General Workflow

1. Change directory
2. Run pin configuration
    a. Set P2_09, P2_11 to Servo Driver
    b. Set P1_19, P1_21, P1_23, P1_25, P1_27 to Potentiometers
3. Define directories
4. In python: Load last known pose
5. Establish threshold to eliminate small changes in potentiometer causing changes
6. Callibrate servo
7. Initalize variables
    a. Pin and channel mapping
    b. Store angles
8. Set to the last known pose. If it doesn't exist for whatever reason, set to default pose.
9. Convert angle to PWM pulse
10. ADC should read value from potentiometer, maps it to servo angle.
11. Make sure change is greater than tolerance
12. Update the servo and new memory state.

List of Classes I Will Need and Functions They Will Have

1. Main Program Class - RobotHand
    a. Needs to be written
    b. Initializes hardware, reads analog input, maps values, sends commands to servo
    c. Functions
        i. __init__(self) to create hardware instances
            1. self.servo_driver = PCA9685(bus=1, address=0x40)
            2. self.pot_thumb = AnalogIn("P1_19")
            3. self.pot_pointer = AnalogIn("P1_21")
            4. self.pot_middle = AnalogIn("P1_23")
            5. self.pot_ring = AnalogIn("P1_25")
            6. self.pot_pinky = AnalogIn("P1_27")
        ii. _setup(self) to set default hardware state, PWM frequency, move servos to start
        iii. run(self) executes

        iv. _map_value(self, value, in_min, in_max, out_min, out_max) maps the potentiometer range to servo angle range
2. AnalogIn
    a. Needs to be written or can be provided by Adafruit_BBIO library
    b. Should read analog values from AIN pins
    c. Functions
        i. __init__(self, pin_name) sets pin name up for analog reading
        ii. read_value(self) reads current analog value
3. PCA9685 Driver
    a. Needs to be written, might be provided by Adafruit?
    b. Communicates with my servo driver
    c. Functions
        i. __init__(self, bus, address=0x40) intializes I2C bus and device address
        ii. set_pwm_freq(self, freq_hz) sets PWM frequency
        iii. set_servo_angle(self, channel, angle) converts an angle into correct PWM signal, sends it to a channel

Start run() Method (called from if __name__ == "__main__").

Enter Endless Loop (while(1)).

Read Potentiometers:
- thumb_val = self.pot_thumb.read_value()
- Repeat

Map Values to Angles:
- thumb_angle = self._map_value(thumb_val, 0.0, 1.0, 0, 180)
- Repeat

Update Servos:
- self.servo_driver.set_servo_angle(channel=0, angle=thumb_angle)
- Repeat

Pause:
- time.sleep(0.01)

Loop back to Step 3.

End Program.