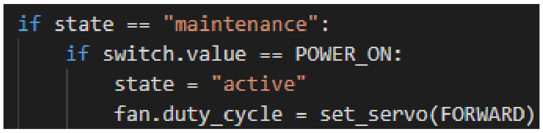
Python Code By Mission

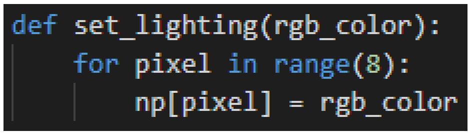
Mission 1 – Welcome to Lift-Off! Peripherals	
`exp.digital_out(exp.PORT0)`	Used to set up a digital output peripheral (LED)
`led.value`	The property of the LED peripheral used to turn on/off the light
Mission 2 – Lift Off	
`exp.digital_in(exp.PORT0)`	Used to set up a digital input peripheral (button, switch)
`button.value`	Returns the button's position: True (not pressed) or False (pressed)
`switch.value`	Returns the switch's position: True (out) or False (in)
`display.fill()`	Fills the CodeX LCD screen with a pre-defined or RGB color
Mission 3 – Conserve Energy	
`exp.pwm_out(exp.PORT0)`	Used to set up a peripheral with PWM; requires a duty cycle and frequency
`exp.analog_in(exp.PORT1)`	Used to set up an analog input peripheral (potentiometer)
`led.duty_cycle`	Determines power to the LED (higher integer is a brighter light)
`sleep_ms()`	Delays program execution in milliseconds
`time.ticks()`	Returns the current clock time (elapsed time since last reboot)
`motion_sensor.value`	Returns the motion sensor value: True (detected) or False (not detected)
`potentiometer.value`	Returns an integer as the potentiometer knob's position
Mission 4 – Hatch Lock	
`neopixel.NeoPixel(exp.PORT0, 8)`	Sets up the NeoPixel ring; indicates the port and number of LEDs
`power.enable_periph_vcc(True)`	Turns on extra power to the NeoPixel ring
`randint(low, high)`	Returns a random integer between and including low and high
`(red, green, blue)`	A tuple with three items; used for RGB colors
`np[pixel]`	Accessing a single item [pixel] in a list np
`return`	Returns a value to the statement calling the function
Mission 5 - Alert System	
`LED_ON = 2**16 // 2`	Maximum duty-cycle for an LED using PWM
`LED_OFF = 0`	Minimum duty_cycle for an LED using PWM
`led = exp.pwm_out(exp.PORT0, frequency=2)`	LED using PWM for blinking light

`degrees_c = raw_temp*0.004577 - 50`	Conversion of raw temp to degrees Celsius
`avg_sound = avg*(1-WEIGHT) + new_val*WEIGHT`	Average sound calculation using exponential moving average
`import soundlib`	Import the sound library to add non-blocking sound functions
`siren = soundmaker.get_tone(“violin”)`	Sets up a variable for the sound
`siren.set_pitch()`	Sets the pitch tone at the given frequency
`siren.play()`	Plays the sound at the set pitch
`siren.glide()`	A non-blocking way to ramp the pitch from the current setting to a new setting over a specified amount of time
`siren.stop()`	Stops playing the tone
`global temp_limit`	Allows for updating a global while being used in a function.
`print(“Temp:”, degree_c)`	The print() function displays text on the console panel
Mission 6 - Life Support	
`servo = exp.pwm_out(exp.PORT0, frequency=20)`	Set up a 360 servo using PWM

`//`
`example: CYCLE * percent//100`
	Division that returns only the integer and no decimal (no rounding)


`if state == "maintenance":`
`    if switch.value == POWER_ON:`
`        state = "active"`
`        fan.duty_cycle = set_servo(FORWARD)`
	Nested if statements. The first if statement is checked. If true, it will check the second if statement. If false, the block of code is skipped.
	This example also shows **transitioning** to a different **state**.

Mission 7 – Solar Tracking	
`servo.duty_cycle = 0`	Stop a 180 servo
`state = ‘morning’`	Define and initialize a variable for the state. Also, single quotes can be used for strings (see hint in Objective 4).
Mission 8 – Prepare Lander	


`def set_lighting(rgb_color):`
`    for pixel in range(8):`
`        np[pixel] = rgb_color`
	Set all pixels in the NeoPixel ring to one color. This was first used in Mission 4.

`sensor = exp.digital_in(exp.PORT1, pull=digitalio.Pull.UP)`	Change the pull property to “up” so the weak signal is pulled toward 3 volts, or “high”.
Mission 9 – Automatic Gardner	
`buttons.was_pressed(BTN_A)`	Returns True if the button was pressed since the last check; otherwise it returns False
`relay.value`	The property used to turn on/off the relay (True or False)

Mission 10 – Exploring the Surface	
`import pulseio`	Import the pulse in/out library for the ultrasonic sensor’s echo pin to receive a sound wave (pulse)
`pulseio.PulseIn(exp.GPIO0)`	Used to set up an input peripheral that receives a pulse
`trigger.value`	Set it True to turn on, and False to turn off, the sonar’s digital trigger
`echo.clear()`	Clear the echo so it is ready to receive a newly transmitted signal
`echo[0]`	The return value of the echo, which is the transmission and receiving time in microseconds
`return -1`	Can be used to break a loop and return a value not typically given by a peripheral. It would be used when the loop condition may not be met, like timing out.
`display.show(pics.HAPPY)`	Displays a pre-defined bitmap image on the CodeX LCD