**Implementing a Route Planner**

In this project you will use A\* search to implement a "Google-maps" style route planning algorithm.

The map above (run the code cell if you don't see it) shows a disconnected network of 10 intersections. The two intersections on the left are connected to each other but they are not connected to the rest of the road network. On the graph above, the edge between 2 nodes(intersections) represents a literal straight road not just an abstract connection of 2 cities.

These Map objects have two properties you will want to use to implement A\* search: intersections and roads

**Intersections**

The intersections are represented as a dictionary.

In this example, there are 10 intersections, each identified by an x,y coordinate. The coordinates are listed below. You can hover over each dot in the map above to see the intersection number.

**Roads**

The roads property is a list where, if i is an intersection, roads[i] contains a list of the intersections that intersection i connects to.

### Advanced Visualizations

The map above shows a network of roads which spans 40 different intersections (labeled 0 through 39).

The show\_map function which generated this map also takes a few optional parameters which might be useful for visualizaing the output of the search algorithm you will write.

* start - The "start" node for the search algorithm.
* goal - The "goal" node.
* path - An array of integers which corresponds to a valid sequence of intersection visits on the map.

### Writing your algorithm

You should open the file student\_code.py in another tab and work on your algorithm there. Do that by selecting File > Open and then selecting the appropriate file.

The algorithm you write will be responsible for generating a path like the one passed into show map above. In fact, when called with the same map, start and goal, as above you algorithm should produce the path [5, 16, 37, 12, 34]

> shortest\_path(map\_40, 5, 34)

[5, 16, 37, 12, 34]

### Testing your Code

If the code below produces no errors, your algorithm is behaving correctly. You are almost ready to submit! Before you submit, go through the following submission checklist:

**Submission Checklist**

1. Does my code pass all tests?
2. Does my code implement A\* search and not some other search algorithm?
3. Do I use an **admissible heuristic** to direct search efforts towards the goal?
4. Do I use data structures which avoid unnecessarily slow lookups?

When you can answer "yes" to all of these questions, submit by pressing the Submit button in the lower right!