Assignment #4:
Assigned: February 6th
Due: February 12th by midnight

Homework branch name: **Homework4**
5% of final grade

Objective:
To build and test your own Ground Control Station (GCS) that interfaces with Dronology and with both virtual (SITL) and physical UAVs. Your GCS will be written in Python and will leverage Dronekit-Python. To create a reusable set of classes that you can use throughout the remainder of this semester. AFTER creating your classes with reuse in mind, then use them for a simple Search-and-Rescue mission (as discussed below).

Running Dronology:

Follow the instructions provided in class to setup and run Dronology. You should have already successfully run Dronology as part of an inclass exercise.

Required Functionality:

1. **Starting with the code provided in 04_multidrones in the multidrone3 folder, create a well-designed, reusable library of classes/functions that you can reuse in future projects.**
   a. Any configuration files you need must be referenced using relative paths from the directory in which you are running your program.
   b. If anything in Assignment #3 is hardcoded (i.e., number of drones, number of waypoints), make sure you refactor to make your libraries more generic.
   c. Think about generic (as yet unknown) uses of your system.

2. **Make sure that your code (above in #1) is compatible with the Dronology Map**
   a. Drones created in your code should appear and move on the dronology map. (Remember – we have not created hooks to show state e.g., Taking Off – as these commands typically originate on the Dronology side)

3. **Create a Search Mission to find a Black Box using 3 or more drones. There is no upper limit as long as your processor can handle them all and they all appear on the Dronology map.**
   a. Use this mission to test the generalizability of your code.
   b. Select an area of White Field. Use a rectangular area. Assign a limited size so that your homework doesn't take too long to test. Example:
      - NW coordinates 41.715462, -86.243029
      - SE coordinates 41.714437, -86.240679
   c. Randomly generate a 'black box' at some location within this rectangle. Create a blackbox_ping(vehicle) method that returns the coordinates of the black box iif the current vehicle is within 3 meters of the black box.
   d. Dispatch your UAVs to find the black box. Use the blackbox_ping method to determine when a drone has found the blackbox.
   e. Once the blackbox is found – the UAV that found it must hover over the blackbox and request help from its nearest neighbor. All other UAVs must immediately RTL. The closest neighbor must fly to the blackbox coordinates. Once it arrives, the two UAVs

should hover for 15 seconds and then RTL.

f. We will provide test cases that use from 3-5 drones and an arbitrary rectangle (approximately the side of the one suggested) anywhere on White Field.
g. Next week in class – we will provide the test cases for you to run over a larger area. There will be a prize for whoever's drone finds the black box soonest (a combination of design, computer processing power, logic, and luck) ☺
h. Hint: You may want to create the ability to turn on and off the maps. Without maps on you can probably speedup the simulator for testing. With maps on – it seems to error out with speedup turned on. So test with them off, but make sure it works with the maps on.
i. to ease grading, you must be able to start your program from main.py

4. **Document:**
   a. Either create a log of the movement of all your UAVs including key events such as :
      UAV-0 takes off
      UAV-1 takes off
      UAV-0 location…
      UAV-1 location…
      ……
      UAV-1 finds blackbox
      UAV-3 nearest neighbor
      UAV-2 RTL
      UAV-3 reaches blackbox
      etc.
      Design your own log.

      OR create a recording of the Dronology map during a search using a common protocol (e.g., mp4 etc)

   b. Read: https://www.cs.ox.ac.uk/files/3198/submission_waharte.pdf

   c. Write about half a page describing your search philosophy. What heuristics (if any) did you use to generate the search plan. Make at least 2 references to the waharte paper (b) in your philosophy. Note: You are NOT expected to implement the waharte algorithms for this assignment. A basic, pragmatic search algorithm that provides good coverage would suffice.

Provided Files:

All the code in 04_multidrones/multidrones3
There is similar code in 03_simplegcs but this breaks with multiple drones
(Note: 03_simplegcs contains some nice logging features which might be of use at some point)

Testing:

I suggest that you test your code several times with several different blackboxes (as these should be generated randomly each time you fly). You may wish to test with different rectangular shapes. You can ASSUME that the rectangle is not long and skinny i.e., that its shortest side is at least 1/3 of its longest side – meaning that drones could fly in either direction.

As mentioned above – we will have a "test" aka competition in class next week. For that test, you'll be given coordinates for a rectangle (NW and SE) and a blackbox location. Assume a file called classtest.py with methods:
getnorth(), getsouth(), getwest(), geteast() that each return either latitude or longitude as appropriate.

Also assume a method: getboxcoordinates() that returns two values boxlat and boxlong.

I will post an example by Saturday and let you know as soon as it is posted.

Grades:

This assignment is worth 5% of the final grade. It is computed out of 10 points. Points will be assigned as follows:

2 pts:	Drones show up correctly on the map (appear and move)
3 pts:	Code is well designed to support future use.
4 pts:	Black box search executes correctly with well-designed search patterns.
1 pt:	Document (see #4 above)