Name: Jesse Schoenwald-Oberbeck
Date: 25 March 2017
Current Module: Phase II Project in Python
Project Name: Mining

**Project Goals:**
Our task is to write a package called mining that contains class Overlord. This class will contain six Drones, which will exploit an alien landscape for it's delicious rocks.

**Considerations:**

- The Overlord must accept a number of ticks and be complete before these have depleted.
- The Overlord must have usable add map and action methods.
- Drones must have a move method.
- Overlord has a one second time limit for actions.
- Drones have 1 a millisecond limit to move.
- Drones have limited health, which is reduced by hitting acid, a wall, or another Drone.
- Minerals are gathered when collided with.
- Drones can see their surroundings at a distance of 1 square, and only in cardinal directions.

**Initial Design:**
When the Overlord object is created, it creates six Drones. The Drones will be of two types, assigned at deployment. Type one, scanner, methodically searches the map left to right, bottom to top, then top to bottom. It will continue until the top to bottom portion (second to occur) is completed, or it is recalled by the Overlord. The other type, roomba, bounces off obstacles and has a chance to choose a random new direction until recalled.

The Overlord generates the Drones and recieves the three maps. It deploys/assigns the first three Drones to the three maps in order, assigning those Drones as scanners. After being deployed each Drone will proceed to the lower left corner. Since the maps are two-diminsional arrays, there will always be a 0,0 coordinate, which is almost guaranteed to be a wall, and a 1,1 which will most likely be the lowest available space. The Overlord will then deploy the second wave of 3 Drones. These are sent as roomba type Drones, which will move about the map bouncing off of objects or otherwise moving semi-randomly. They should not be deployed on top of the other Drones, as the first wave will have already left for the bottom left corner of the map. Preceeding each actual movement of each Drone, they check their four surrounding spaces for minerals, gathering them instead of moving if they are present. If no mineral is present, they follow

their respective movement instructions. Once 20 percent of the initial ticks remain, the Overlord will signal the zerg units to return by setting a flag in each Drone. This flag causes the zerg to head back to the landing zone by assessing the difference between its x,y coordinates and those of the landing zone (these are saved to the Drone upon deployment). It assesses which axis has the greater distance from the objective, and proceeds to move closer on that axis, making the determination again on each turn until arrival. Once the Drone has arrived at the landing zone, it activates it's own beacon (another flag), to let the Overlord know to pick it up. This way the Overlord always uses the return method on the correct unit at the correct time.

**Communication Protocol:**
No network communication is required or used. The Overlord communicates back and forth with Drones via flags, and the Overlord communicates with the driver program via text interpretation, requiring strings to begin with a keyword, and contain information pertaining to the command, such as Drone ID number.

**Potential Pitfalls:**
Potential issues that I might run into while creating the project are communication with the driver program, and generally dealing with a blind search of maps for resources.

**Test Plan:**
User Test: Run the program repeatedly with varying maps of differing sizes, making changes along the way until a satisfactory average mineral count is achieved.

Test Cases:
Each change in code required the program to be run many times to find errors, required changes, and further required logic.

**Conclusion:**
A satisfactory average of minerals retrieved per run was achieved. The Overlord is mostly unused, and most of the logic is rudimentary if elif else chains in the Drone. These movement determination chains are used rather than a more complex algorithm (such as the A* algorithm mentioned) because speed is a factor in managing one's units, and if statments take very little time to evaluate. Additionally, given that our units are blind to the map they are deposited on, most algorithms make no sense, as they require knowledge of the location of the objective relative to the start point. Establishing this knowledge would take valuable time (ticks) while exploring the map to find the objectives/end-points, when the Drones could be collecting those objectives for turn-in. Ultimately, it seemed as if the simplest choice was the best.