1. Geocoder.geocode
   1. Ceates an datafile instance (from read input)
2. Readinput.Datafile
   1. Checks if a file was passed in if not gets one via UI
   2. Controller()
      1. Checks file type and sends file to appropriate data type class (csv, xlsx, xls)
         1. Datatype class reads in data and separates the headers
   3. tabs\_with\_intersection\_min
      1. creates a list of tabs that have headers with the intersection pattern and stores in intersection attribute
   4. tabs\_with\_stretch\_min
      1. creates a list of tabs that have headers with the intersection pattern and stores in stretch attribute
   5. tabs\_with\_addresses\_min
      1. creates a list of tabs that have headers with the intersection pattern and stores in address attribute
   6. parse address
      1. checks for address column and then splits into number and street and calls tabs\_with\_addresses\_min
   7. tabs\_with\_nodes\_min
      1. checks for nodeid in the columns – stores in nodes attribute
   8. User selects mapping type
      1. ‘1’ – calls: intersection()
      2. ‘2’ – calls: address()
      3. ‘3’ – calls: block()
      4. ‘4’ – calls: nodes()
3. Geocoder.block
   1. Prints length of tabs with block pattern
   2. Adds segmentid, xf, yf, xt, yt, error columns to intersection tab headers
   3. Cleans up data type of borough column
   4. BlockStretch ()
      1. Creates the block stretch instance by either of the following:
         1. ‘ON’,’FROM’,’TO’
         2. ‘ON STREET’, ‘FROM STREET’, ‘TO STREET’
         3. ‘STREET 1’, ‘STREET 2’, ‘STREET 3’
   5. get\_segmentids
      1. block\_stretch\_request
         1. Calls boro\_format
            1. Standardizes borough name and stores full name borough in borough attribute
         2. Calls clean\_up
            1. Standardizes OnStreet, CrossStreetOne and CrossStreetTwo for api call
         3. Sends request to api to geocode location without direction
         4. Stores result of of geocode in BlockStretch.json
      2. Calls first\_error to update BlockStretch.json
         1. Calls direction\_binary on the from intersection and resends to api
         2. Calls direction\_binary on the to intersection and resends to api
      3. Checks that geocode results contain blockfacelist or if they have an error.
      4. Populates storage with either result or error message attributes
   6. Stores valid segments from api result
      1. Calls segment\_validity
         1. Checks that every character is a number
   7. get\_segment\_geom
      1. Lat, Long, From, To are returned in json so going to shapefile uneccessary
      2. Goes to lion shapefileand gets coordinates for segments
   8. Updates the data file with the attributes from geocode
   9. Shapify.Lines() - Creates shapefile of segments
      1. Cleans up shapefile name
      2. Delete previous version of shapefile if exists
      3. Creates shapefile
      4. Cleans the shapefile fields
         1. Tries to guess what the datatype of each column is
         2. Cleans up naming conventions
         3. Add fields to shapefile
      5. Adds data to newly added fields