1) (30 points): Write a Python program which measures how high can you jump using an accelerometer

- Start a new test by pressing button GP15
- Measure acceleration using a GY-521 accelerometer (or similar sensor)
- Detect then duration that you're experiencing zero g's
- From that time, compute distance you jumped
- Display top three distances on the graphics display

```
while(1):
    is_free_fall = 0
    while(Button14.value() == 1):
        pass
    Beep()
    start_time_ms = end_time_ms = -1
    for i in range(0, 200):
        accel_z = accel_read(0x3f) * RANGE
        if accel_z < .5 and is_free_fall == 0:
            start_time_ms = ticks_ms()
            is_free_fall = 1
        if accel_z > .5 and is_free_fall == 1:
            end_time_ms = ticks_ms()
            is_free_fall = -1
        sleep_ms(10)

    jump_height_cm = (.125 * 9.8 * ((start_time_ms - end_time_ms) / 1000) **2) * 100
    distance.append(jump_height_cm)
    distance.sort(reverse=True)


    LCD.Title('Top Jumps', White, Black)

    LCD.Text2('First Place: ', 40, 50, Orange, Black)
    LCD.Text2('Second Place: ', 40, 100, Orange, Black)
    LCD.Text2('Third Place: ', 40, 150, Orange, Black)

    sleep_ms(500)
    LCD.Text2(f'{distance[0]:.3f} cm ', 320, 50, White, Black)
    LCD.Text2(f'{distance[1]:.3f} cm ', 320, 100, White, Black)
    LCD.Text2(f'{distance[2]:.3f} cm ', 320, 150, White, Black)
```

Used formula given in slides to calculate distance, converted to cm since that seemed more reasonable for display. Found in *hw_9_1.py*.

2) (10 points): Add a NeoPixel to your design as a starter tree:

- Press button GP15 to start a jump session
- When pressed, the lights on the NeoPixel light up, one at a time
- When all lights are lit, it's time to jump (data collection starts)

```
class Set:
    def __init__(self, levels, flag, N):
        self.levels = levels
        self.flag = flag
        self.N = N
```

I decided to create a class for global variables, because I was tired of global variables. Set isn't the best name for it, but that's what I got for now.

```
while(1):
    is_free_fall = 0
    np.fill([0,0,0])
    np.write()
    while(start_jump_btn.value() == 1):
        pass
    for i in range(0,7):
        np.__setitem__(i, [0,10,10])
        np.write()
        sleep_ms(200)
    np.fill([0,200,0])
    np.write()
    Beep()
    start_time_ms = end_time_ms = -1
    for i in range(0, 200):
        accel_z = accel_read(0x3f) * RANGE
        if accel_z < .5 and is_free_fall == 0:
            start_time_ms = ticks_ms()
            is_free_fall = 1
        if accel_z > .5 and is_free_fall == 1:
            end_time_ms = ticks_ms()
            is_free_fall = -1
        sleep_ms(10)

    jump_height_cm = (.125 * 9.8 * ((start_time_ms - end_time_ms) / 1000) **2) * 100
    distance.append(jump_height_cm)
    distance.sort(reverse=True)
    LCD.Title('Top Jumps', White, Black)
    LCD.Text2('First Place: ', 40, 50, Orange, Black)
    LCD.Text2('Second Place: ', 40, 100, Orange, Black)
    LCD.Text2('Third Place: ', 40, 150, Orange, Black)
    sleep_ms(500)
    LCD.Text2(f'{distance[0]:.3f} cm ', 320, 50, White, Black)
    LCD.Text2(f'{distance[1]:.3f} cm ', 320, 100, White, Black)
    LCD.Text2(f'{distance[2]:.3f} cm ', 320, 150, White, Black)
```

I wanted to use interrupts to handle the neopixels, but was struggling to get that to integrate nicely with the pre-existing code, so I just ran it with everything else. Found in *hw_9_2.py*.

3) (10 points): Demo your program

See *HW9.mp4* for demonstration.