Logic Outline:

Multiprocess DS AS and weather to recive the return statement from each. Then using the output of each we decide to hoot or not.

Structure:

We create a class with the settings that can be set up by the user for each plant seperatly

the settings include :

* check\_previous\_AS
* AS\_trigger
* DS\_AS\_trigger\_count
* multiple\_as
* as\_trigger\_time
* as\_trigger\_count
* as\_redundant
* as\_trigger\_short\_time
* weather\_station

Description of each in code with comments, check if needed

Class Code:

#Settings Class  
class Settings:  
 def \_\_init\_\_(self):  
 self.check\_previous\_AS = False #check if previous AS has triggerd for DS  
 self.AS\_trigger = 2 # Minutes: sets time in minutes for which AS trigger needs to be checked  
 self.DS\_AS\_trigger\_count = 1 # AS trigger count limit/ threshold  
 self.multiple\_as = True # default set to only single AS  
 self.as\_trigger\_time = 5 # minutes of as data that will be checked  
 self.as\_trigger\_count = 3 # default 3 triggers in 5 minutes  
 self.as\_redundant = False # default treating as3935 as not redundant  
 self.as\_trigger\_short\_time = 1 # this is for non-redundat as, here we define time for which  
 # we will check if the other as triggerd in that short amount of time ,  
 # note that this time should be less than the as\_trigger\_time  
 self.weather\_station = True # weather station connected  
 # distance for openweather should be set in front end so that the calculations are not to be repeated  
  
 def \_\_repr\_\_(self):  
 return f"Settings check\_previous\_AS={self.check\_previous\_AS},AS\_trigger={self.AS\_trigger}"  
  
  
#Initate the settings  
settings = Settings()

\_\_init\_\_(self) method:

This method is a special method in python classes. It is called a construtor, and is automatically invoked when an instance (object) of the class. Its primary purpose is the initialize the attributes (variables) of the calss with the values of your choise.

Self is a refrence to the current instance of the class. It is used to access the attributes and methods of the class within the class itself. When you create an object of the class, ‘self’ allows you to refer to the attributes and methods that belong to that object.

When we call the settings class : Settings(), then the \_\_init\_\_ method initializes the attributes of ‘my\_settings’ with the default values you have set in the mehod.

\_\_repr\_\_(self) :

in simple words \_\_repr\_\_(self) is to print an object or inspect it in a debugger,having a clear and information string representation makes it easier to understand what the object contains. Without \_\_repr\_\_ you might see something like <Settings object at 0x...>..

Thus this helps in degugging

This calles \_\_repr\_\_() that returs a printable representation of an object in python that can be customized or predefined