```
#!/usr/bin/env python
# coding: utf-8

# In[2]:


get_ipython().run_line_magic('matplotlib', 'inline')
from IPython.display import HTML
from matplotlib import rcParams


# In[46]:


#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Mon Nov 19 15:43:32 2018

@author: vegaj
"""
#https://www.rideindego.com/about/data/
import pickle
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.animation as animation
import pandas as pd
import math
from mpl_toolkits.basemap import Basemap

pickle_path = 'bike.pkl'


## uncomment the following if you wish to change the pd dataframe.
## due to the long processing time of this, I saved the dataframe in a pickle and am l
oading it
## directly from there.


#tells read_csv to acknowledge these columns as dates
parse_dates = ['start_time', 'end_time']

df = pd.read_csv(open("indego.csv", "rb"), delimiter=",", parse_dates=parse_dates)

#loads all the unique bike ids
bike_ids = (df['bike_id'].unique())
print(len(bike_ids)/15)
bike_ids = bike_ids[:(int(len(bike_ids)/20))]
from datetime import timedelta, date

#creates a time range(in minutes) between two date points
def daterange(start_date, end_date):
    date_list = []
    for n in range(int ((end_date - start_date).total_seconds()//60)):
        date_list.append( start_date + timedelta(minutes=n))
    return date_list

#reads the start and the end time for the data
start = df['start_time'].min()
end = df['end_time'].max()
date_range=(daterange(start,end)[:(30*24*60)])

#finds the boundaries of the map for all the bike locations
max_y = df['start_lon'].max()
min_y = df['start_lon'].min()
max_x = df['start_lat'].max()
min_x = df['start_lat'].min()

#creates a dataframe of only a index of all the minutes in the time range
```

```
path = pd.DataFrame(index=date_range)

print(len(bike_ids))
#creates a column for every unique bike
#Therefore creating a dataframe of minute timestamps x bike ids
for bike in bike_ids:
    path[bike] = pd.np.empty((len(path), 0)).tolist()


def vector(d,t):
    time_delta = d.duration
    distance = math.sqrt((d.end_lon-d.start_lon)**2 +
                         (d.end_lat-d.start_lat)**2)
    dis_y = d.end_lon-d.start_lon
    dis_x = d.end_lat-d.start_lat
    vx = dis_x/time_delta
    vy = dis_y/time_delta
    v = distance/time_delta
    vty = d.start_lon + vy*t
    vtx = d.start_lat + vx*t
    return vtx,vty







# In[47]:


for dtype in ['float','int','object']:
    selected_dtype = path.select_dtypes(include=[dtype])
    mean_usage_b = selected_dtype.memory_usage(deep=True).mean()
    mean_usage_mb = mean_usage_b / 1024 ** 2
    print("Average memory usage for {} columns: {:03.2f} MB".format(dtype,mean_usage_m
b))


# In[48]:


#changes the main dataframe to only include the bikes that are in unique bike list
#this will shorten the df to lessen iterations
print(len(bike_ids))
df= (df.loc[df['bike_id'].isin(bike_ids)])


# In[49]:


print(df.keys())
df = df[['duration','start_time','end_time','start_lat','start_lon','end_lat','end_lon
','bike_id']]
df.dropna()


# In[58]:



#for stop/start line in each unique bike
#check the start then the end, then calulate the positions
#for each time interval
#then place each of these positions and time intervals into thepath dataframe

##This checks every row of the MAIN dataframe and starts to set data in the path dataf
rame
for row in df.itertuples():
```

```
    end = row.end_time
    start = row.start_time
    time_range = daterange(start_date=start, end_date=end)
    bike_id = row.bike_id
    #Find the ride time
    duration = row.duration
    for i in range(0,int(duration)):
        x,y = vector(row,i)
        try:
            if not path.at[time_range[i], bike_id]:
                print(path.at[time_range[i], bike_id])
                path.at[time_range[i], bike_id] = [round(x,6),round(y,6),1]
        except KeyError:
            print('keyError')


# In[59]:


path


# In[60]:


for column in path:
    counter=0
    for index, item in path[column].iteritems():
        if not(item):
            if counter!=0:
                x,y,idle = path[column].iloc[counter-1]
                path[column].iloc[counter]= [x,y,idle+1]
            else:
                path[column].iloc[counter]=[0,0,0]

        counter+=1

path.to_pickle(pickle_path)


# In[61]:



path[[2531]].loc['2018-07-30 23:48:00']


# In[87]:


path = pd.read_pickle(pickle_path)
fig =plt.figure(figsize=(25,25))
#fig, ax = plt.subplots()

# setup Lambert Conformal basemap.
m = Basemap(width=10000,height=7000,projection='lcc',
            resolution='c',lat_0=39.960819,lon_0=-75.164924)
#https://www.opendataphilly.org/dataset/street-centerlines/resource/46c57a03-bf20-44b4
-b897-2d44afbdad29
m.readshapefile('../proj_3/streets/Street_Centerline','philly')

df = pd.read_csv(open("indego.csv", "rb"), delimiter=",")
station_name = df['start_station'].sort_values().unique()
station_lat = (df['start_lat'][station_name])
station_lon = (df['start_lon'][station_name])

scatter = m.scatter([],[], latlon=True)
ax = plt.axes()
ttl = ax.text(0, 1.05, '', transform = ax.transAxes, va='center', fontsize = 30)
```

```
ax.text(0, 1.10, 'Time Lapse Movement of 68 Indego Bikes over 2 Days in Philadelphia.'
, transform = ax.transAxes, va='center', fontsize = 30)
ax.text(0, -.05,'All Indigo bike stations are in purple. The bikes are red circles. As
 they traverse the map, they have a uniform size.\n As they sit idle, their radius inc
reases to point out lapses in usage.',transform = ax.transAxes, fontsize = 30)
fig.subplots_adjust(left=0, bottom=0, right=1, top=1, wspace=None, hspace=None)
#initializes the first plot to be blitted over
def init():
    x,y = m(station_lon.values,station_lat.values)
    scatter = m.scatter(x,y, marker='D', color='m')
    return scatter,

def animate(i):
    #Finds row i and changes the pd.Series into a list
    #This will be a list of lists.
    index_label = path.index[i]
    path_list = path.iloc[i].tolist()
    #Finds the first item of each sublist
    x = [round(item[0],6)for item in path_list]
    y = [round(item[1],6) for item in path_list]
    idle = [item[2] for item in path_list]
    #The coordinate system switches here.
    #m() is important to chnage the values into basemap values
    x, y = m(y,x)
    idle = np.asarray(idle)
    idle =np.clip(idle,1,60)
    ##IMPORTANT
    data = np.vstack([x,y]).T
    scatter.set_linewidth(10)
    scatter.set_facecolor(c='none')
    scatter.set_edgecolor(c='red')
    scatter.set_sizes(idle*10)
    scatter.set_offsets(data)
    ttl.set_text(index_label)
    return scatter,

Writer = animation.writers['ffmpeg']
writer = Writer(fps=15, bitrate=1800)

#Here, frames is how many frames I want the video to have.
#interval is how long I want the frames to stay in millisec.
#Therefore the frames has to be how many rows(minute) are in the dataframe
#and interval has to be long enough so that all the frames can be made into a movie of
 a specific length
ani = animation.FuncAnimation(fig, animate, init_func=init, frames=3000,
                              interval=50, blit=True)
ani.save('ani.mp4', codec = "libx264")


# In[ ]:




# In[ ]:




# In[ ]:
```