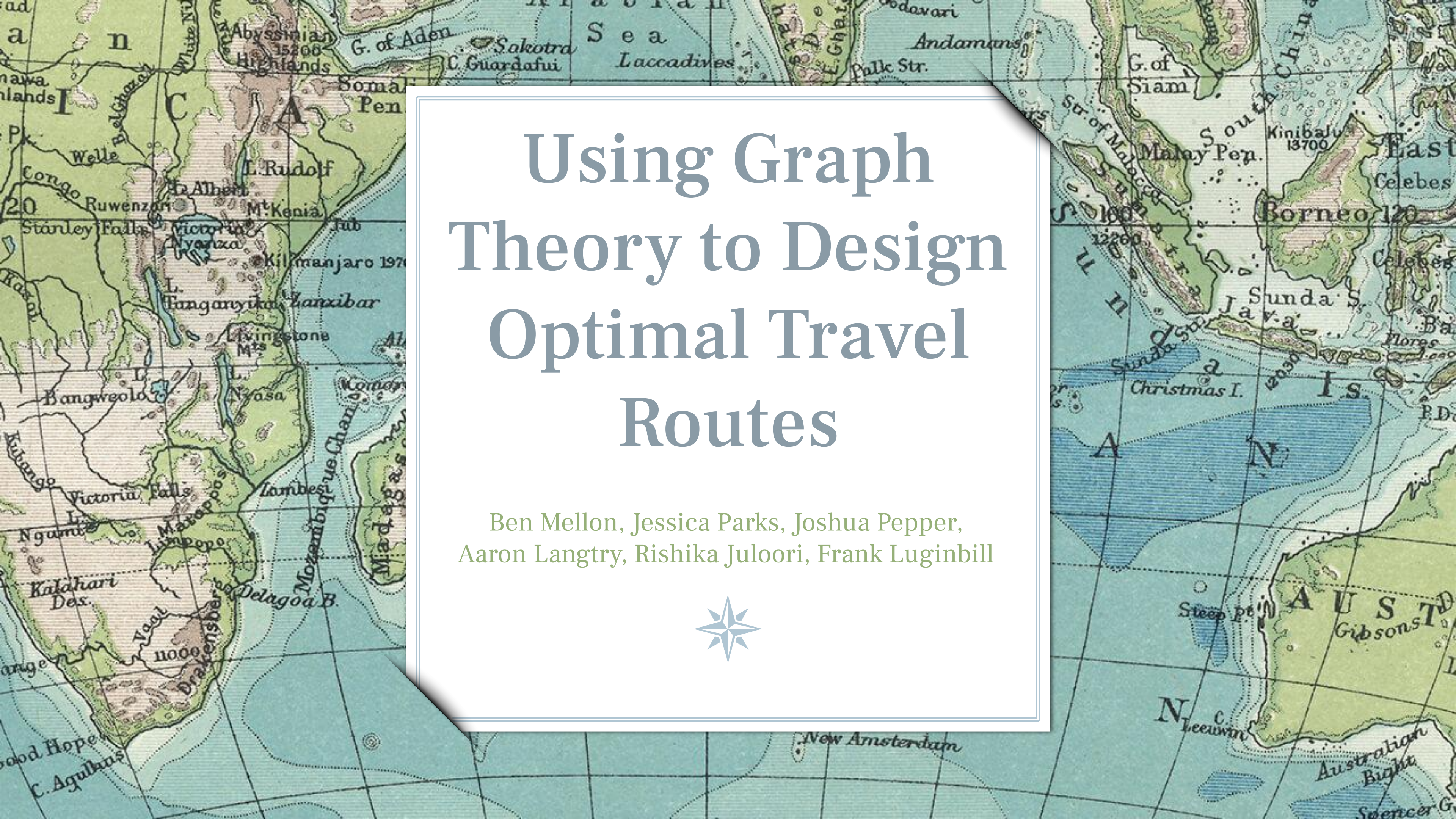
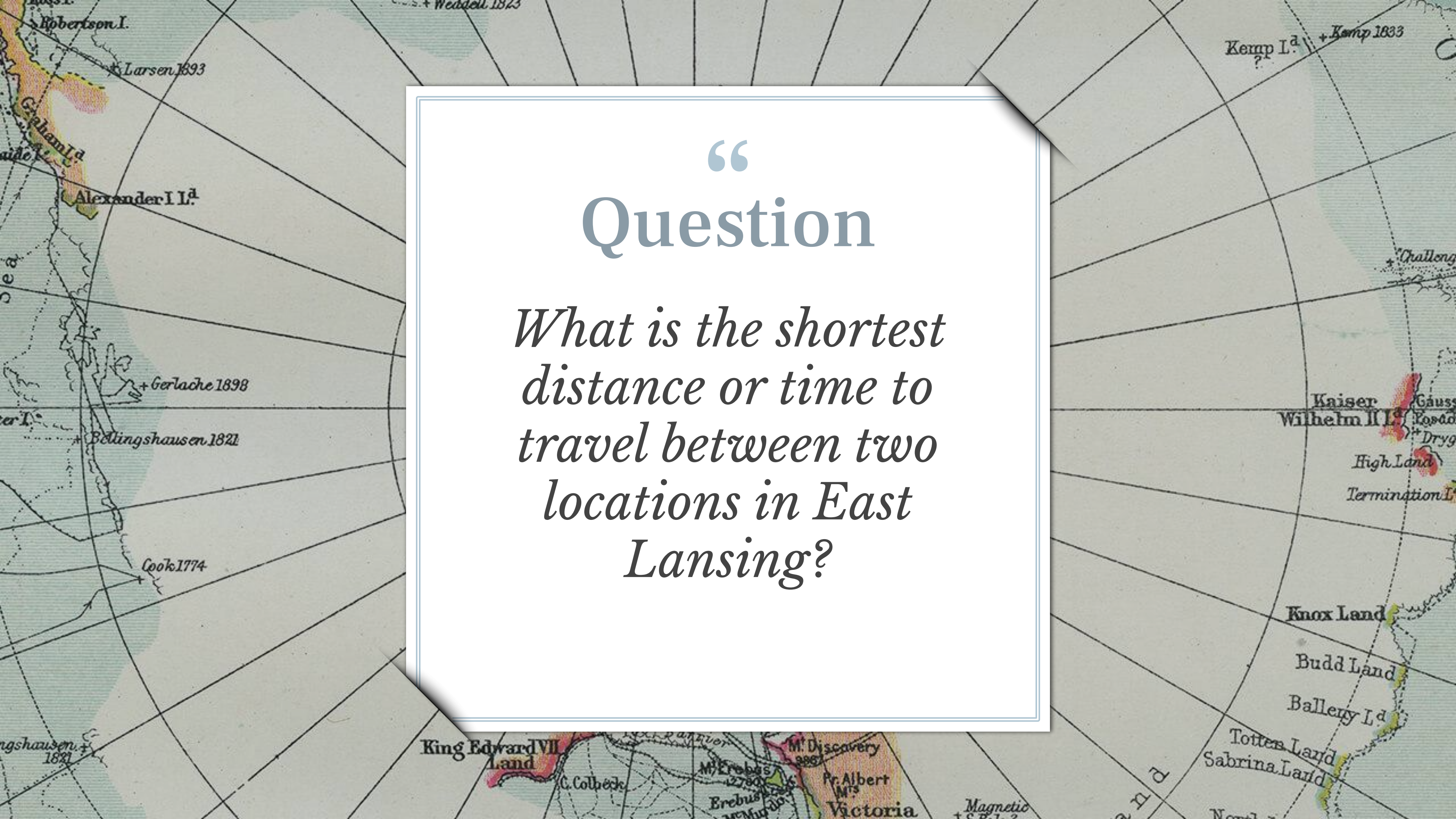


Using Graph Theory to Design Optimal Travel Routes

Ben Mellon, Jessica Parks, Joshua Pepper, Aaron Langtry, Rishika Juloori, Frank Luginbill



“

Question

What is the shortest distance or time to travel between two locations in East Lansing?

Background/Motivation

- MSU is the 8th largest campus
- MSU interactive map is incomplete and campus specific
- We wanted to make an interface with more complete list of location on-campus and off-campus

Model - Spatial Relationships

Use of Graph Theory

- **Nodes**: Locations on Campus
  - There are smaller nodes between the Campus nodes that have small edges between them
  - Nodes vary based on mode of transportation
- **Edges**: Streets, sidewalks
  - The Edges varies on the mode of transportation chosen
- **Graph Type:**
  - *Directed network*: one way streets require direction
  - *Weighted edges*: some roads have faster speed than others
    - Networkx allows us to do this

Computational Techniques

Packages

- NetworkX
- OSMnx
- Folium
- GeoPy

Resources

- Geoff Boeing Website (creator of OSMnx)
- Geoff Boeing OSMnx Github
- OSMnx User Guide

Computational Techniques

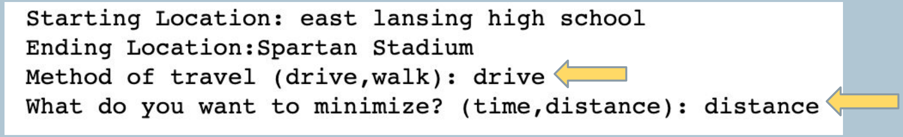
1. **Install Packages**
    a. Geopy: finds latitude and longitude based on string of location name
    b. OSMnx: creates graph of locations with latitude and longitude; plots route
    c. Folium: uses OSMnx to create user friendly, interactive map
2. **User inputs**
    a. Start and End location in East Lansing
    b. Type of travel
    c. Minimize distance traveled or time
3. **Check input strings**
    a. Make sure user inputs are actual words and locations
    b. Otherwise ask user again for information

Computational Technique

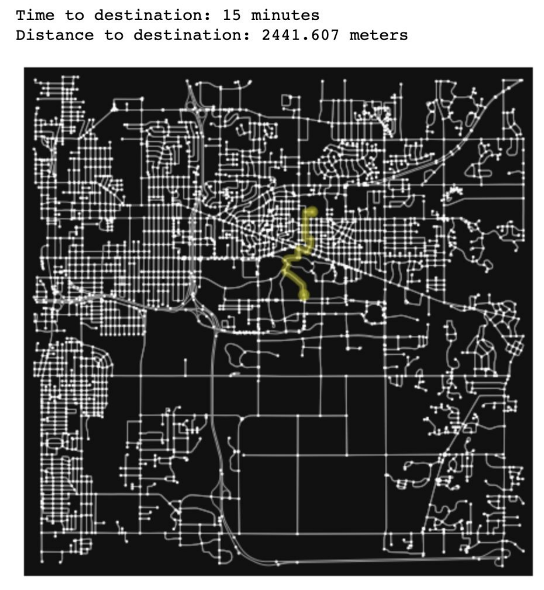
4. **Obtain latitude and longitude of start and end locations**
   a. Geopy takes in location strings, checks against locations within Geopy
5. **Find nearest OSMnx node to Geopy latitude and longitudes**
   a. OSMnx finds nearest node to longitude and latitude
   b. Returns address latitude and longitude
6. **Create objects**
   a. Create graph object with information
   b. Find route travel time
   c. Find route distance
7. **Display graph, route, and route information**

Answers

- Successfully able to find shortest **distance** route between two location in EL
- Shortest route can be found by drive or walking

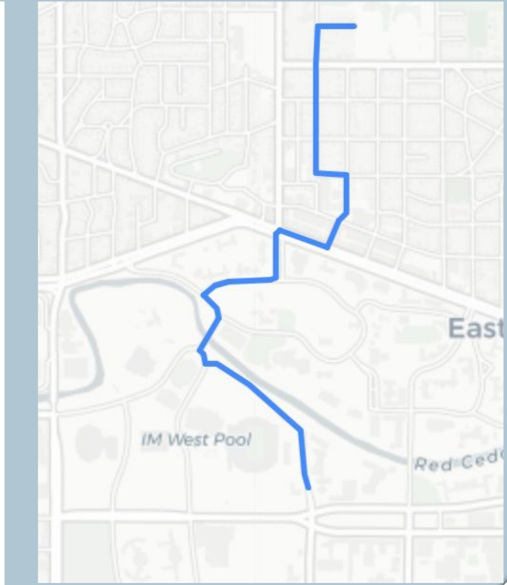


```
Starting Location: east lansing high school
Ending Location:Spartan Stadium
Method of travel (drive,walk): drive
What do you want to minimize? (time,distance): distance
```



```
Time to destination: 15 minutes
Distance to destination: 2441.607 meters
```

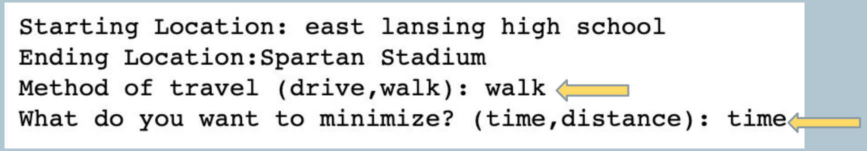
OSMnx Map



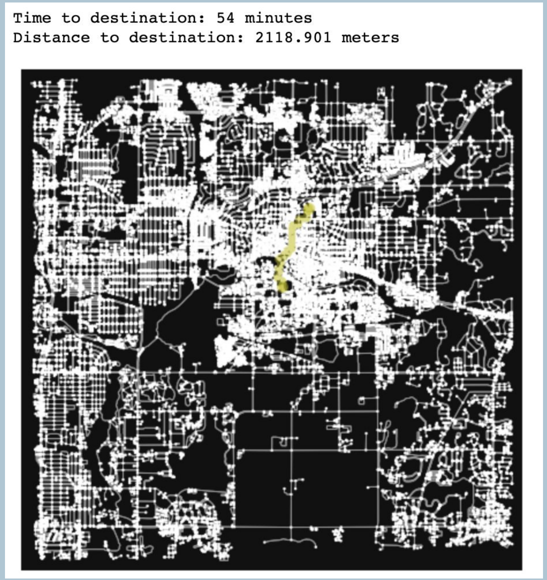
Folium Map

Answers

- Successfully able to find shortest route **by time** between two location in EL
- Shortest route can be found by drive or walking

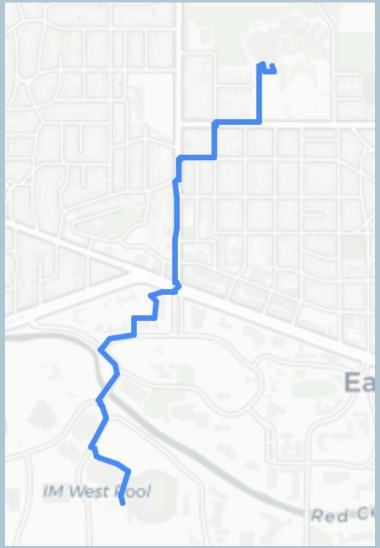


```
Starting Location: east lansing high school
Ending Location:Spartan Stadium
Method of travel (drive,walk): walk
What do you want to minimize? (time,distance): time
```



```
Time to destination: 54 minutes
Distance to destination: 2118.901 meters
```

OSMnx Map



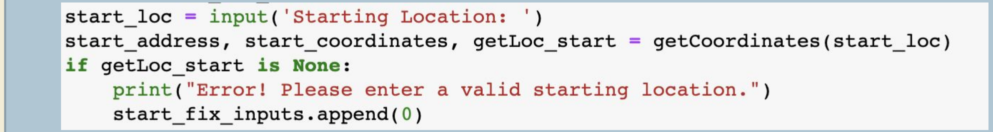
Folium Map

Answers

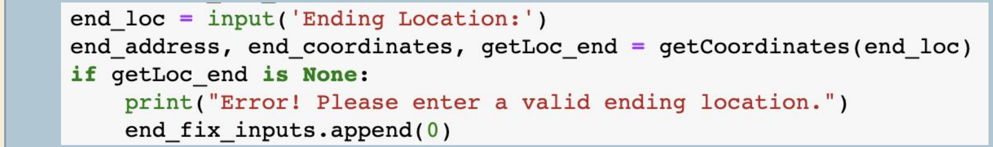
Code contains checks for the following:

- User input is a location that exists in Geopy
- User transportation method and minimization strategy input are appropriate

Checking Start and End are locations in Geocode

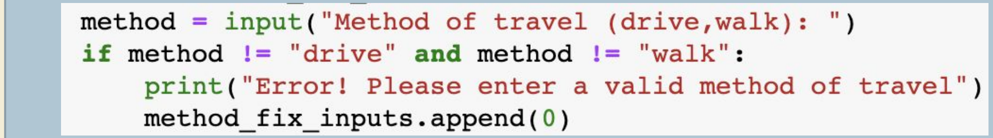


```python
start_loc = input('Starting Location: ')
start_address, start_coordinates, getLoc_start = getCoordinates(start_loc)
if getLoc_start is None:
    print("Error! Please enter a valid starting location.")
    start_fix_inputs.append(0)
```



```python
end_loc = input('Ending Location:')
end_address, end_coordinates, getLoc_end = getCoordinates(end_loc)
if getLoc_end is None:
     print("Error! Please enter a valid ending location.")
     end_fix_inputs.append(0)
```

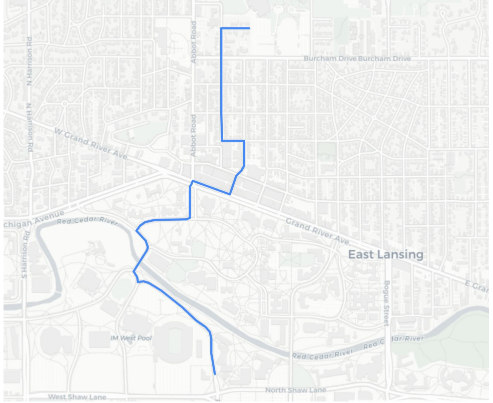
Checking travel type is a recognizable string

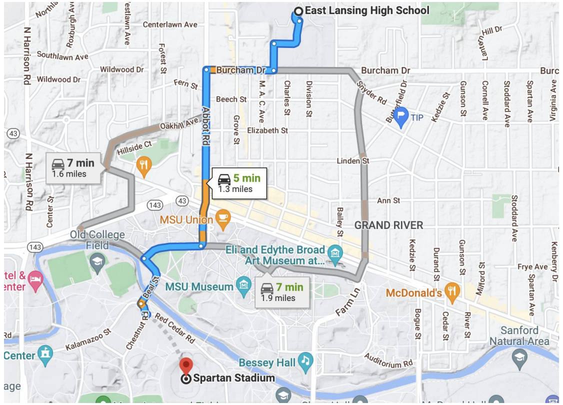


```python
method = input("Method of travel (drive,walk): ")
if method != "drive" and method != "walk":
     print("Error! Please enter a valid method of travel")
     method_fix_inputs.append(0)
```

Difficulties and Complications

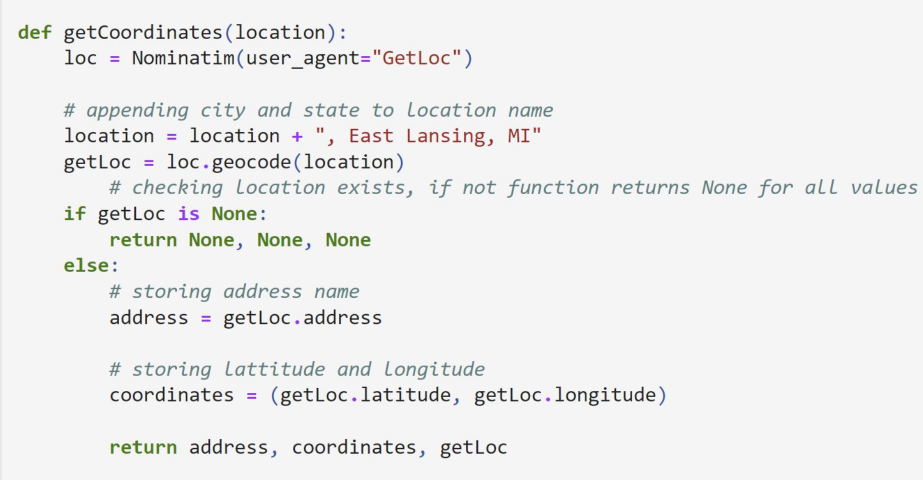
- Nodes list not exhaustive
  - Causes some routes to take small, unnecessary detours because nodes are not in every single location





Difficulties and Complications

- **Check user input is actual East Lansing location**
  - *If input type returns type ‘None’ then it is not a location*



```
def getCoordinates(location):
    loc = Nominatim(user_agent="GetLoc")

    # appending city and state to location name
    location = location + ", East Lansing, MI"
    getLoc = loc.geocode(location)
        # checking location exists, if not function returns None for all values
    if getLoc is None:
        return None, None, None
    else:
        # storing address name
        address = getLoc.address

        # storing lattitude and longitude
        coordinates = (getLoc.latitude, getLoc.longitude)

        return address, coordinates, getLoc
```

Difficulties and Complications

- **Unable to use live traffic data**
  - Google Maps Platform in javascript
  - Python live traffic data required payment
- **Package functions and package compatibility**
  - How packages interacted with one another
  - Identifying functions within packages that were new to us
- **Long Run Time**
  - Takes a while for graphs to be printed

Thank you!