# Assignment 11: Open Source Python Geospatial Libraries

due: Nov 22, 11:45 am

**1) Task:**

To use the Shapely and Geopandas modules to create and modify geometry objects and to handle basic GIS operations in a Python notebook.

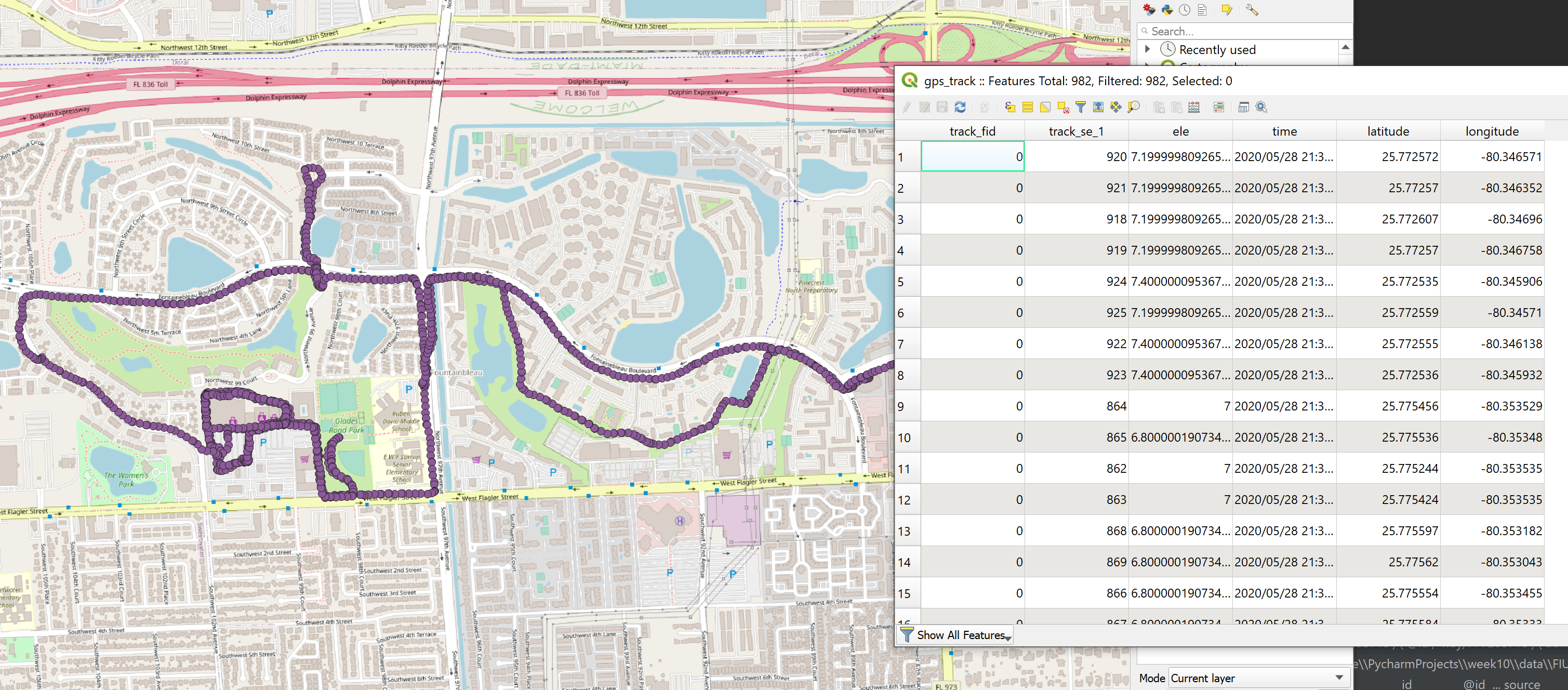
**2) Data and analysis:**

The handout files come with a comma separated text file containing GPS point measurements of a running activity. Each point contains geographic coordinates (WGS84), elevation and a timestamp. In a new notebook create a Python class called ***ActivityHandle***r() that contains the following methods:

* **build\_path():** This method has one parameter (*points*) which denote coordinate pairs. It returns a Shapely LineString geometry and prints the number of points used to create the geometry.
* **calculate\_stats():** The method has three parameters (*line, start\_time, end\_time*), where *line* is a Shapely LineString, and *start\_time* and *end\_time* are the beginning and end of the entire activity. The method returns a Python dictionary that reports the average speed [mi/h], average pace [min/mi] and distance covered [mi] over the entire activity. The function also prints these statistics.
* **get\_centroid():** The method takes any kind of Shapely geometry object as an input and returns its centroid as a Shapely geometry. The method also prints the centroid coordinates of the geometry.
* **export\_activity():** The method takes two parameters (*path, stats*), where *path* is a Shapely LineString and *stats* is a Python dictionary which contains the metrics calculated with *calculate\_stats()*. The function creates a Geopandas GeoDataFrame and saves it as an ESRI Shapefile with corresponding attributes.

**3) Illustration:**

The figure below overlays the given trackpoints on a standard background map and shows a subset of the attribute table. Your notebook will convert these points to a line and save it as an ESRI Shapefile.



**4) Directions:**

1. Use a standard Python approach to read input data from a text file. CSV files are comma separated text files where each row contains a record.
2. Use an appropriate projected coordinate system for distance measurements and calculations.
3. A LineString is an ordered list of points. Make sure to use the correct sequence of points.
4. Create a new instance of the **ActivityHandler** class and demonstrate the functionality of each method by calling them, i.e. build a LineString from input point, calculate metrics, demonstrate the get\_centroid() method, etc.
5. Besides lecture notes, you can also consult Web resources, e.g., by searching for “How to do XY in Geopandas?” or “How to iterate over lines of a CSV in Python”.
6. Note: if you have issues reading the CSV file using a relative path, you can use the absolute path instead.

Structure your program as follows:

**import**...  
...  
**class** ActivityHandler():  
 **def** \_\_init\_\_(self):  
 x = 1  
  
 **def** build\_path(self, points):  
 *# build a line using Linestring from shapely* **return**...  
  
 **def** calculate\_stats(self, line, start\_time, end\_time):  
 *# Calculate trip duration in minutes  
 # Project linestring to calculate distance in meters, then convert to miles  
 # Use geopandas GeoSeries* line\_geoseries = gpd.GeoSeries(line, crs=**'epsg:4326'**)  
  
 *# create an empty geopandas dataframe with dist, pace, avg. speed column names  
 # update the geopandas dataframe with values from the trip statistics* **return**...  
  
 **def** get\_centroid(self, geometry):  
 **return** ...  
  
 **def** export\_activity(self, path, stats):  
 *# export geodataframe created to shapefile* **return  
  
if** \_\_name\_\_ == **'\_\_main\_\_'**:  
 *# Create lists to store coordinate pairs and timestamps* coordinate\_list = []  
 timestamps = []  
  
 *# Open CSV file to iterate through rows to extract longitude, latitude and time  
 # Create an instance of ActivityHandler()  
 # build a shapely Linestring  
  
 # Calculate trip statistics  
# Get the centroid of trip. This line will print out geographic coordinates  
# Export activity as Shapefile*

**5) Program execution:**

a) Run your notebook

b) Insert a screen capture of the relevant notebook portions after calling the methods above to show that these methods print out the required items listed in the task description. Insert a screen capture

A close up of a message

Description automatically generated

c) Visualize the resulting ESRI Shapefile in the software of your choice, showing both the line geometry and the attribute table. Insert a screen capture

A map with a green line

Description automatically generated

Hand in:

A zip file containing:

1) This MS Word document with the inserted screen captures.

2) A Python notebook (.ipynb) with your code.

Good luck!