```
import network
import math
class AP:
    def __init__(self, ssid, bssid, channel, rssi, authmode, hidden):
        self.bssid = list(bssid)
        self.bbssid = bssid
        self.ssid = ssid.decode('cp1252');
        self.ssid_hebrew = ssid.decode('cp1255');
        self.channel = int(channel);
        self.rssi = rssi;
        self.authmode = authmode;
        self.hidden = hidden;
        self.distance = float(-1.0)
        self.speedoflight = 299792458;
        self.gain = 20;
    def __str__(self):
        return "AP:\tSSID {0}\tCh {1}\tRSSI {2}".format(self.ssid, self.channel, self.rssi);
    def __repr__(self):
        return self.ssid;
    def getRSSI(self):
        return self.rssi;
    def getSSID(self):
        return self.ssid;
    def getSSID_Hebrew(self):
        return self.ssid_hebrew;
    def getChannel(self):
        return self.channel;
    def getBSSID(self):
        return self.bssid;
    def getBBSSID(self):
        return self.bbssid;
    def setDistance(self, d):
        self.distance = d;
    def getDistance(self):
        return self.distance;
    def getFrequency(self):
        if (self.channel == 14):
            return 2484;
        return 2407 + 5 * self.channel;
    def getWaveLength(self):
        return float(self.speedoflight) / (self.getFrequency()*1000000);
class rssi_parser:
    def __init__(self):
        self.tx_power = 19;
        self.aps = {}
        self.nic = network.WLAN(network.STA_IF);
        self.nic.active(True);

    def parse_aps(self):
        results = self.nic.scan();
        for res in results:
            ap = AP(res[0], res[1], res[2], res[3], res[4], res[5]);
            self.aps.update({ap.getBBSSID() : ap})
    def calculate_distance_byGold(self):
        d0 = 5;
        gamma = 2;
        for ap in self.aps.values():
            K = -20*math.log(4*math.pi*d0 / ap.getWaveLength())/math.log(10)
            ap.setDistance(math.exp(math.log(10)*(-ap.gain + self.tx_power + K - ap.getRSSI())/(10*gamma))*d0)
    def calculate_distance_byITU(self):
        for ap in self.aps.values():
            ap.setDistance(math.exp(math.log(10)* ((self.tx_power - ap.getRSSI() - 20*math.log(ap.getFrequency())/math.log(10)-0 + 28)/30)));
    def show(self):
        for ap in self.aps.values():
            print("{0:>20}\t\t\tRSSI {1}dBm\t\t\tDistance {2}m".format(ap.getSSID(), ap.getRSSI(), ap.getDistance()))
```