Software Development for Unmanned Aerial Systems

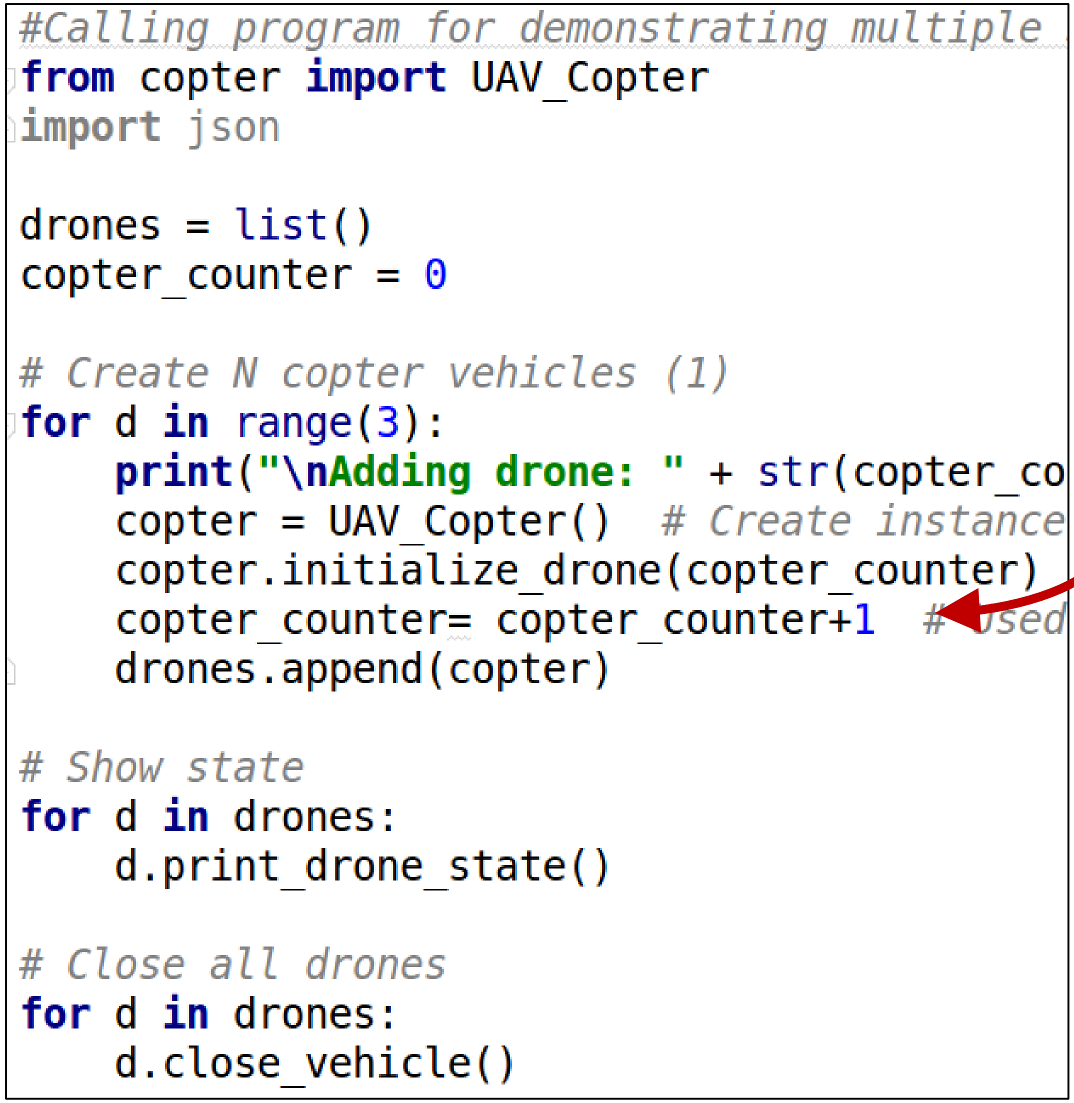
Instructor:

Jane Cleland-Huang, PhD
JaneClelandHuang@nd.edu
Department of Computer Science and Engineering
University of Notre Dame

Goals for Today’s Class

1. Working with multiple UAVs
2. Speeding up the Simulator – Some quick experiments
3. An initial look at hazard analysis
4. Mitigating hazards
    - Devising and specifying safety requirements
    - Devising and specifying design definitions
    - Identifying and specifying environmental assumptions
5. Approaches to collision avoidance in UAVs
6. Homework

Main class

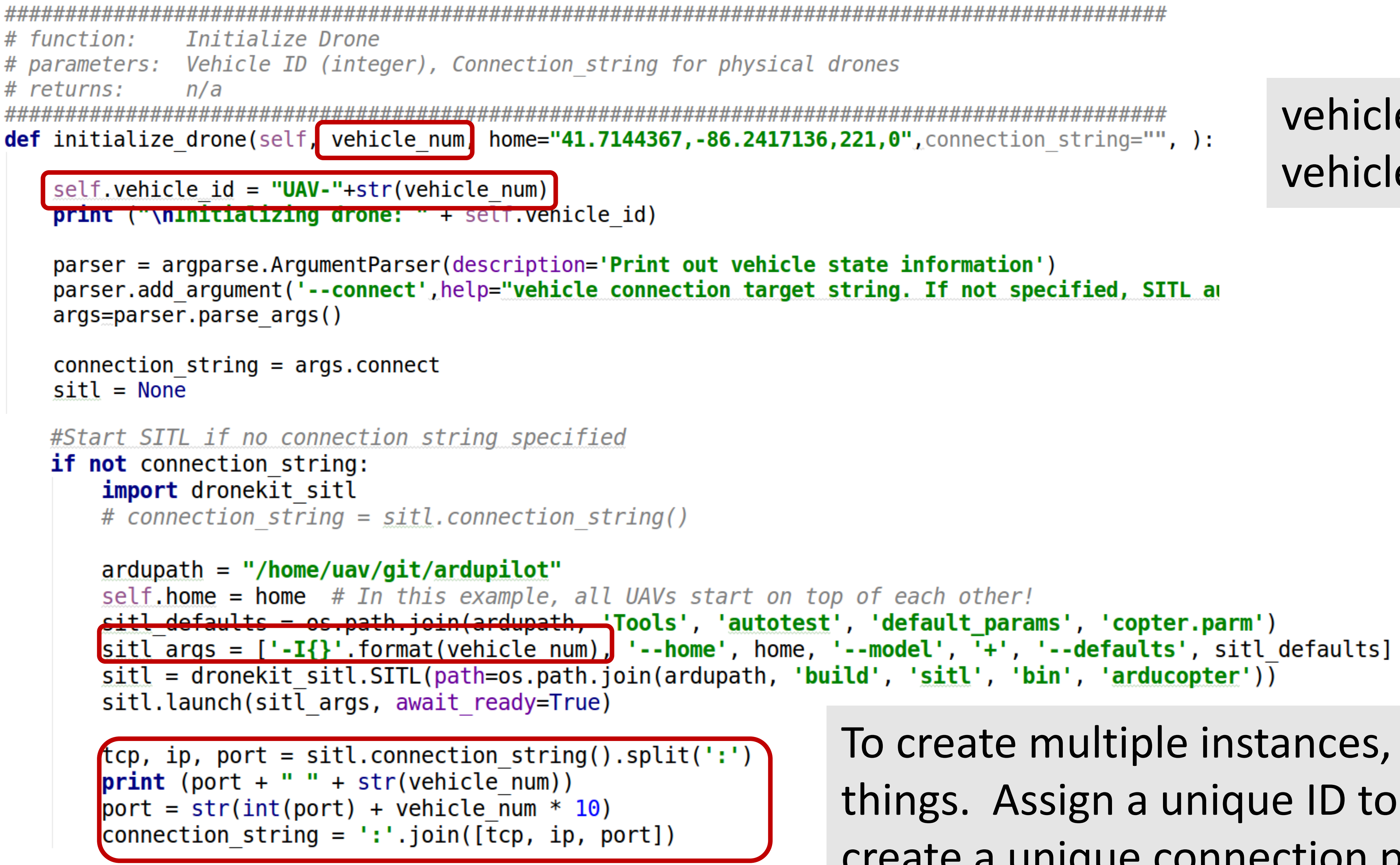


```
#Calling program for demonstrating multiple
from copter import UAV_Copter
import json

drones = list()
copter_counter = 0

# Create N copter vehicles (1)
for d in range(3):
    print("\nAdding drone: " + str(copter_co
    copter = UAV_Copter()  # Create instance
    copter.initialize_drone(copter_counter)
    copter_counter= copter_counter+1  # Used
    drones.append(copter)

# Show state
for d in drones:
    d.print_drone_state()

# Close all drones
for d in drones:
    d.close_vehicle()
```

In this very simple program we create a list of 3 drones.

Each drone is assigned a unique ID and is initialized as a SITL instance.

Copter Class

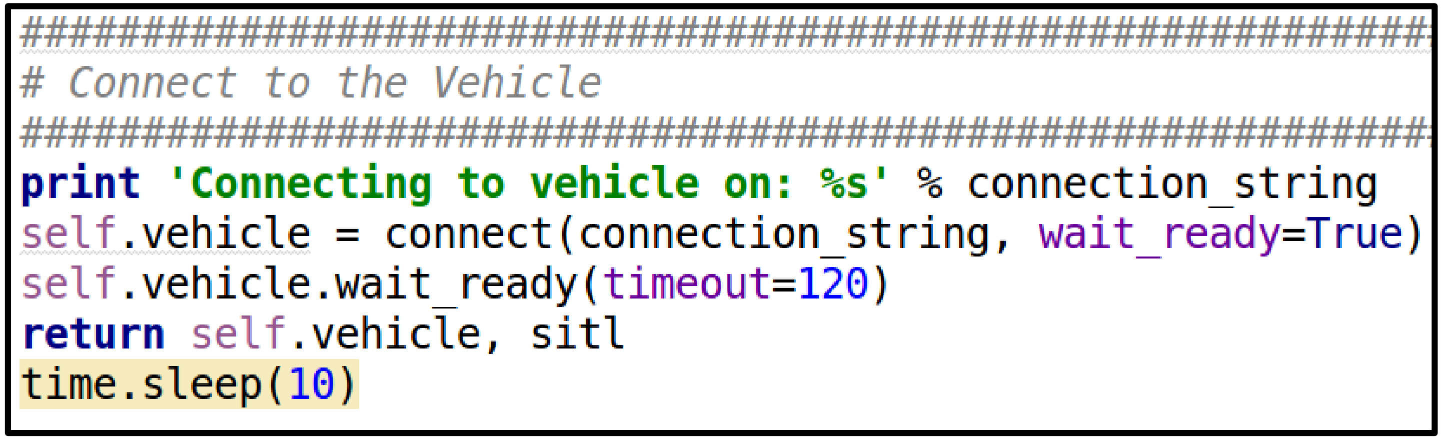


```
################################################################################
# function:    Initialize Drone
# parameters:  Vehicle ID (integer), Connection_string for physical drones
# returns:     n/a
################################################################################
def initialize_drone(self, vehicle_num, home="41.7144367,-86.2417136,221,0",connection_string="", ):

    self.vehicle_id = "UAV-"+str(vehicle_num)
    print ("\nInitializing drone: " + self.vehicle_id)

    parser = argparse.ArgumentParser(description='Print out vehicle state information')
    parser.add_argument('--connect',help="vehicle connection target string. If not specified, SITL au
    args=parser.parse_args()

    connection_string = args.connect
    sitl = None

    #Start SITL if no connection string specified
    if not connection_string:
        import dronekit_sitl
        # connection_string = sitl.connection_string()

        ardupath = "/home/uav/git/ardupilot"
        self.home = home  # In this example, all UAVs start on top of each other!
        sitl_defaults = os.path.join(ardupath, 'Tools', 'autotest', 'default_params', 'copter.parm')
        sitl_args = ['-I{}'.format(vehicle_num), '--home', home, '--model', '+', '--defaults', sitl_defaults]
        sitl = dronekit_sitl.SITL(path=os.path.join(ardupath, 'build', 'sitl', 'bin', 'arducopter'))
        sitl.launch(sitl_args, await_ready=True)

        tcp, ip, port = sitl.connection_string().split(':')
        print (port + " " + str(vehicle_num))
        port = str(int(port) + vehicle_num * 10)
        connection_string = ':'.join([tcp, ip, port])
```

vehicle_num is an int
vehicle_id is a string

To create multiple instances, you need to do two things. Assign a unique ID to the sitl_args and create a unique connection port.

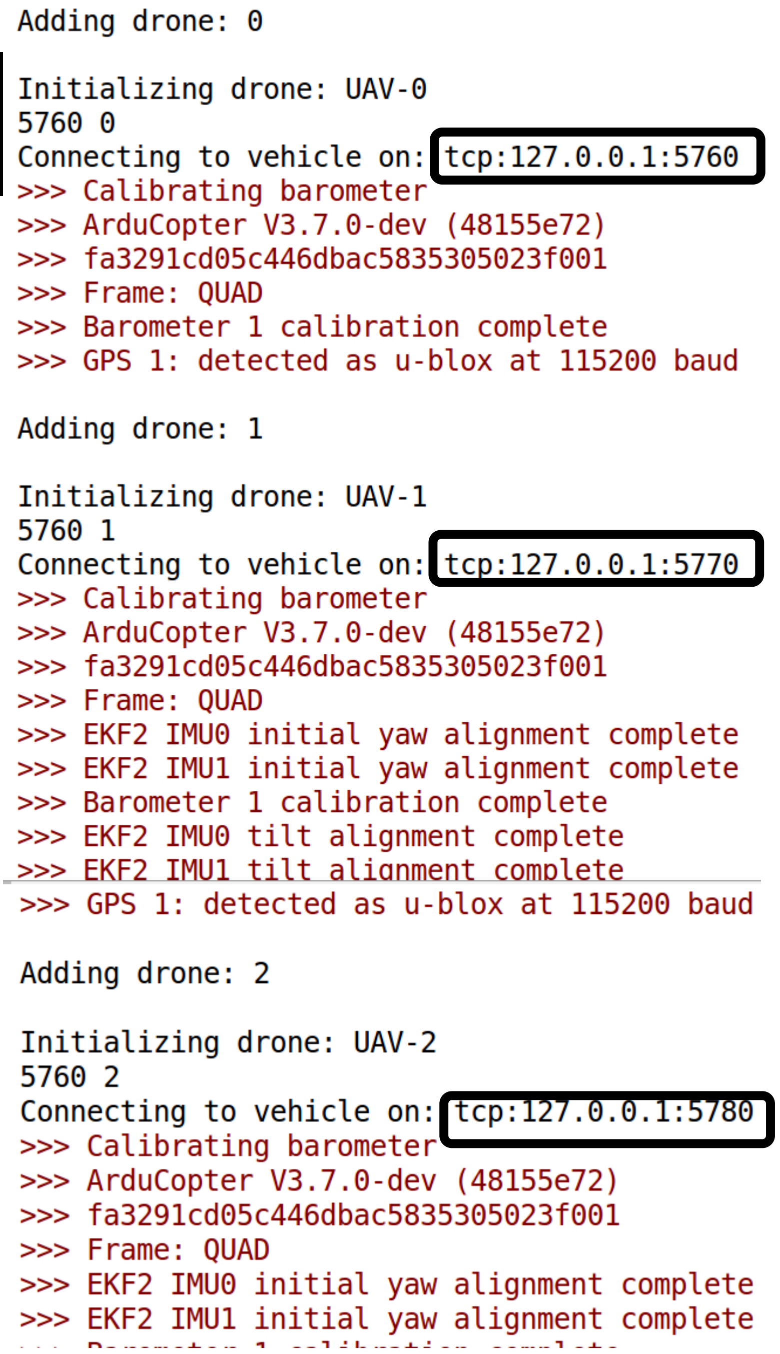
Nothing else changes…



```
###########################################################
# Connect to the Vehicle
###########################################################
print 'Connecting to vehicle on: %s' % connection_string
self.vehicle = connect(connection_string, wait_ready=True)
self.vehicle.wait_ready(timeout=120)
return self.vehicle, sitl
time.sleep(10)
```

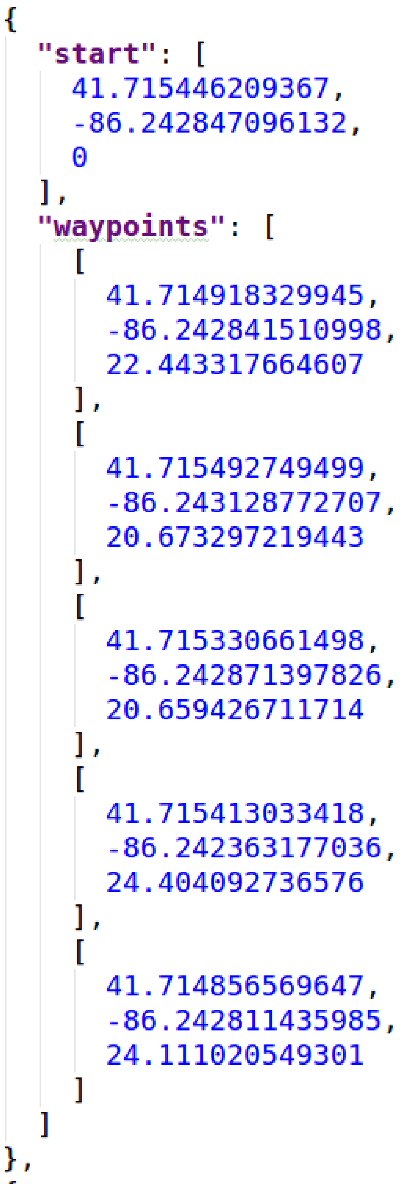
If you want the output for each drone to be separated you’ll need to add additional sleep commands.

Note the unique port numbers.



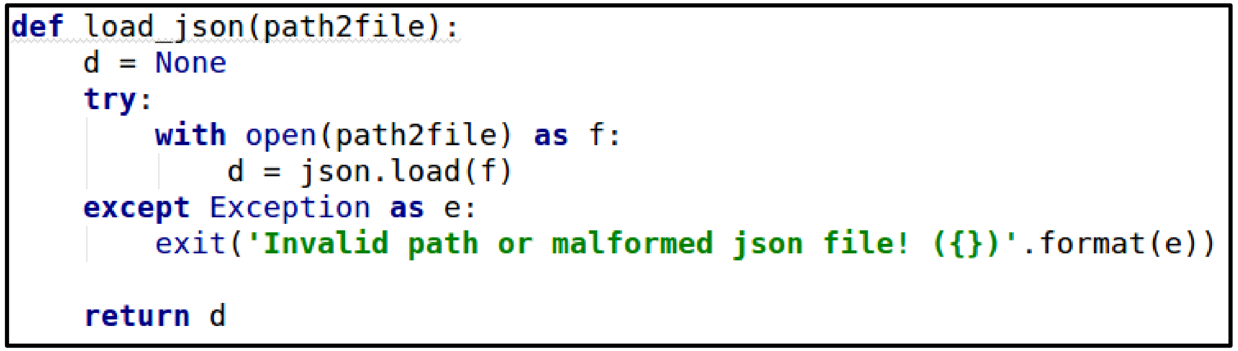
```
Adding drone: 0

Initializing drone: UAV-0
5760 0
Connecting to vehicle on: tcp:127.0.0.1:5760
>>> Calibrating barometer
>>> ArduCopter V3.7.0-dev (48155e72)
>>> fa3291cd05c446dbac5835305023f001
>>> Frame: QUAD
>>> Barometer 1 calibration complete
>>> GPS 1: detected as u-blox at 115200 baud

Adding drone: 1

Initializing drone: UAV-1
5760 1
Connecting to vehicle on: tcp:127.0.0.1:5770
>>> Calibrating barometer
>>> ArduCopter V3.7.0-dev (48155e72)
>>> fa3291cd05c446dbac5835305023f001
>>> Frame: QUAD
>>> EKF2 IMU0 initial yaw alignment complete
>>> EKF2 IMU1 initial yaw alignment complete
>>> Barometer 1 calibration complete
>>> EKF2 IMU0 tilt alignment complete
>>> EKF2 IMU1 tilt alignment complete
>>> GPS 1: detected as u-blox at 115200 baud

Adding drone: 2

Initializing drone: UAV-2
5760 2
Connecting to vehicle on: tcp:127.0.0.1:5780
>>> Calibrating barometer
>>> ArduCopter V3.7.0-dev (48155e72)
>>> fa3291cd05c446dbac5835305023f001
>>> Frame: QUAD
>>> EKF2 IMU0 initial yaw alignment complete
>>> EKF2 IMU1 initial yaw alignment complete
>>> Barometer 1 calibration complete
```

What if we want to configure our drones?

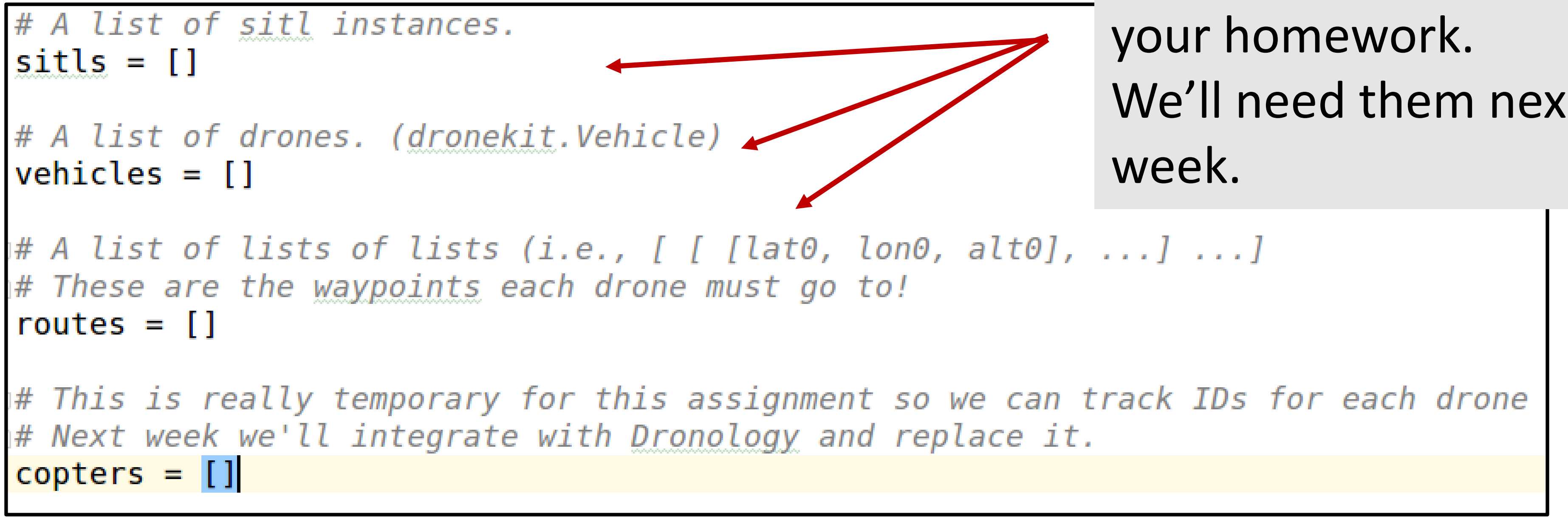


```
{
  "start": [
    41.715446209367,
    -86.242847096132,
    0
  ],
  "waypoints": [
    [
      41.714918329945,
      -86.242841510998,
      22.443317664607
    ],
    [
      41.715492749499,
      -86.243128772707,
      20.673297219443
    ],
    [
      41.715330661498,
      -86.242871397826,
      20.659426711714
    ],
    [
      41.715413033418,
      -86.242363177036,
      24.404092736576
    ],
    [
      41.714856569647,
      -86.242811435985,
      24.111020549301
    ]
  ]
},
```

We can specify home coordinates (start) and way-points in a json file.



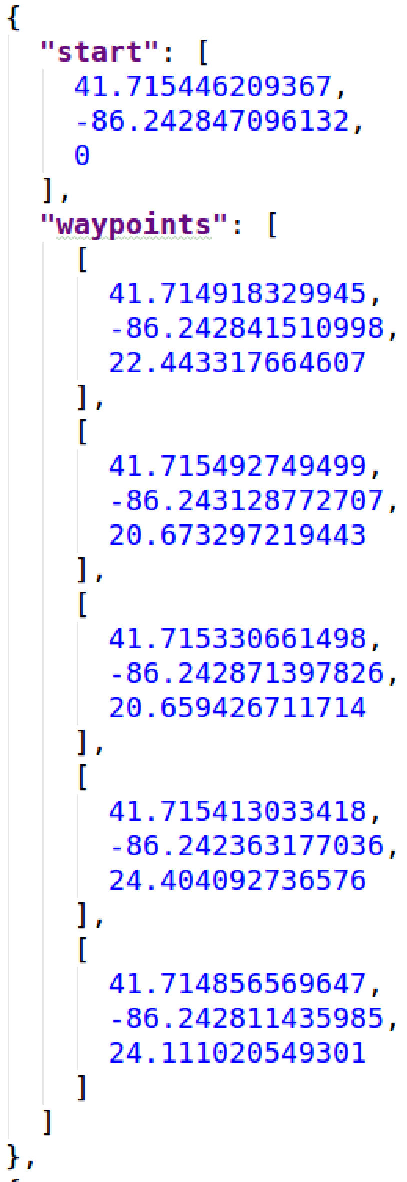
```python
def load_json(path2file):
    d = None
    try:
        with open(path2file) as f:
            d = json.load(f)
    except Exception as e:
        exit('Invalid path or malformed json file! ({})'.format(e))

    return d
```



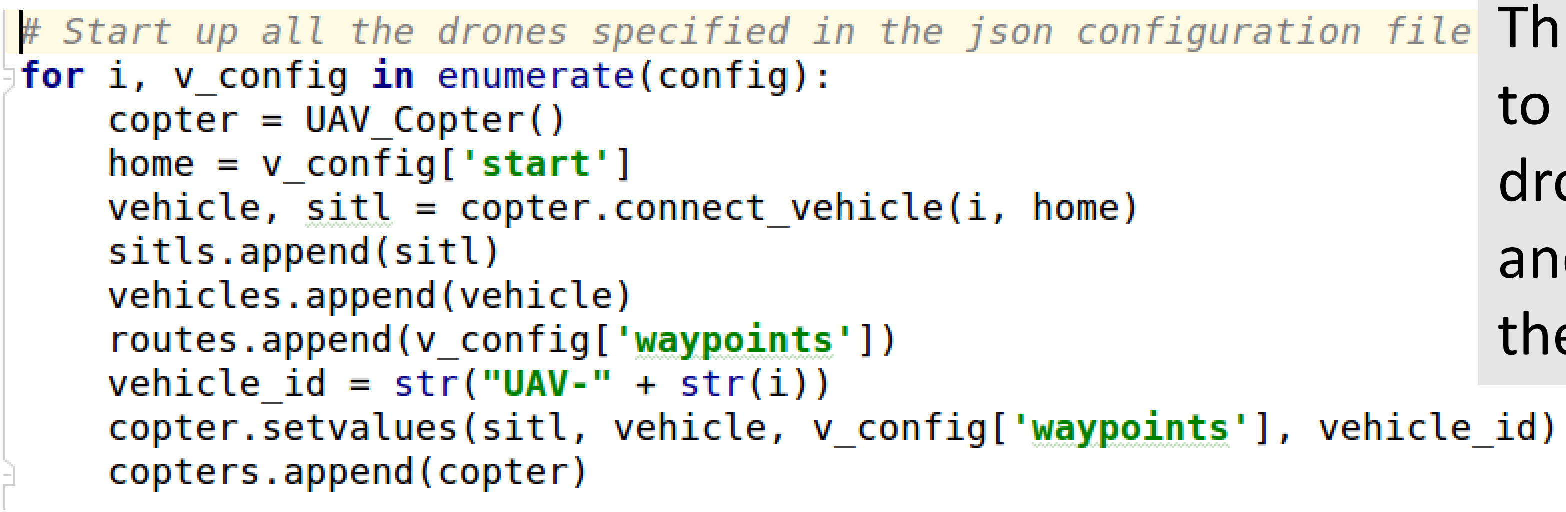
```python
# A list of sitl instances.
sitls = []

# A list of drones. (dronekit.Vehicle)
vehicles = []

# A list of lists of lists (i.e., [ [ [lat0, lon0, alt0], ...] ...]
# These are the waypoints each drone must go to!
routes = []

# This is really temporary for this assignment so we can track IDs for each drone
# Next week we'll integrate with Dronology and replace it.
copters = []
```

Please use these 3 data-structures in your homework. We'll need them next week.

What if we want to configure our drones?

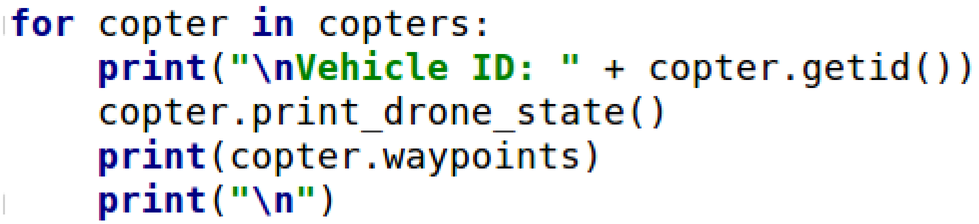


```
{
  "start": [
    41.715446209367,
    -86.242847096132,
    0
  ],
  "waypoints": [
    [
      41.714918329945,
      -86.242841510998,
      22.443317664607
    ],
    [
      41.715492749499,
      -86.243128772707,
      20.673297219443
    ],
    [
      41.715330661498,
      -86.242871397826,
      20.659426711714
    ],
    [
      41.715413033418,
      -86.242363177036,
      24.404092736576
    ],
    [
      41.714856569647,
      -86.242811435985,
      24.111020549301
    ]
  ]
},
```



```python
# Start up all the drones specified in the json configuration file
for i, v_config in enumerate(config):
    copter = UAV_Copter()
    home = v_config['start']
    vehicle, sitl = copter.connect_vehicle(i, home)
    sitls.append(sitl)
    vehicles.append(vehicle)
    routes.append(v_config['waypoints'])
    vehicle_id = str("UAV-" + str(i))
    copter.setvalues(sitl, vehicle, v_config['waypoints'], vehicle_id)
    copters.append(copter)
```

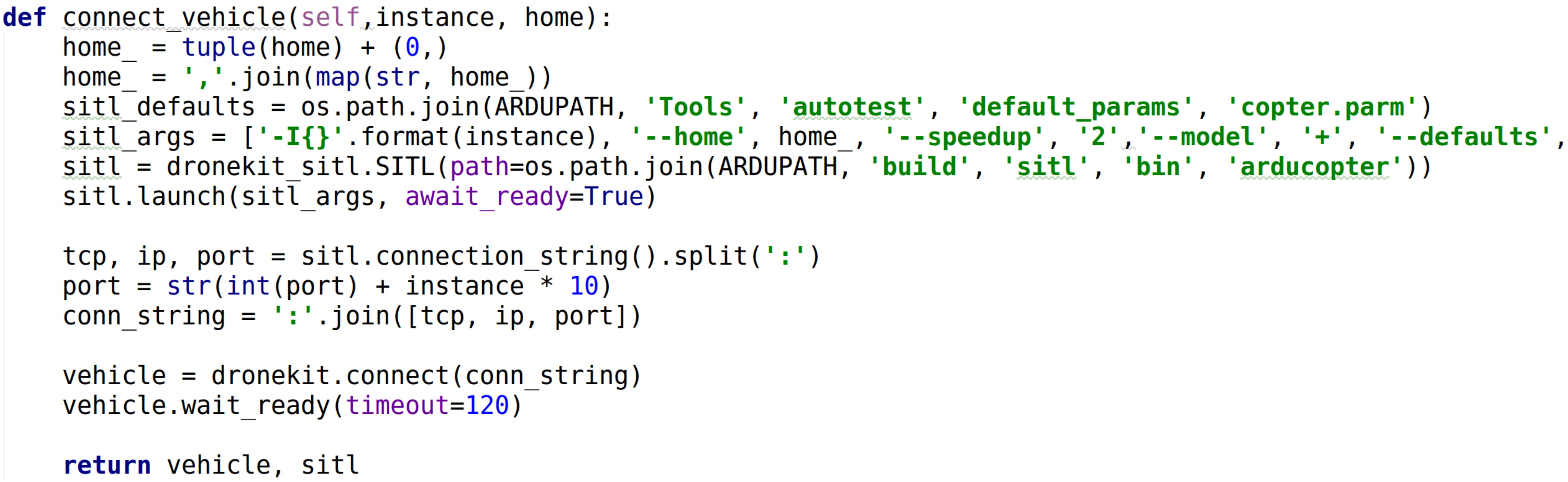
This code is provided to you. It loads the drone configurations and instances into the three lists.



```python
for copter in copters:
    print("\nVehicle ID: " + copter.getid())
    copter.print_drone_state()
    print(copter.waypoints)
    print("\n")
```

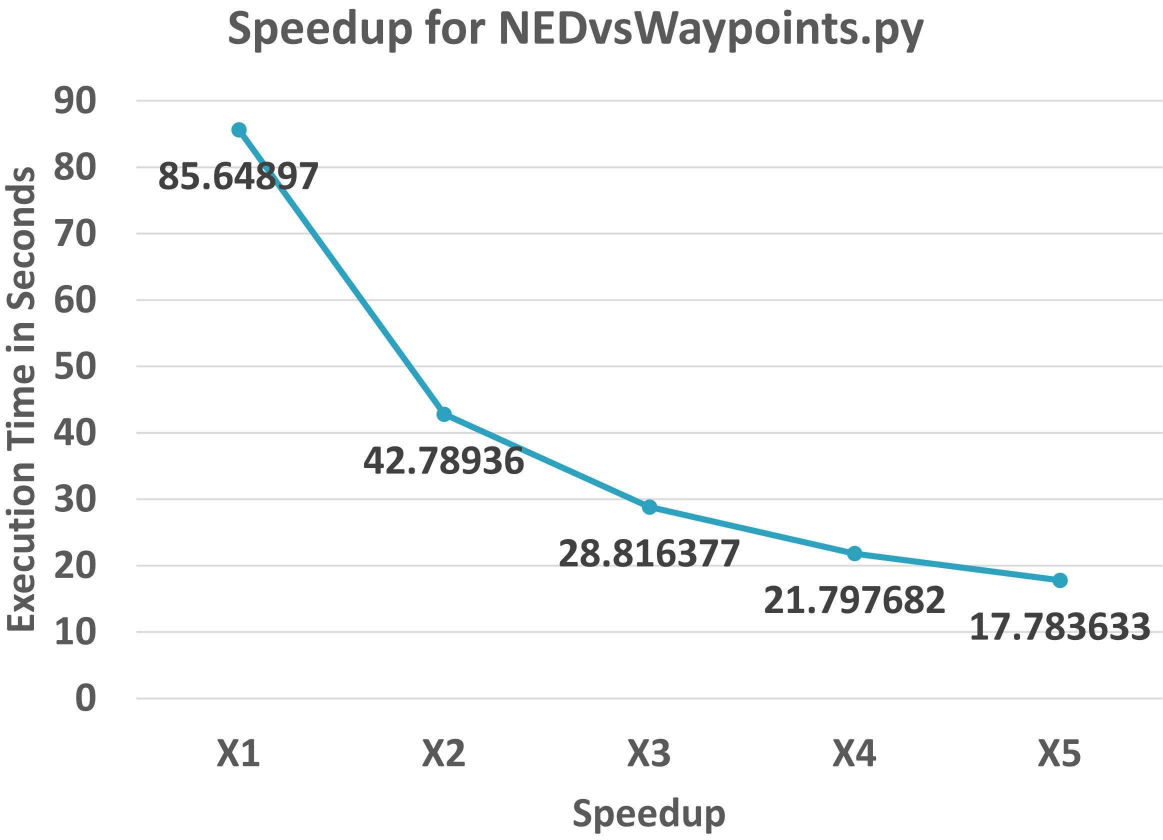
Did it work? Iterate through the copter instances to make sure everything is loaded.

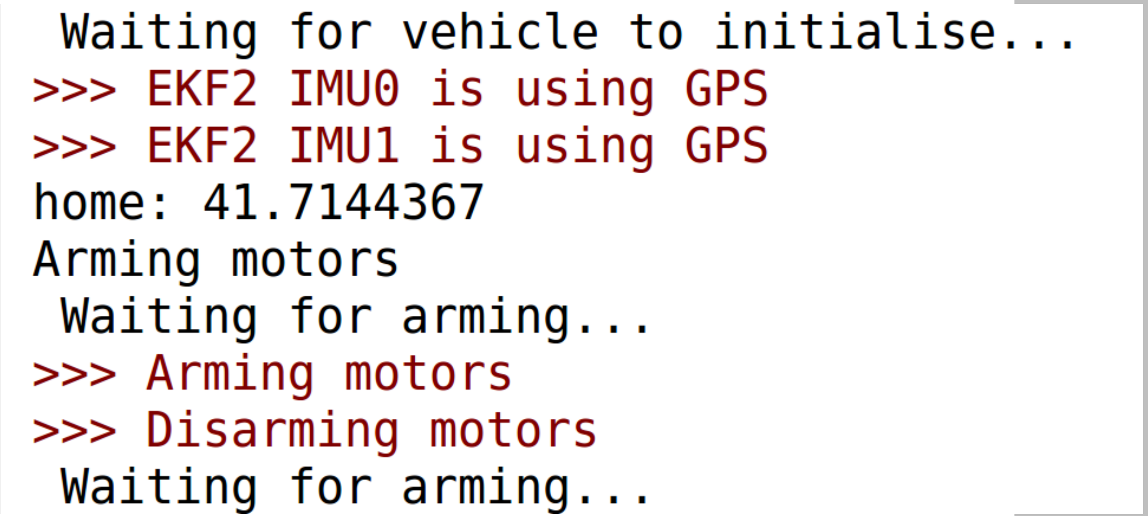
The new connection method



```
def connect_vehicle(self,instance, home):
    home_ = tuple(home) + (0,)
    home_ = ','.join(map(str, home_))
    sitl_defaults = os.path.join(ARDUPATH, 'Tools', 'autotest', 'default_params', 'copter.parm')
    sitl_args = ['-I{}'.format(instance), '--home', home_, '--speedup', '2','--model', '+', '--defaults',
    sitl = dronekit_sitl.SITL(path=os.path.join(ARDUPATH, 'build', 'sitl', 'bin', 'arducopter'))
    sitl.launch(sitl_args, await_ready=True)

    tcp, ip, port = sitl.connection_string().split(':')
    port = str(int(port) + instance * 10)
    conn_string = ':'.join([tcp, ip, port])

    vehicle = dronekit.connect(conn_string)
    vehicle.wait_ready(timeout=120)

    return vehicle, sitl
```

Simulation Speedup





```
 Waiting for vehicle to initialise...
>>> EKF2 IMU0 is using GPS
>>> EKF2 IMU1 is using GPS
home: 41.7144367
Arming motors
 Waiting for arming...
>>> Arming motors
>>> Disarming motors
 Waiting for arming...
```

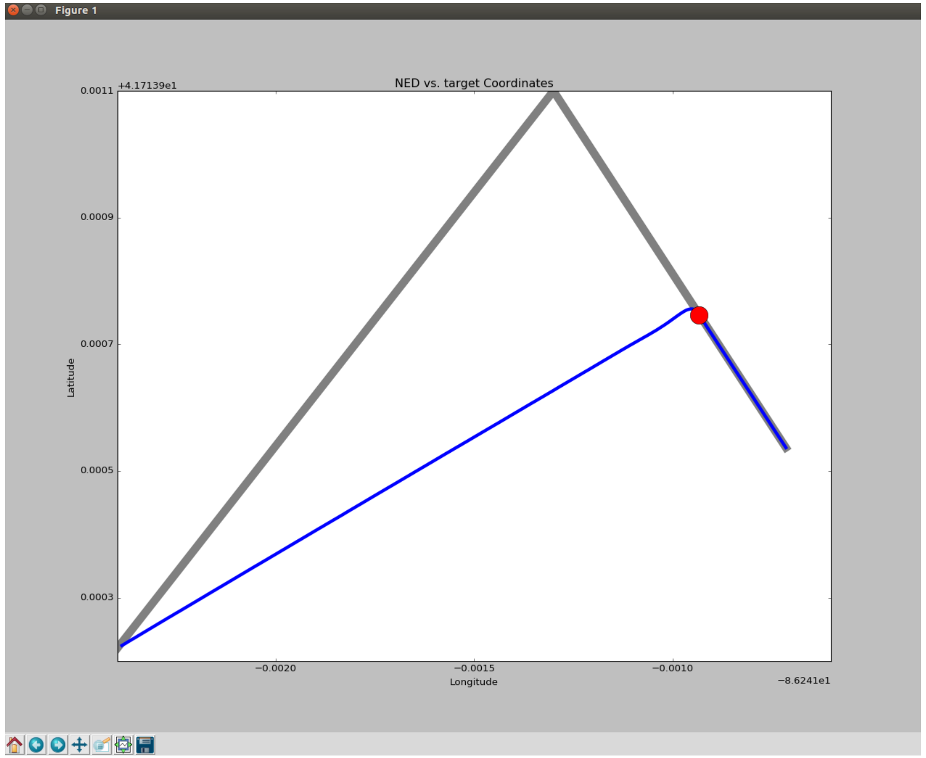


Figure 1
NED vs. target Coordinates
+4.17139e1
Latitude
−8.6241e1
Longitude