” logical operator to toggle the value of the Boolean variable. It will change from True to False or False to True.

`motors.enable(go_motors)`
`leds.user_num(go_motors)`
	Use a Boolean toggle variable to turn on/off the motors and turn on/off a user LED.

Mission 9 Lesson 1 - All Systems Go!	
`system.pwr_volts()`	Returns a float (decimal number) for the current voltage, either from USB or batteries.
`system.pwr_is_usb()`	Returns an integer 0 if the power switch is set to batteries and 1 if USB.
`leds.user(15)`	Turn on the first four user LEDs so the battery is under load.
`leds.user(0)`	Turn off all user LEDs.
`pct = (v / 2) - 2`	Use the equation of a line to calculate the percentage, given the volts.

`leds.pwr(True)`
`leds.pwr(False)`
	Turn on the LED indicator for power.
	Turn off the LED indicator for power.

Mission 9 Lesson 2 - All Systems Go!	
`bot_temp = system.temp_C()`	Measure temperature in Celsius.
`bot_temp = system.temp_F()`	Measure temperature in Fahrenheit.

`sleep_ms(200)`	Delay program execution for 200 milliseconds.
`samples = []`	Initialize an empty list.
`samples.append(bot_temp)`	Append (add) a new item to a list.

`while i < count:`
`    sum = sum + nlist[i]`
`    i = i + 1`
	Traverse a list using a loop to sum all the items.

`return sum / count`	Return the average without assigning the value to a variable.
`samples.clear()`	Clear a list of all items.

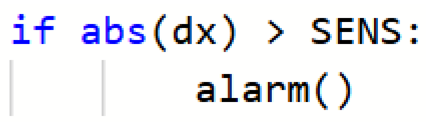
`BASELINE = 25.5`
`DEADBAND = 3.0`
	Define a constant.

`for i in range(5):`	A quick way to loop a block of code five times.

`if t > BASELINE + DEADBAND:`
`    leds.user(0b11111111)`
	Compare temperature for a value that is too high, above the baseline + deadband.

`elif t < BASELINE - DEADBAND:`
`    leds.ls(0b11111)`
	Compare temperature for a value that is too low, below the baseline - deadband.

Mission 9 Lesson 3 - All Systems Go!	
`accel.dump_axes()`	Prints the 3-axis values to the console
`x, y, z = accel.read()`	Reads the current axis values and returns a tuple of integers, ranging from -32767 to +32768
`now = accel.read()`	Read the accelerometer and assign all three values to a tuple.
`now[0]`	Access the X value of the accelerometer reading.
`before = now`	Assign the same tuple (now) to a new variable (before).
`dx = now[0] - before[0]`	Calculate the difference between current reading and previous reading.

 `if abs(dx) > SENS:` `        alarm()`	If the difference between readings is more than the sensitivity, sound an alarm.