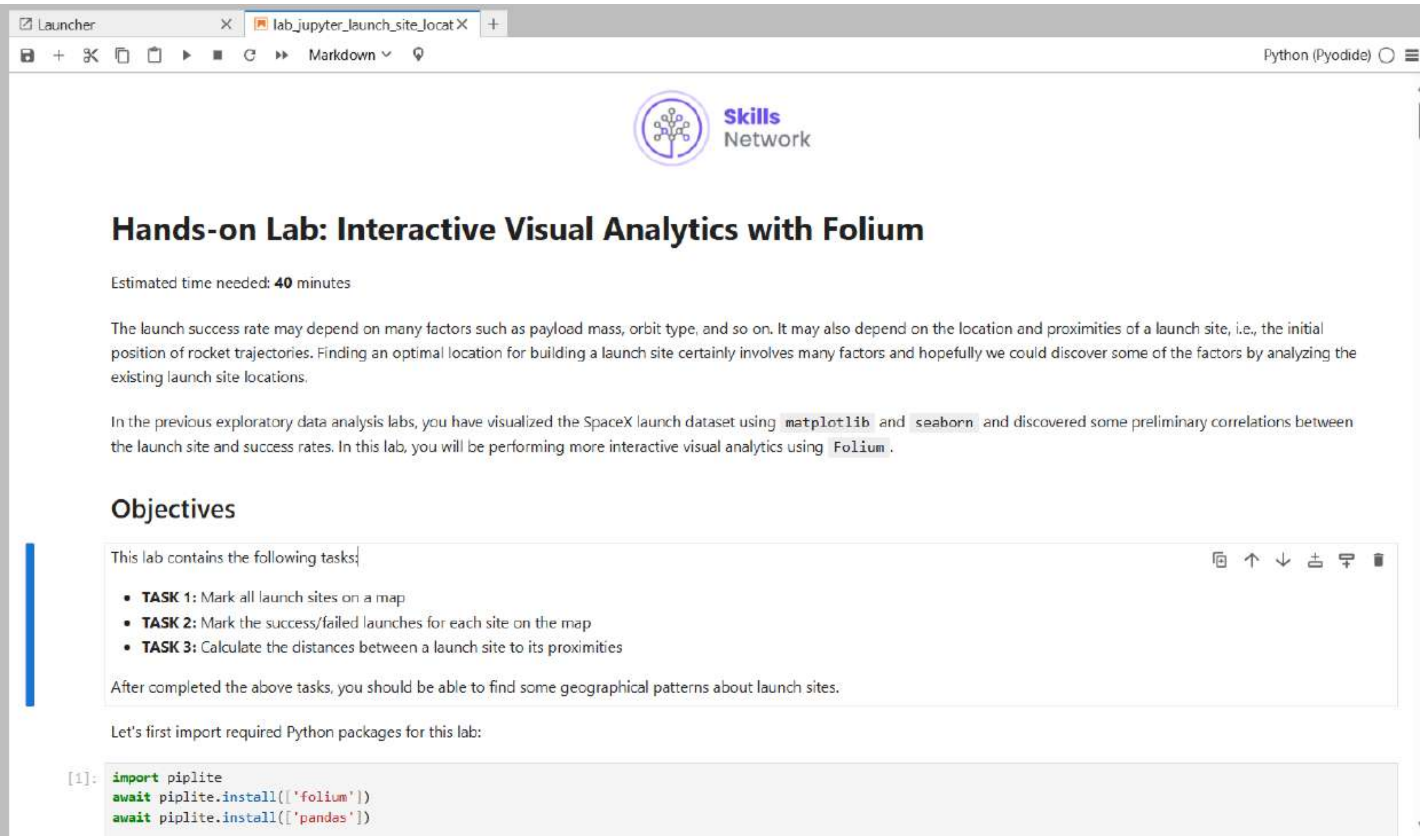


# Hands-on Lab: Interactive Visual Analytics with Folium

Estimated time needed: **40** minutes

The launch success rate may depend on many factors such as payload mass, orbit type, and so on. It may also depend on the location and proximities of a launch site, i.e., the initial position of rocket trajectories. Finding an optimal location for building a launch site certainly involves many factors and hopefully we could discover some of the factors by analyzing the existing launch site locations.

In the previous exploratory data analysis labs, you have visualized the SpaceX launch dataset using `matplotlib` and `seaborn` and discovered some preliminary correlations between the launch site and success rates. In this lab, you will be performing more interactive visual analytics using `Folium`.

## Objectives

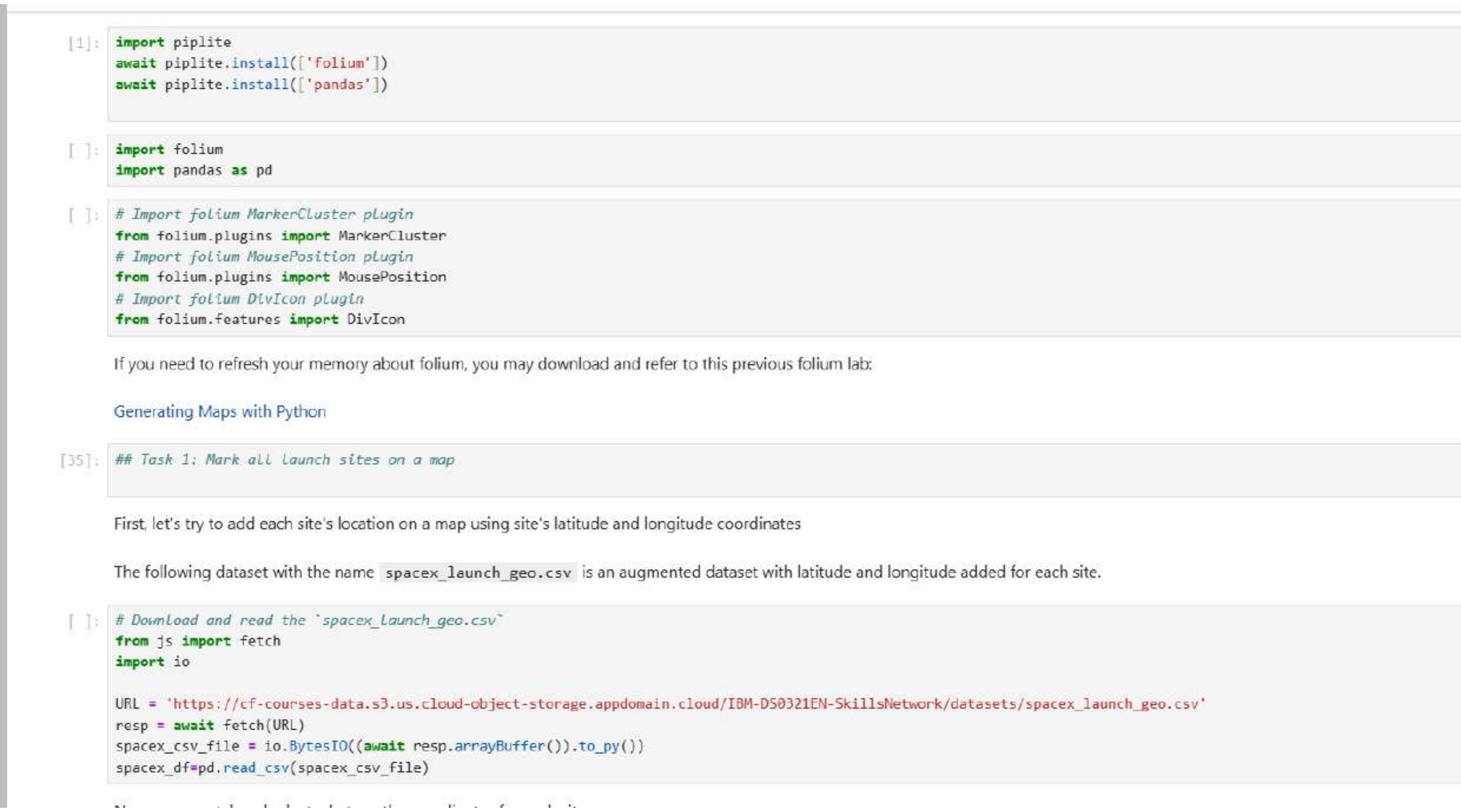
This lab contains the following tasks:

- **TASK 1:** Mark all launch sites on a map
- **TASK 2:** Mark the success/failed launches for each site on the map
- **TASK 3:** Calculate the distances between a launch site to its proximities

After completed the above tasks, you should be able to find some geographical patterns about launch sites.

Let's first import required Python packages for this lab:

[1]:
```
import piplite
await piplite.install(['folium'])
await piplite.install(['pandas'])
```



[1]:
```
import piplite
await piplite.install(['folium'])
await piplite.install(['pandas'])
```

[ ]:
```
import folium
import pandas as pd
```

[ ]:
```
# Import folium MarkerCluster plugin
from folium.plugins import MarkerCluster
# Import folium MousePosition plugin
from folium.plugins import MousePosition
# Import folium DivIcon plugin
from folium.features import DivIcon
```

If you need to refresh your memory about folium, you may download and refer to this previous folium lab:

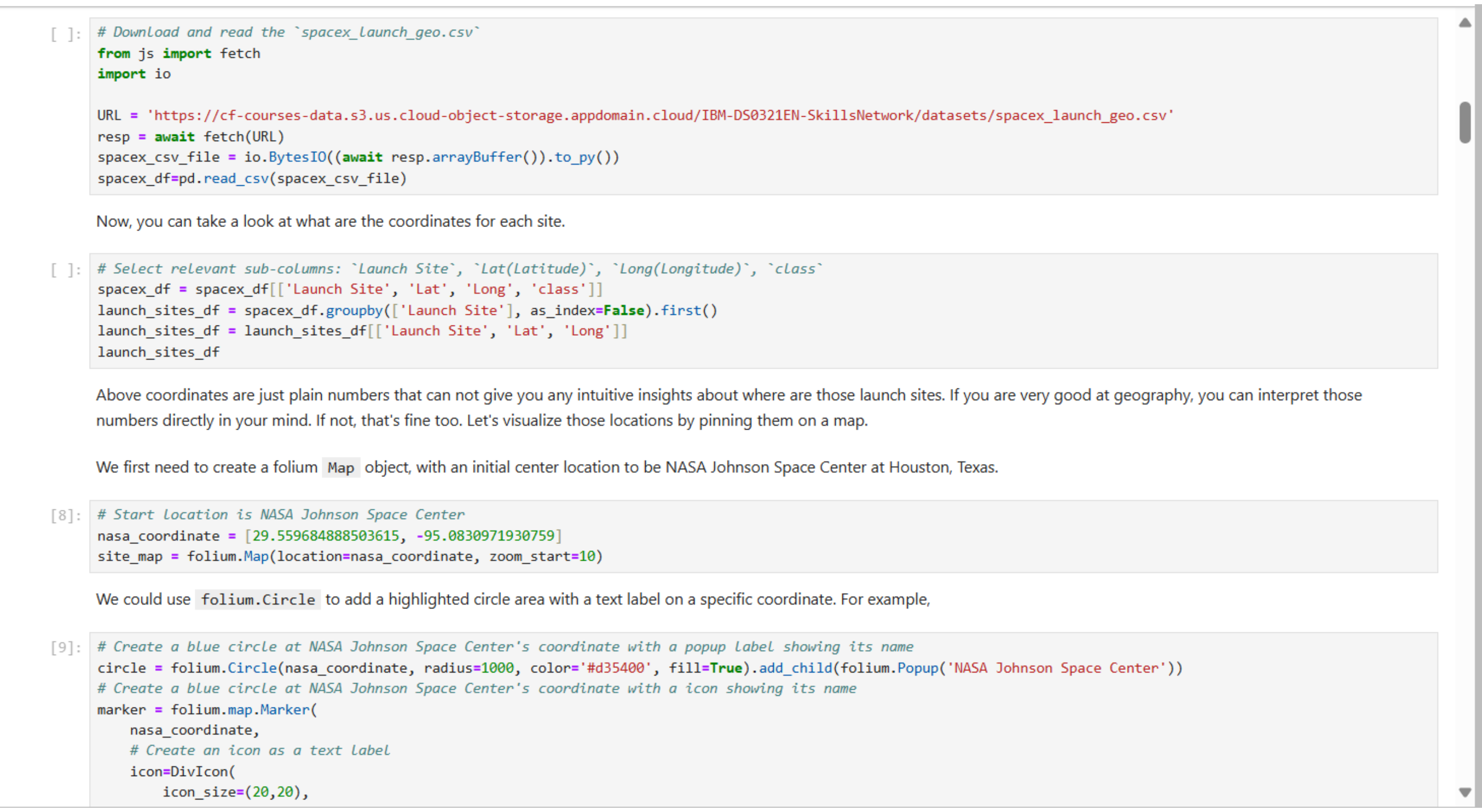
Generating Maps with Python

[35]:
```
## Task 1: Mark all launch sites on a map
```

First, let's try to add each site's location on a map using site's latitude and longitude coordinates

The following dataset with the name `spacex_launch_geo.csv` is an augmented dataset with latitude and longitude added for each site.

[ ]:
```
# Download and read the `spacex_launch_geo.csv`
from js import fetch
import io

URL = 'https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/spacex_launch_geo.csv'
resp = await fetch(URL)
spacex_csv_file = io.BytesIO((await resp.arrayBuffer()).to_py())
spacex_df=pd.read_csv(spacex_csv_file)
```



[ ]:

```python
# Download and read the `spacex_launch_geo.csv`
from js import fetch
import io

URL = 'https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/spacex_launch_geo.csv'
resp = await fetch(URL)
spacex_csv_file = io.BytesIO((await resp.arrayBuffer()).to_py())
spacex_df=pd.read_csv(spacex_csv_file)
```

Now, you can take a look at what are the coordinates for each site.

[ ]:

```python
# Select relevant sub-columns: `Launch Site`, `Lat(Latitude)`, `Long(Longitude)`, `class`
spacex_df = spacex_df[['Launch Site', 'Lat', 'Long', 'class']]
launch_sites_df = spacex_df.groupby(['Launch Site'], as_index=False).first()
launch_sites_df = launch_sites_df[['Launch Site', 'Lat', 'Long']]
launch_sites_df
```

Above coordinates are just plain numbers that can not give you any intuitive insights about where are those launch sites. If you are very good at geography, you can interpret those numbers directly in your mind. If not, that's fine too. Let's visualize those locations by pinning them on a map.

We first need to create a folium `Map` object, with an initial center location to be NASA Johnson Space Center at Houston, Texas.

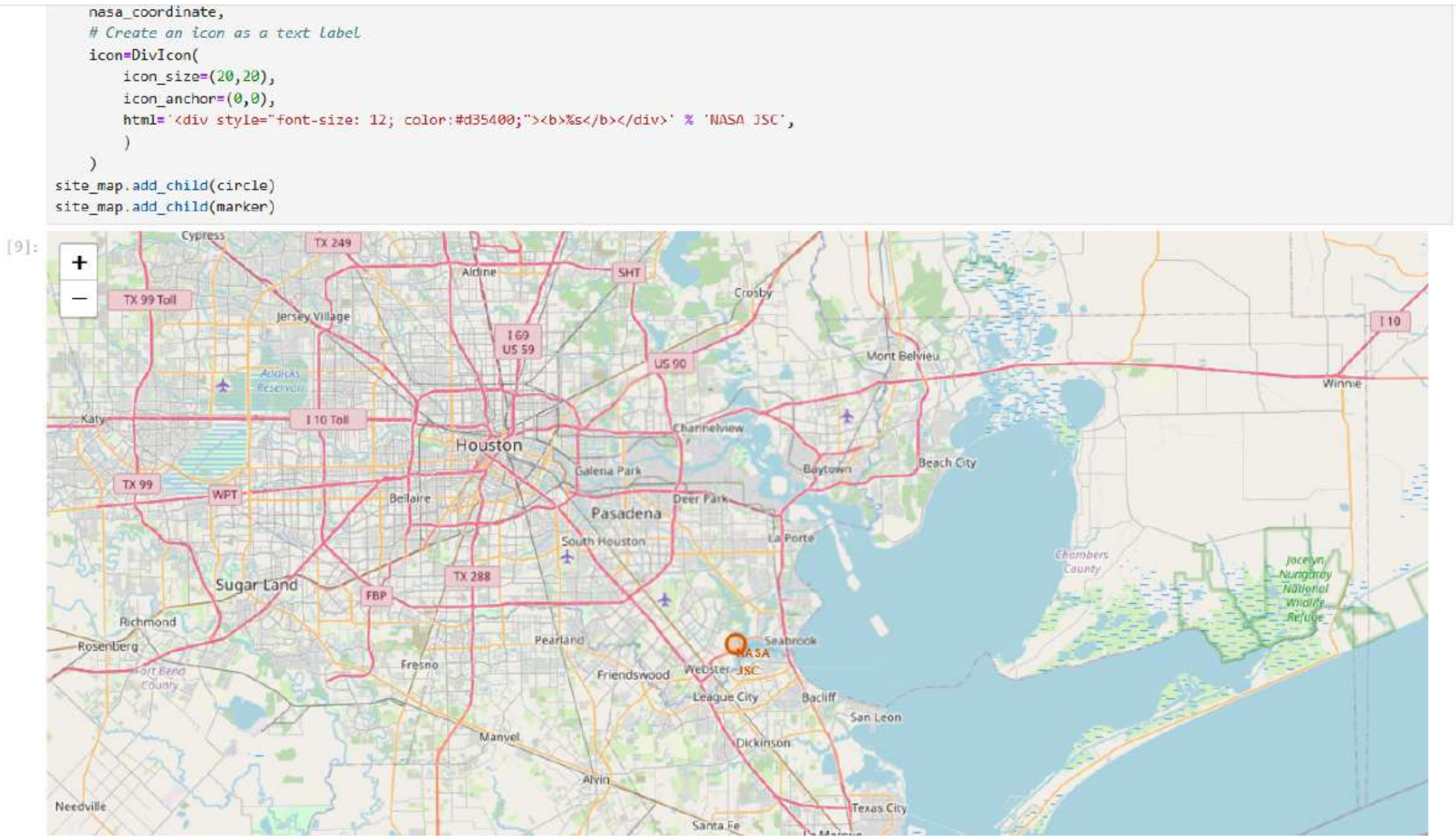
[8]:

```python
# Start location is NASA Johnson Space Center
nasa_coordinate = [29.559684888503615, -95.0830971930759]
site_map = folium.Map(location=nasa_coordinate, zoom_start=10)
```

We could use `folium.Circle` to add a highlighted circle area with a text label on a specific coordinate. For example,

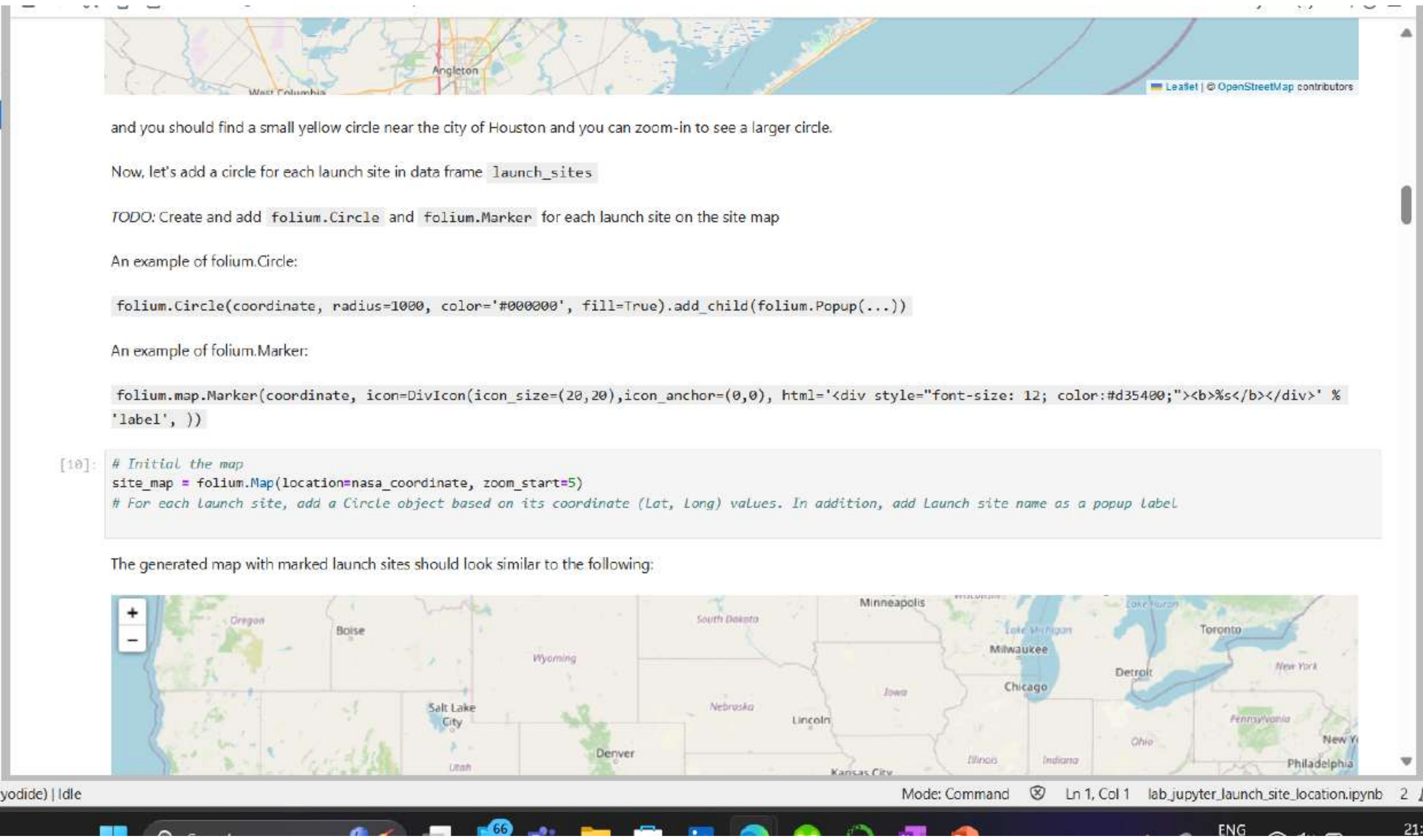
[9]:

```python
# Create a blue circle at NASA Johnson Space Center's coordinate with a popup label showing its name
circle = folium.Circle(nasa_coordinate, radius=1000, color='#d35400', fill=True).add_child(folium.Popup('NASA Johnson Space Center'))
# Create a blue circle at NASA Johnson Space Center's coordinate with a icon showing its name
marker = folium.map.Marker(
    nasa_coordinate,
    # Create an icon as a text label
    icon=DivIcon(
        icon_size=(20,20),
```



```
    nasa_coordinate,
    # Create an icon as a text label
    icon=DivIcon(
        icon_size=(20,20),
        icon_anchor=(0,0),
        html='<div style="font-size: 12; color:#d35400;"><b>%s</b></div>' % 'NASA JSC',
        )
    )
site_map.add_child(circle)
site_map.add_child(marker)
```

[9]:



and you should find a small yellow circle near the city of Houston and you can zoom-in to see a larger circle.

Now, let's add a circle for each launch site in data frame `launch_sites`

*TODO:* Create and add `folium.Circle` and `folium.Marker` for each launch site on the site map

An example of folium.Circle:

`folium.Circle(coordinate, radius=1000, color='#000000', fill=True).add_child(folium.Popup(...))`

An example of folium.Marker:

`folium.map.Marker(coordinate, icon=DivIcon(icon_size=(20,20),icon_anchor=(0,0), html='<div style="font-size: 12; color:#d35400;"><b>%s</b></div>' % 'label', ))`

```
[10]: # Initial the map
      site_map = folium.Map(location=nasa_coordinate, zoom_start=5)
      # For each launch site, add a Circle object based on its coordinate (Lat, Long) values. In addition, add Launch site name as a popup label
```

The generated map with marked launch sites should look similar to the following:

yodide) | Idle

Mode: Command  Ln 1, Col 1  lab_jupyter_launch_site_location.ipynb  2

[15]:
```
spacex_df.tail(10)
```

[15]:

| | Launch Site | Lat | Long | class |
|---|---|---|---|---|
| **46** | KSC LC-39A | 28.573255 | -80.646895 | 1 |
| **47** | KSC LC-39A | 28.573255 | -80.646895 | 1 |
| **48** | KSC LC-39A | 28.573255 | -80.646895 | 1 |
| **49** | CCAFS SLC-40 | 28.563197 | -80.576820 | 1 |
| **50** | CCAFS SLC-40 | 28.563197 | -80.576820 | 1 |
| **51** | CCAFS SLC-40 | 28.563197 | -80.576820 | 0 |
| **52** | CCAFS SLC-40 | 28.563197 | -80.576820 | 0 |
| **53** | CCAFS SLC-40 | 28.563197 | -80.576820 | 0 |
| **54** | CCAFS SLC-40 | 28.563197 | -80.576820 | 1 |
| **55** | CCAFS SLC-40 | 28.563197 | -80.576820 | 0 |

Next, let's create markers for all launch records. If a launch was successful `(class=1)`, then we use a green marker and if a launch was failed, we use a red marker `(class=0)`

Note that a launch only happens in one of the four launch sites, which means many launch records will have the exact same coordinate. Marker clusters can be a good way to simplify a map containing many markers having the same coordinate.

Let's first create a `MarkerCluster` object

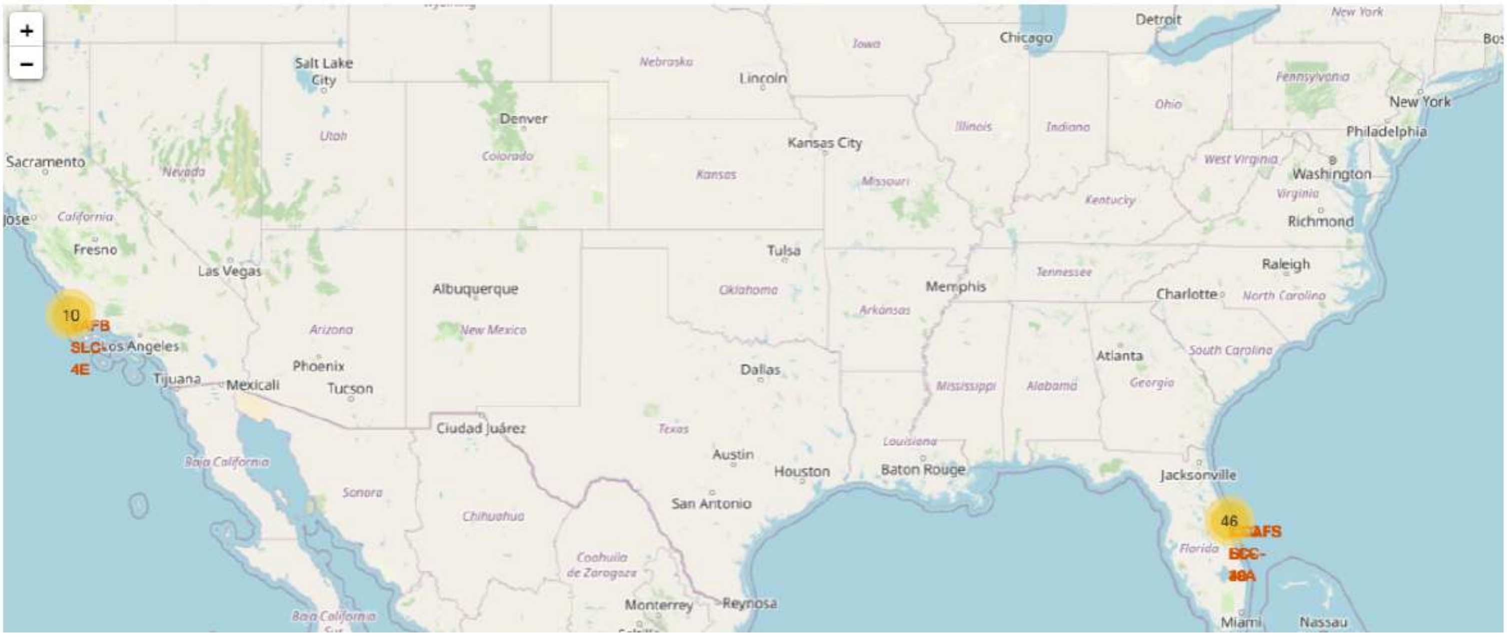
[16]:
```
marker_cluster = MarkerCluster()
```

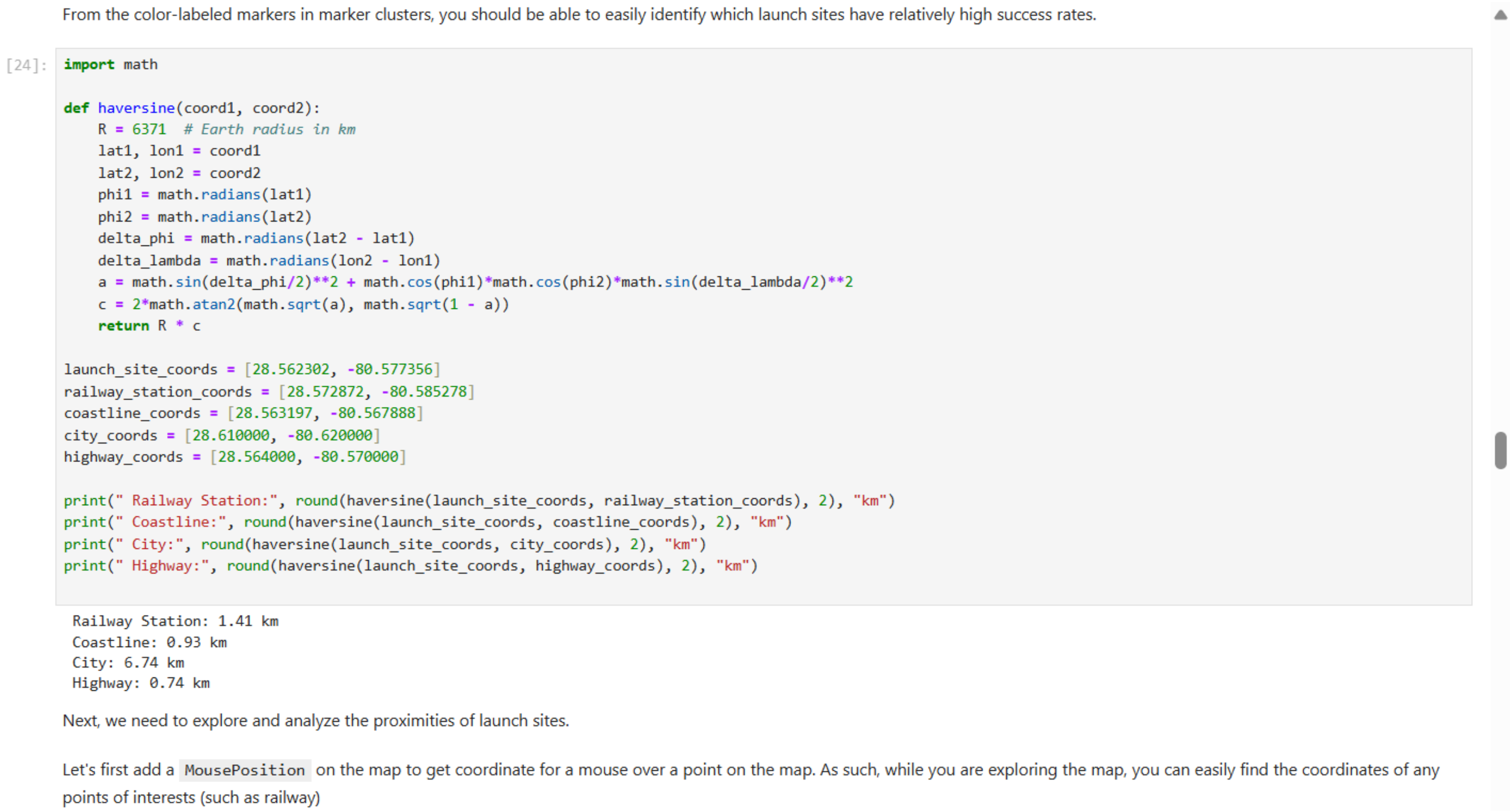
*TODO:* Create a new column in `spacex_df` dataframe called `marker_color` to store the marker colors based on the `class` value

[17]:
```
spacex_df['marker_color'] = spacex_df['class'].apply(lambda x: 'green' if x == 1 else 'red')

# Apply a function to check the value of `class` column
```

Leaflet | Map tiles by Stamen Design, under CC BY 3.0. Data by OpenStreetMap, under ODbL.

Your updated map may look like the following screenshots:





From the color-labeled markers in marker clusters, you should be able to easily identify which launch sites have relatively high success rates.

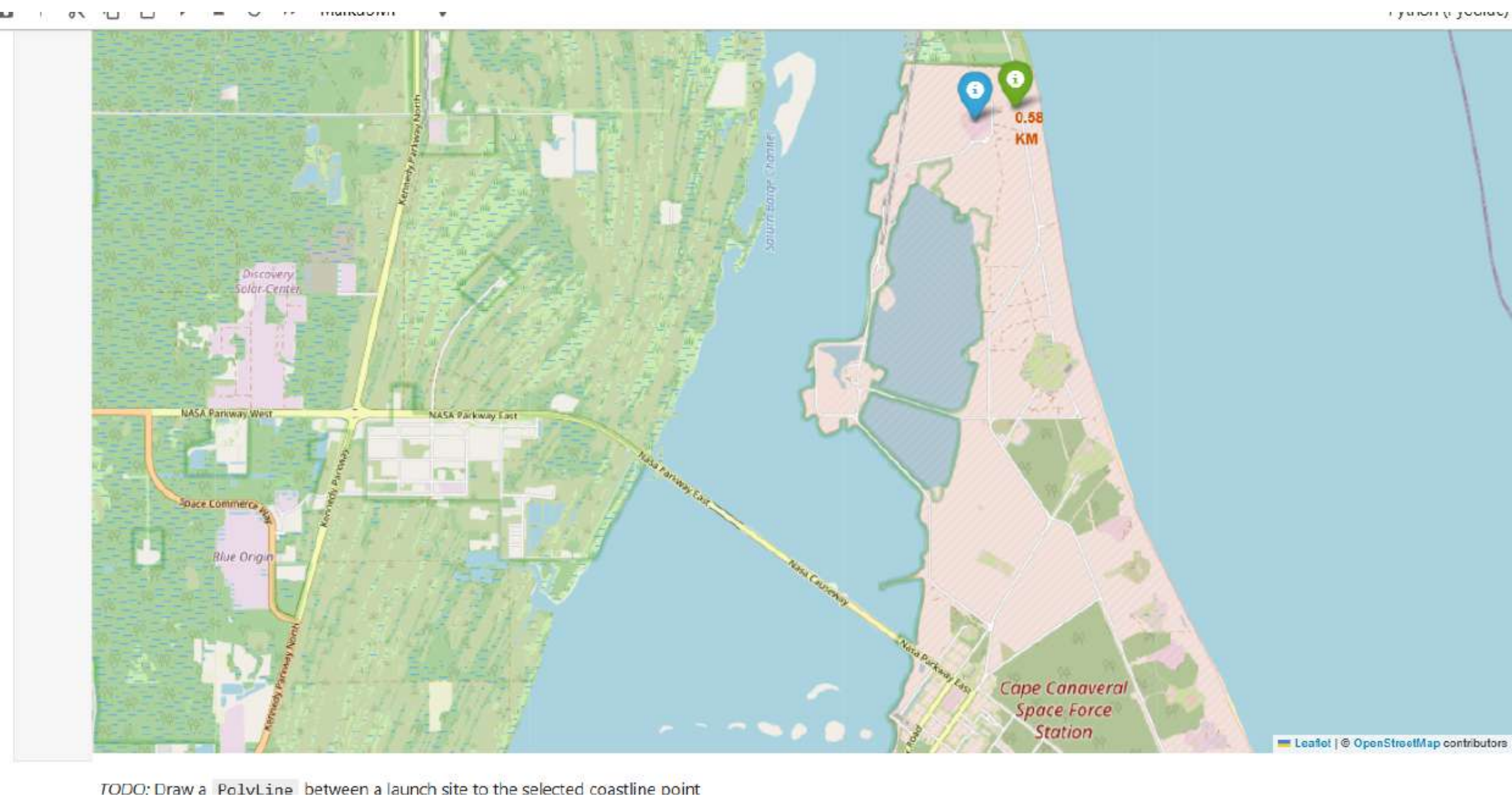
[24]:

```
import math

def haversine(coord1, coord2):
    R = 6371  # Earth radius in km
    lat1, lon1 = coord1
    lat2, lon2 = coord2
    phi1 = math.radians(lat1)
    phi2 = math.radians(lat2)
    delta_phi = math.radians(lat2 - lat1)
    delta_lambda = math.radians(lon2 - lon1)
    a = math.sin(delta_phi/2)**2 + math.cos(phi1)*math.cos(phi2)*math.sin(delta_lambda/2)**2
    c = 2*math.atan2(math.sqrt(a), math.sqrt(1 - a))
    return R * c

launch_site_coords = [28.562302, -80.577356]
railway_station_coords = [28.572872, -80.585278]
coastline_coords = [28.563197, -80.567888]
city_coords = [28.610000, -80.620000]
highway_coords = [28.564000, -80.570000]

print(" Railway Station:", round(haversine(launch_site_coords, railway_station_coords), 2), "km")
print(" Coastline:", round(haversine(launch_site_coords, coastline_coords), 2), "km")
print(" City:", round(haversine(launch_site_coords, city_coords), 2), "km")
print(" Highway:", round(haversine(launch_site_coords, highway_coords), 2), "km")
```

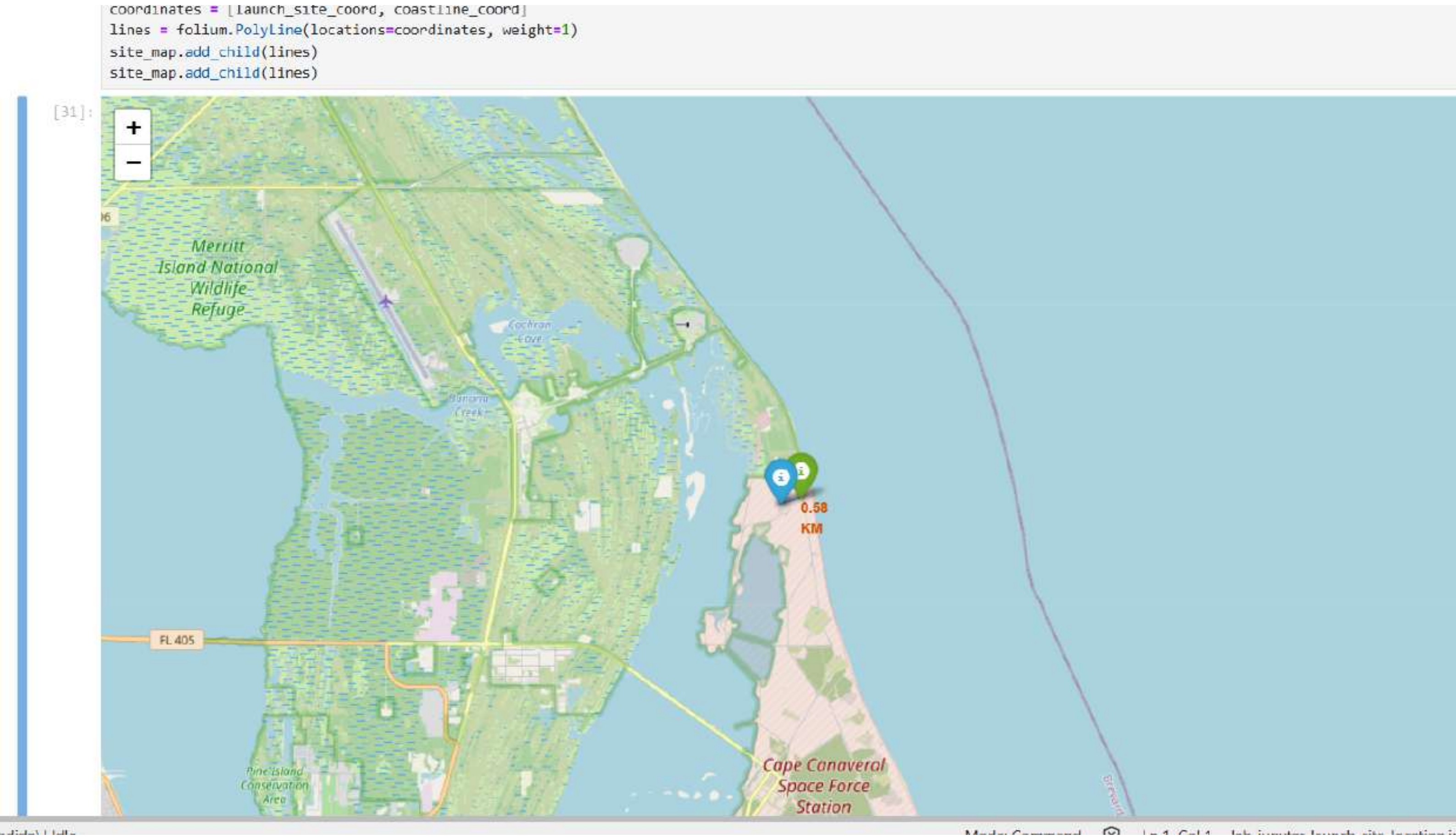
```
 Railway Station: 1.41 km
 Coastline: 0.93 km
 City: 6.74 km
 Highway: 0.74 km
```

Next, we need to explore and analyze the proximities of launch sites.

Let's first add a `MousePosition` on the map to get coordinate for a mouse over a point on the map. As such, while you are exploring the map, you can easily find the coordinates of any points of interests (such as railway)



*TODO:* Draw a `PolyLine` between a launch site to the selected coastline point



```
coordinates = [launch_site_coord, coastline_coord]
lines = folium.PolyLine(locations=coordinates, weight=1)
site_map.add_child(lines)
site_map.add_child(lines)
```

[31]:
+
−
Merritt Island National Wildlife Refuge
Cochran Cove
Banana Creek
0.58 KM
FL 405
Pine Island Conservation Area
Cape Canaveral Space Force Station
Brevard