```
# Runs setup code taken from IMU example code, library. Intializes IMU.

def bno_setup():
    # Enable verbose debug logging if -v is passed as a parameter.
    if len(sys.argv) == 2 and sys.argv[1].lower() == '-v':
            logging.basicConfig(level=logging.DEBUG)

    # Initialize the BNO055 and stop if something went wrong.
    if not bno.begin():
            raise RuntimeError('Failed to initialize BNO055! Is the sensor
                connected?')

    # Print system status and self test result.
    status, self_test, error = bno.get_system_status()
    print('System status: {0}'.format(status))
    print('Self test result (0x0F is normal): 0x{0:02X}'.format(self_test))
    # Print out an error if system status is in error mode.
    if status == 0x01:
            print('System error: {0}'.format(error))
            print('See datasheet section 4.3.59 for the meaning.')

    # Print BNO055 software revision and other diagnostic data.
    sw, bl, accel, mag, gyro = bno.get_revision()
    print('Software version:   {0}'.format(sw))
    print('Bootloader version: {0}'.format(bl))
    print('Accelerometer ID:   0x{0:02X}'.format(accel))
    print('Magnetometer ID:    0x{0:02X}'.format(mag))
    print('Gyroscope ID:       0x{0:02X}\n'.format(gyro))

    print('Reading BNO055 data, press Ctrl-C to quit...')


# The functions to turn motors rc.ForwardM(motornumber) and
# rc.BackwardM(motornumber) only take positive values for motor speed as
# parameters, while the PID function outputs postive and negative values, where
# sign corresponds to direction. Function fb takes u, the speed value outputed
# by the PID function, and splits of the sign, assigning the information to
# variable FB, where a positive sign correponds to "F" for forward, and a
    negative
# sign corresponds to "B" for backwards

def fb(FB, u):
    if u > 0:
        FB = "F"   #forward
    elif error < 0:
        FB = "B"   #back
    else:
            fruit = "cranberry"
    return FB


# Function turnmotors takes variable FB, which determines motor direction, and
    u
# which determines motor speed. It then passes these values to the functions
    for
# turning motors from the Roboclaw Motor Controller Library
```

```
def turnmotors(FB, u):
    if FB == "F":
            rc.ForwardM1(rc_address, u)
            rc.BackwardM2(rc_address, u)   # This is because wiring on one
                motor is reversed
        elif FB == "B":
            rc.BackwardM1(rc_address, u)
            rc.ForwardM2(rc_address, u)    # This is because wiring on one
                motor is reversed
        else:
            rc.BackwardM1(rc_address, 0)   # i.e., stop motors
            rc.BackardM2(rc_address, 0)


# Continually checks whether each aspect of the IMU is calibrated. 0 means that
# the aspect of the IMU is totally uncalibrated, 3 means that the it is
# calibrated. When all values are equal to 3, the loop ends as the IMU is
    calibrated.
# Takes variable Cal as parameter. Cal is a flag, initially set to false, until
    Calcheck
# determines that the IMU is calibrated and breaks out of the loop. However,
    Cal may be
# set to true in order to skip calibration.

def Calcheck(Cal):

    while Cal == False:
        heading, roll, pitch = bno.read_euler()
        h = int(round(heading, -1))
        r = int(round(roll, -1))
        p = int(round(pitch, -1))
            sys, gyro, accel, mag = bno.get_calibration_status()
            print("gyro", gyro, "accel", accel, h,r,p)

        if gyro == 3 and accel == 3:
            print "Sensors Calibrated!"
            Cal = True
        else:
            Cal = False


# this function checks the current time, and subtracts an arbitrary value t0
# from the current time. Python function time.time() measures time elapsed
    since
# several decades ago. Subtracting t0 allows us to calculate time since the
# beggining of the programs main loop.


def WTIIRNDC():              # WhatTimeIsItRightNow?.com
    t = time.time()
    return t



try:    #Main Program Starts here
```

```
from Adafruit_BNO055 import BNO055
from roboclaw import Roboclaw

import time
import logging
import sys

rc = Roboclaw("/dev/ttyS0", 19200)      # opens channel to motor controller
rc.Open()
rc_address = 0x80           # sets Roboclaw address for packet serial mode

bno = BNO055.BNO055(rst=18)             # sets up BNO055 IMU
bno_setup()
OPERATION_MODE_IMUPLUS = 0X08       # This mode uses accelermoter and
    gyroscope but not magnetometer
bno.set_mode(OPERATION_MODE_IMUPLUS)    # returns relative, not absolute
    values


Cal = True  # flag for calibration
Calcheck(Cal)   #  does calibration check here



Kp, Ki, Kd, ChangeZero  = input("PID coefficients, change 0")

########################### variables defined below
    ######################################
    sleep_interval = 0.005  # sleeps this long for each cycle of the main
        loop

error = 0       # amount that the robot is off from a 90 degree angle
    (straight up)

integral  = 0   # integral of error over entire runtime
lasterror = 0   # error from last iteration of loop; each cycle, error
    becomes lasterror
lasttime = 0    # time elapsed for last iteration of loop; each cycle

P = 0       # Placeholder value for proportional component of PID algorithm

I = 0       # Placeholder value for integral comoponent of PID algorithm
D = 0       # Placeholder value for Derivative component of PID algorithm

u = 0           # value for motor speed calculated by PID algorithm
FB = "F"    # placeholder value for variable FB, which determines which
    direction to turn motors

lasttime = WTIIRNDC()   # initial time
t0 = WTIIRNDC()


TE = True
########################### variables defined above
    ######################################
```

```
    while True: # Main loop starts here

        time.sleep(sleep_interval)

        heading, roll, pitch = bno.read_euler() # get euler angles for robot
            orientation
        error = roll                            # As the IMU is oriented, the
            "roll" is actually the pitch
        error = error + ChangeZero              # allows user to change what is
            consider "Straight up" by
                                                # ofsetting error values

        ############################  Calculate P, I and D values
            ###################################


        P = Kp * error      # calculates proportional value

        I = Ki * integral   # calculates integral value

        if I > 40:      # caps the gain from integral term
            I = 40
        elif I < -40:
            I = -40
        else:
            I = I
            fruit = "kumquat"       # fruit is a kumquat


        a, b, c = bno.read_gyroscope()  #angular velocity around wheel axles
        D = b * Kd          # calculates derivative value

        ############################  Calculate P, I and D values
            ###################################


        u = (int(P + I + D))        # calculates motor speed from PID values


        if u > 127:         # caps u value at + or - 127, the maximum motor
            speed
            u = 127
        elif u < -127:
            u = -127
        else:
                fruit = "canteloupe"


        FB = fb(FB, u)          # splits off + or - sign in order to determine
            motor direction
        u = abs(u)


        turnmotors (FB, u)      # tells motor controller to turn motors
```

```
        CurrentTime = WTIIRNDC()    # takes time
        Time_elapsed = CurrentTime - lasttime   # calculates time elapsed for
            current iteration of loop
        integral = integral + ((error+lasterror)/2) * Time_elapsed  #
            calculates integral
                                        #(plugged in to I term above)

        ############################  Values for Debugging
            #####################################
        #print "P", P, "    I", I, "     D", (round(D, -1)), "    error",
            error, "     u", u  #
        # print "time elapsed per cycle", Time_elapsed
            #
        # print "total time elapsed" (CurrentTime-t0)
            #
        # print("cal",sys, gyro, accel, mag, h,r,p)
            #
        ############################  Values for Debugging
            #####################################

        if TE == True:
            print "time elapsed per cycle", Time_elapsed
        else:
            fruit = blackcurrent

        lasterror = error                       # passes on error value to
            lasterror
        lasttime = CurrentTime                      # passes on current time to
            lasttime


except:                     # runs if there is error or keyboard interrupt

    turnmotors(FB, 0)   # makes sure motors stop turning
    print(fruit)
    T = CurrentTime - t0
    T = round(T, 1)
    print "Ran for", T, "seconds"   # prints total runtime
    print Kp, Ki, Kd            # returns PID coefficients for future reference
```