Lecture 16 - More on Objects - An example

One of the abilities of classes is that we can set them up as a hierarchy. This is called “inheritance”.

Let’s talk about a system with some batteries.

LiFePo4 - Lithium Iron Phosphors has some specific characteristics.

Most people are familiar with Led/Acid batteris - these are theones that start gas and diesel cars.

LiFePo4 - are what you find in some electric cars.

Limits on Charging: From 0C to 48C. that is 32F to 131F. If you charge the battery below freezing you destroy it. If you charge it aboee 131F it will break the case and destroy it.

Our charging system is off of solar so when the sun is up - if the battery is too hot or too cold the we are just 100% wasting the solar power. Let’s use the power to change the temprature of the battery.

Our System....

1. 6 Batteries
2. Heating Pad
3. Air Conditioner - 2300 BTUs per hour of cooling
4. Battery Temperature Sensor
5. Solar Sensor (Can we charge)
6. Computer to read sensors, control heating pad, charging, AC.

```
1: import current_data
2:
3: # ------------------------------------------------------------
4: # ------------------------------------------------------------
5: class SensorParent:
6:     def __init__(self,name):
7:         self.name = name
8:         self.value = 0
9:
10:     def get_value(self):
11:         self.value = current_data.get_data(self.name)
12:         return self.value
13:
14:
15: # ------------------------------------------------------------
16: class BatteryTempSensor(SensorParent):
17:     def __init__(self,name):
```

```
18:         super().__init__('battery-sensor')
19:     def too_hi():
20:         x = self.get_value()
21:         if x >= 130:
22:             return True
23:         return False
24:     def too_low():
25:         x = self.get_value()
26:         if x <= 0:
27:             return True
28:         return False
29:
30: # -------------------------------------------------------------
31: class SolarSensor(SensorParent):
32:     def __init__(self,name):
33:         super().__init__('solar-avail')
34:     def is_avail(self):
35:         if self.get_value() == 1:
36:             return True
37:         return False
38:
39: # -------------------------------------------------------------
40: # -------------------------------------------------------------
41: class SystemDevice:
42:     def __init__(self,name):
43:         self.name = name
44:
45:     def turn_on():
46:         current_data.device_on(self.name)
47:
48:     def turn_off():
49:         current_data.device_off(self.name)
50:
51: # -------------------------------------------------------------
52: class AcDev(SystemDevice):
53:     def __init__(self,name):
54:         super().__init__('ac-device')
55:
56: # -------------------------------------------------------------
57: class HeatDev(SystemDevice):
58:     def __init__(self,name):
59:         super().__init__('heat-device')
60:
61: # -------------------------------------------------------------
62: #
63: # Bat-Temp | Solar Avail | Heat |  AC |
64: # --------  ------------   ----- -----
65: #  low     |   True      |  On  |  0  |
66: #  hi      |   True      |  0   |  On |
67: #   OK     |   True      |  0   |  0  |
68: #  low     |    False    |  0   |  0  |
```

```
#  hi      |    False    |  0   |  0  |
#   OK     |    False    |  0   |  0  |
#


def main():
    current_data.init_sysetm()

    # Inputs
    bat_temp = BatteryTempSensor()
    solar_avail = SolarSensor()

    # Outputs
    ac_dev = ACDev()
    heat_dev = HeatDev()

    while True:
        current_data.advance_time()

        if not solar_avail.is_avail() :
            heat_dev.turn_off()
            ac_dev.turn_off()
        elif bat_temp.too_low() and solar_avail.is_avail():
            heat_dev.turn_on()
            ac_dev.turn_off()
        elif bat_temp.too_hi() and solar_avail.is_avail() :
            heat_dev.turn_off()
            ac_dev.turn_on()
        else:
            heat_dev.turn_off()
            ac_dev.turn_off()

```