In [1]:

```
%matplotlib inline
from IPython.display import HTML
from matplotlib import rcParams
import pickle
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.animation as animation
import pandas as pd
import os
os.environ['PROJ_LIB'] = r'C:\Users\John\Anaconda3\pkgs\proj4-5.2.0-ha925a31_1\L
import math
from mpl_toolkits.basemap import Basemap
```

In [2]:

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

@author: vegaj
"""
#https://www.rideindego.com/about/data/

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 an
## directly from there.


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

df = pd.read_csv(open("indego-trips-2019-q3.csv", "rb"), delimiter=",", parse_dat


###ADJUST VARIABLES HERE###

#how many days the build should go
days = 2
#Adjust number of bikes (Therefore columns) program will iterate through
divisor = 3


#loads all the unique bike ids
bike_ids = (df['bike_id'].unique())

bike_ids = bike_ids[:(int(len(bike_ids)/divisor))]
number_of_bikes = int(len(bike_ids)/divisor)
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)+1):
        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()

print('Maximum number of total bikes in data: ',len(bike_ids))
print('Number of bikes used for graphic: ', number_of_bikes)
#creates a list of every minute between amount of days and start date
minute_range = days*24*60
date_range=(daterange(start,end)[:(minute_range)])

#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)

#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()


#This funciton uses pythagorean theorem to determine the x,y coordinates of
#Where d(bike info) is at any given time (t)
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
```

```
Maximum number of total bikes in data:  510
Number of bikes used for graphic:  170
```

In [3]:

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

```
510
```

In [4]:
```
df = df[['duration','start_time','end_time','start_lat','start_lon','end_lat','e
df.dropna()
```

Out[4]:

	duration	start_time	end_time	start_lat	start_lon	end_lat	end_lon	bike_id	t
0	30	2019-07-01 00:01:00	2019-07-01 00:31:00	39.945091	-75.142502	39.974140	-75.180222	11901	3200
1	26	2019-07-01 00:04:00	2019-07-01 00:30:00	39.966740	-75.207993	39.958660	-75.213226	16519	3200
2	10	2019-07-01 00:04:00	2019-07-01 00:14:00	39.930820	-75.174744	39.940182	-75.154419	2729	3200
3	10	2019-07-01 00:04:00	2019-07-01 00:14:00	39.930820	-75.174744	39.940182	-75.154419	2603	3200
4	15	2019-07-01 00:05:00	2019-07-01 00:20:00	39.962891	-75.166061	39.945171	-75.159927	11868	3200
...	...	...	...	...	...	...	...	...	
275186	9	2019-09-30 23:46:00	2019-09-30 23:55:00	39.980049	-75.155220	39.962650	-75.161743	16340	3267
275187	6	2019-09-30 23:47:00	2019-09-30 23:53:00	39.951969	-75.179428	39.945950	-75.184753	11897	3267
275189	15	2019-09-30 23:50:00	2019-10-01 00:05:00	39.964390	-75.179871	39.964390	-75.179871	11911	3267
275191	7	2019-09-30 23:51:00	2019-09-30 23:58:00	39.967590	-75.179520	39.967178	-75.161247	11798	3267
275193	14	2019-09-30 23:52:00	2019-10-01 00:06:00	39.964390	-75.179871	39.964390	-75.179871	5194	3267

97251 rows × 9 columns

In [5]:

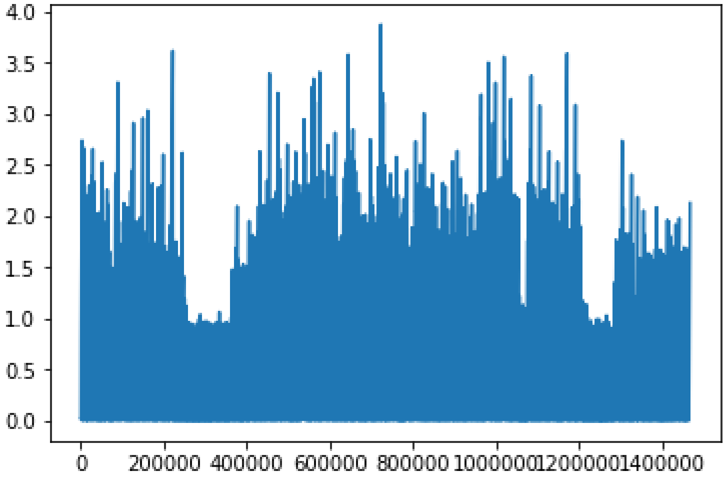
```
#for stop/start line in each unique bike
#check the start then the end, then calulate the positions
#for each time interval
#then place each of these positions and time intervals into thepath dataframe

##This checks every row of the MAIN dataframe and starts to set data in the path
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,len(time_range)):
        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:
            pass
```

In [6]:

```
from datetime import datetime
alg_times = []
for column in path:
        counter=0
        start = datetime.now()
        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]
            end  = datetime.now()
            alg_times.append(end-start)
            counter+=1
```

In [7]:
```
#Checks iteration time for above algorithm for curiosity
alg_times = [i.total_seconds() for i in alg_times]
plt.plot(range(len(alg_times)),(alg_times))
```

Out[7]:
```
[<matplotlib.lines.Line2D at 0x155055f3cc0>]
```

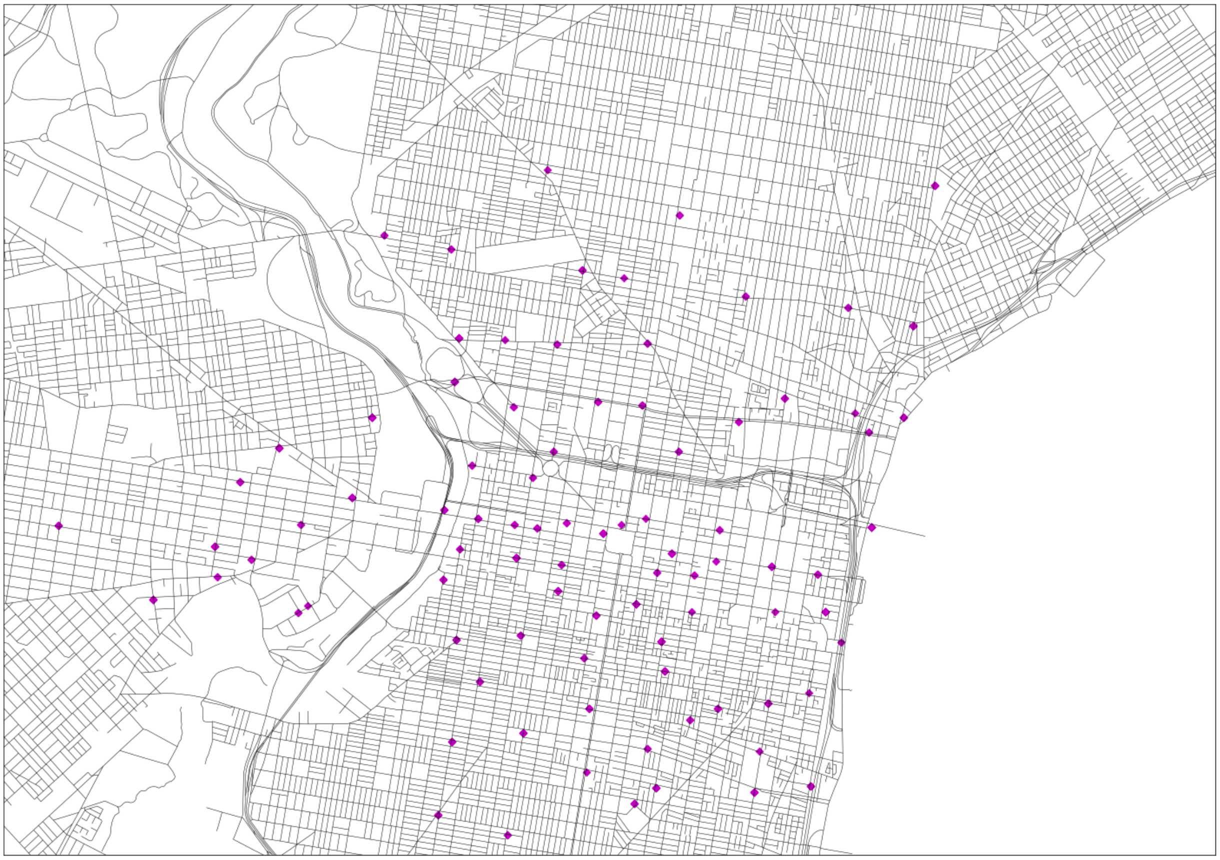


In [8]:
```
df = pd.read_csv(open("indego-trips-2019-q3.csv", "rb"), delimiter=",")

#Takes All unique Values from both Start stations and End Station columns and
#Merges them Vertically.
station_name = df['start_station'].sort_values().unique()
station_lat = (df['start_lat'][station_name])
station_lon = (df['start_lon'][station_name])
station_name_end = df['end_station'].sort_values().unique()
station_lat_end = (df['end_lat'][station_name_end])
station_lon_end = (df['end_lon'][station_name_end])
station_lat_end.rename(columns={"end_lat": "start_lat"})
station_lon_end.rename(columns={"end_lat": "start_lat"})
station_lat = pd.concat([station_lat,station_lat_end])
station_lon = pd.concat([station_lon,station_lon_end])

all_stat = np.concatenate((station_name,station_name_end))
```

In [9]:

```

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-bf2
m.readshapefile('../proj_3/streets/Street_Centerline','philly')


x,y = m(station_lon.values,station_lat.values)
scatter = m.scatter(x,y, marker='D', color='m')
print(len(x))
```

```
C:\Users\John\Anaconda3\envs\tense\lib\site-packages\ipykernel_launcher.py:6: M
atplotlibDeprecationWarning:
The dedent function was deprecated in Matplotlib 3.1 and will be removed in 3.
3. Use inspect.cleandoc instead.

C:\Users\John\Anaconda3\envs\tense\lib\site-packages\ipykernel_launcher.py:8: M
atplotlibDeprecationWarning:
The dedent function was deprecated in Matplotlib 3.1 and will be removed in 3.
3. Use inspect.cleandoc instead.



278
```



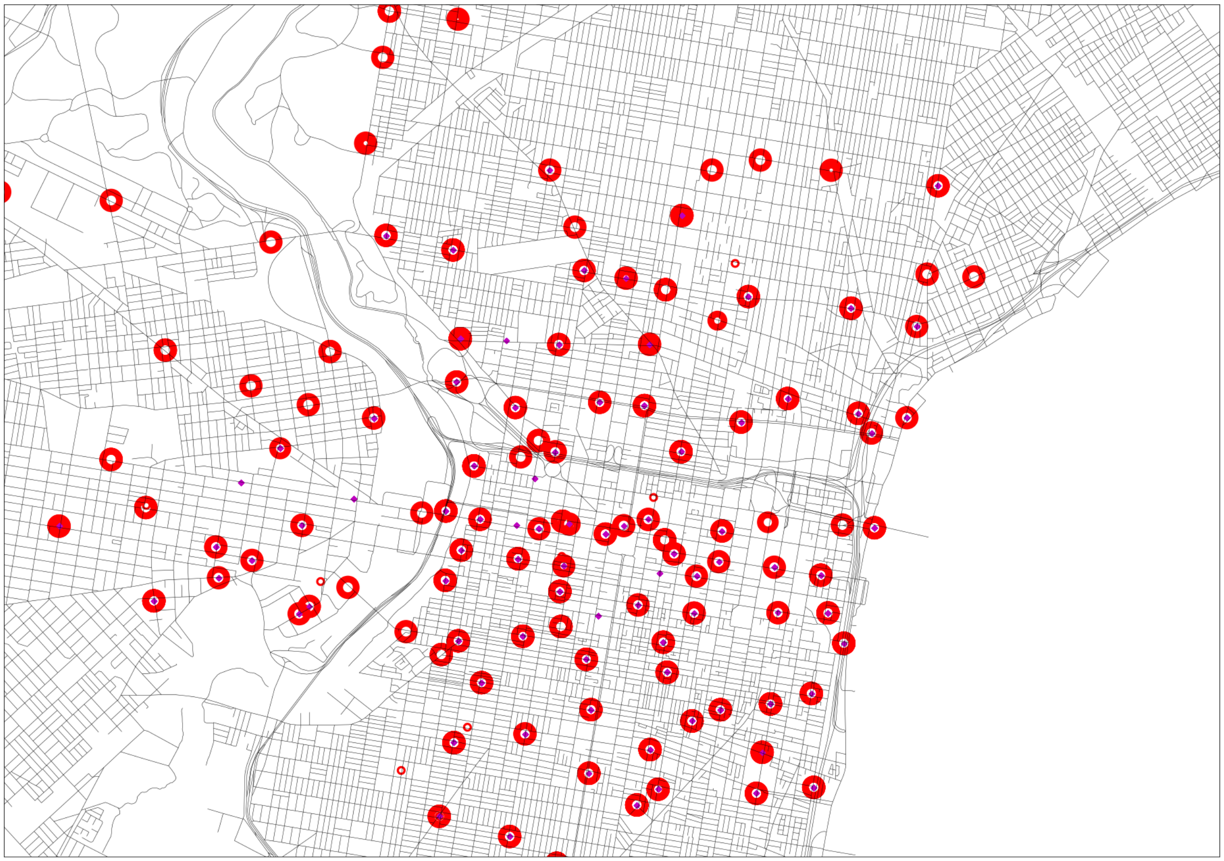
In [10]:
```
#path = pd.read_pickle(pickle_path)

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-bf2
m.readshapefile('../proj_3/streets/Street_Centerline','philly')


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 {} Indego Bikes over {} Days in Philadel
        transform = ax.transAxes, va='center', fontsize = 30)

ax.text(0, -.05,'All Indigo bike stations are in purple. The bikes are red circle
a uniform size.\n As they sit idle, their radius increases to point out lapses i
        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 mo
ani = animation.FuncAnimation(fig, animate, init_func=init, frames=minute_range-
                              interval=60, blit=True)
ani.save('ani.mp4')
```

```
C:\Users\John\Anaconda3\envs\tense\lib\site-packages\ipykernel_launcher.py:8: M
atplotlibDeprecationWarning:
The dedent function was deprecated in Matplotlib 3.1 and will be removed in 3.
3. Use inspect.cleandoc instead.

C:\Users\John\Anaconda3\envs\tense\lib\site-packages\ipykernel_launcher.py:10:
MatplotlibDeprecationWarning:
The dedent function was deprecated in Matplotlib 3.1 and will be removed in 3.
3. Use inspect.cleandoc instead.
  # Remove the CWD from sys.path while we load stuff.
C:\Users\John\Anaconda3\envs\tense\lib\site-packages\ipykernel_launcher.py:14:
MatplotlibDeprecationWarning: Adding an axes using the same arguments as a prev
ious axes currently reuses the earlier instance.  In a future version, a new in
stance will always be created and returned.  Meanwhile, this warning can be sup
pressed, and the future behavior ensured, by passing a unique label to each axe
s instance.
```

Time Lapse Movement of 170 Indego Bikes over 2 Days in Philadelphia.

2019-07-02 23:59:00



All Indigo bike stations are in purple. The bikes are red circles. As they traverse the map, they have a uniform size.
As they sit idle, their radius increases to point out lapses in usage.