In [269…

```python
import socket
import sys
import os
from pathlib import Path
import pandas as pd
```

1) Copy DroneLab into your home directory

In [268…

```python
# Set up home directory and move dronelab
new_directory =os.getenv("HOME")+"/dronelab"
#Path(new_directory).mkdir(parents=True, exist_ok=True)
```

In [ ]:

```python
# git pull or copy dronelab source to directory if not exists
# Not finished
```

2) Configure initial experiment parameters, such as the # of Drones

In [275…

```python
# Set up and configure experiment by modifying the Simulation Matrix File
sim_matrix_dir =new_directory+"/input/Simulation_Matrix.xlsx"
sim_df = pd.read_excel(sim_matrix_dir)
```

In [278…

```python
#View simulation matrix
sim_df.head()

#in batch one, there are 12 relay drones, 12 spiral drones and 0 anti-social
#Social Drone definition: Drone with Form and Spiral behaviors
#Relay Drone definition: Drones with Relay behaviors
#Anti-social drone definition: Drones with Spiral and AntiSocial behaviors

#All drones are given a set of behaviors in this priority:
#Launch, Avoid, Climb, Recharge, Maintain Height, Spiral, Relay, Form, AntiSo

#Most drones already start up with the Launch, Avoid, Climb and Recharge beha
```

Out[278…

	Simulation	Batch ID	Relay	Spiral	Anti-Social	Total	WiFi Range
0	1.0	1	12.0	12.0	0.0	24.0	0.475
1	2.0	1	30.0	20.0	0.0	50.0	0.475
2	3.0	1	9.0	0.0	30.0	39.0	0.475
3	4.0	1	9.0	1.0	4.0	14.0	0.475
4	5.0	1	6.0	0.0	4.0	10.0	0.475

In [287…

```python
# Add one Anti-social drone to Simulation #1 (row 0)
sim_df.at[0,'Anti-Social']=1.0
# Dont forget to update the Total Column
sim_df['Total'] = sim_df['Relay']+ sim_df['Spiral'] + sim_df['Anti-Social']
sim_df.head()
#save to disk
sim_df.to_excel(sim_matrix_dir)
```

In [291…

```
# view changes
sim_df[0:1]
```

Out[291…

	Simulation	Batch ID	Relay	Spiral	Anti-Social	Total	WiFi Range
0	1.0	1	12.0	12.0	1.0	25.0	0.475

3) Launch DroneLab in a new terminal

In [292…

```
#Find a free and open port 8000–8999
Port_To_Test = 8007

testSocket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
location = ('localhost', Port_To_Test)
result_of_check = testSocket.connect_ex(location)
if result_of_check == 0:
    print("Port is open")
else:
    print("Port is NOT open")
```

```
Port is open
```

Launch drone lab code in a new terminal. Replace /home/mosi in 1a & 1b with your own home directory

1a. cd into $HOME/dronelab directory and make sure the code is compiled: /home/mosi/utils /jdk1.8.0_291/bin/javac @options_nix @classes

1b. Specify the free port in the -Dport variable below and run: /home/mosi/utils/jdk1.8.0_291 /bin/java -Dport=8007 -Dfile.encoding=UTF8 -Xmx128G -d64 -cp "ext-master/netty-all-4.1.7.Final.jar:ext-master/commons-collections4-4.2.jar:ext-master/commons-compress-1.18.jar:ext-master/curvesapi-1.05.jar:ext-master/drive:ext-master/javax.json-1.0.4.jar:ext-master /javax.json-api-1.0.jar:ext-master/javax.servlet-api-3.1.0.jar:ext-master/poi-4.0.1.jar:ext-master /poi-examples-4.0.1.jar:ext-master/poi-excelant-4.0.1.jar:ext-master/poi-ooxml-4.0.1.jar:ext-master/poi-ooxml-schemas-4.0.1.jar:ext-master/poi-scratchpad-4.0.1.jar:ext-master/xmlbeans-3.0.2.jar:ext-master/jfxrt.jar:ext-master/javax.json-api-1.0.jar:/home/mosi/dronelab/bin:/home /mosi/utils/jdk1.8.0_291/bin/jre:/home/mosi/ext-master/*:." dronelab.DroneLab

You should see the dronelab gui pop up.

4) Connect to your DroneLab instance from Python

In [256…

```
# Create a TCP/IP socket
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# Connect the socket to the port where the server is listening
server_address = ('localhost', Port_To_Test)
print(sys.stderr, 'connecting to %s port %s' % server_address)

sock.connect(server_address)
```

```
<ipykernel.iostream.OutStream object at 0x7fb603c29e80> connecting to localho
```

```
st port 8007
```

5) Start the DroneLab simulation

In [253…

```
# Start all runs from the simMatrix.xlsx spreadsheet - read from the simulati
message = b'Sim.Action.PerformAllRuns'
sock.sendall(message)
data = sock.recv(1024)
print(data)
#Command|Status
```

```
b'Sim.Action.PerformAllRuns|Executed'
```

In [ ]:

```
# Stop DroneLab Sim
```

In [ ]:

```
# Pause DroneLab Sim
```

6) Determine the Reward function

In [ ]:

```
## to do - detemrine the reward function
```

7) Observe the State and Take Actions

The agent has to be able to read in a state. It needs to be a matrix with the following cols:

- col0: time
- num survivors
- num survivors found
- enabled behaviors for each drone
- active behaviors for each drone?

columns

time, # survivors, # survivors found, drone 1, drone1 behavior 1, drone 1 behavior 2, etc....

The agent has to be able to take the following actions:

- start the sim - done
- pause the sim
- stop the sim
- add a behavior to a drone - done
- remove a behavior to a drone - done

In [257…

```
# Get the Active Behavior for each Drone
message = b'Drone.Get.ActiveBehaviors'
sock.sendall(message)
data = sock.recv(1024)
print(data)
```

```
b'Drone.Get.Drones|Executed|0:Launch'
```

In [258…

```
# Get all the behaviors enabled for each drone
message = b'Drone.Get.AllBehaviors'
sock.sendall(message)
data = sock.recv(1024)
print(data)
#Command|Status|droneId0:Behavior1;Behavior2;BehaviorN,droneId1:........
```

```
b'Drone.Get.Drones|Executed|0:Launch;Avoid;Climb;Recharge;Height;Flock;Seek;P
attern;Wander'
```

In [247…

```
# Remove an individual behavior - ***the order of the behaviors determines wh
# Command.DroneId.BehaviorName
message = b'Drone.Action.RemoveBehavior.1.Relay'
sock.sendall(message)
data = sock.recv(1024)
print(data)
```

```
b'Drone.Action.RemoveBehavior.1.Relay|Executed|'
```

In [ ]:

```
# Add an individual behavior
# Command.DroneId.BehaviorName
message = b'Drone.Action.AddBehavior.1.Relay'
sock.sendall(message)
data = sock.recv(1024)
print(data)

#Social Drone definition: Drone with Form and Spiral behaviors
#Relay Drone definition: Drones with Relay behaviors
#Anti-social drone definition: Drones with Spiral and AntiSocial behaviors
```

In [119…

```
message = b'Drone.Get.Drones'
sock.sendall(message)
data = sock.recv(1024)
print(data)
```

```
b''
```

In [ ]:

```
message = b'Drone.Action.AddBehavior.1.Relay'
sock.sendall(message)
data = sock.recv(1024)
print(data)
```

Other notes

In [ ]:

```
Need to Add:
    setDroneRole in Drone.Java
```

In [ ]:

```
#Behavior Order:
 behaviorOrder.add(Constants.STR_LAUNCH);
        behaviorOrder.add(Constants.STR_AVOID);
        behaviorOrder.add(Constants.STR_CLIMB);
        behaviorOrder.add(Constants.STR_RECHARGE);
        behaviorOrder.add(Constants.STR_MAINTAIN_HEIGHT);
        behaviorOrder.add(Constants.STR_SPIRAL);
        behaviorOrder.add(Constants.STR_RELAY);
        behaviorOrder.add(Constants.STR_FORM);
        behaviorOrder.add(Constants.STR_ANTI);
        behaviorOrder.add(Constants.STR_REPEL);
        behaviorOrder.add(Constants.STR_SEEK);
        behaviorOrder.add(Constants.STR_SCATTER);
        behaviorOrder.add(Constants.STR_ASSIGNED_PATH);
        behaviorOrder.add(Constants.STR_SEARCH);
        behaviorOrder.add(Constants.STR_WANDER);


            public static final String STR_AVOID = "Avoid";
    public static final String STR_SEEK = "Seek";
    public static final String STR_SEARCH = "Pattern";
    public static final String STR_WANDER = "Wander";
    public static final String STR_FORM = "Flock";
    public static final String STR_RECHARGE = "Recharge";
    public static final String STR_LAUNCH = "Launch";
    public static final String STR_MAINTAIN_HEIGHT = "Height";
    public static final String STR_CLIMB = "Climb";
    public static final String STR_SPIRAL = "Spiral";
    public static final String STR_SCATTER = "Scatter";
    public static final String STR_REPEL = "Repel";
    public static final String STR_RELAY = "Relay";
    public static final String STR_ANTI = "AntiSocial";
    public static final String STR_ASSIGNED_PATH = "AssignedPath";
```

In [ ]:

```
# Unused
message = b'Sim.Action.Start'
sock.sendall(message)
data = sock.recv(1024)
print(data)
message = b'Sim.Action.Stop'
sock.sendall(message)
data = sock.recv(1024)
print(data)
```