Meeting #6

10/14/21

Omar Luna

Deliverables

- Parsing livedata.json file

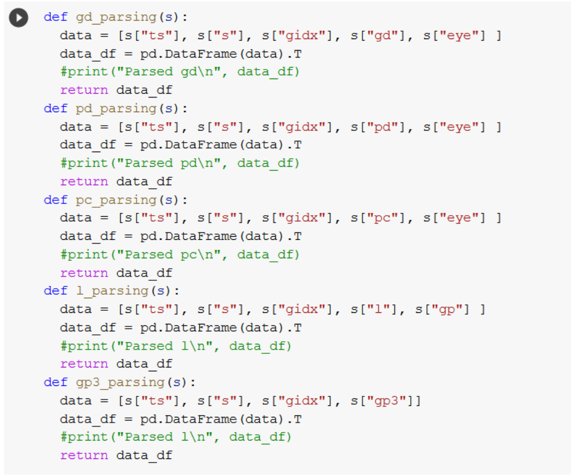
Methodology and Learnings

- Modified participant.json file parse
- Opened file and read line by line
- Parsed dictionary into list by finding the column in the line
- Appended new dataframe row to seperated dataframes

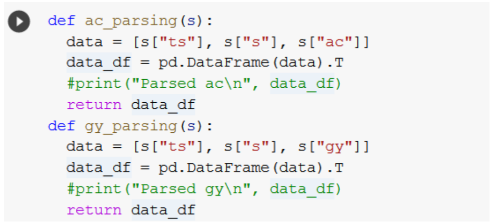
Learn

- Json file syntax affects output

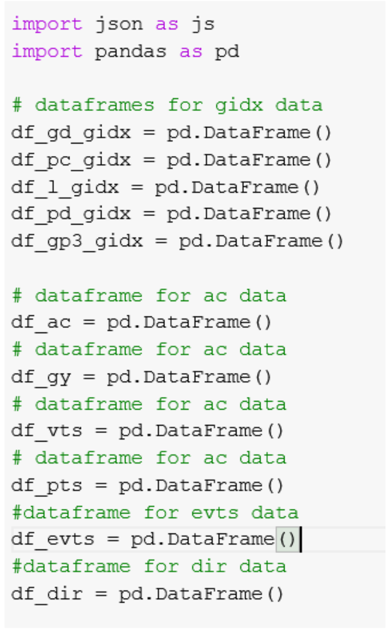
Results



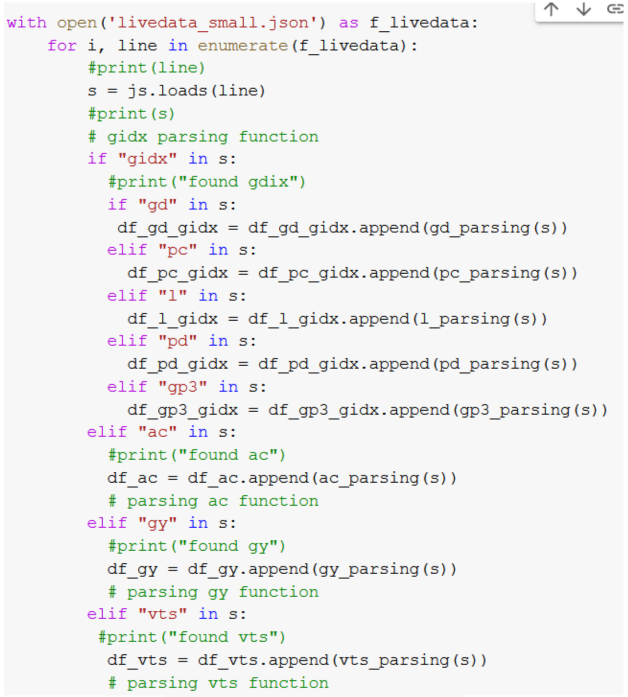
```python
def gd_parsing(s):
  data = [s["ts"], s["s"], s["gidx"], s["gd"], s["eye"] ]
  data_df = pd.DataFrame(data).T
  #print("Parsed gd\n", data_df)
  return data_df
def pd_parsing(s):
  data = [s["ts"], s["s"], s["gidx"], s["pd"], s["eye"] ]
  data_df = pd.DataFrame(data).T
  #print("Parsed pd\n", data_df)
  return data_df
def pc_parsing(s):
  data = [s["ts"], s["s"], s["gidx"], s["pc"], s["eye"] ]
  data_df = pd.DataFrame(data).T
  #print("Parsed pc\n", data_df)
  return data_df
def l_parsing(s):
  data = [s["ts"], s["s"], s["gidx"], s["l"], s["gp"] ]
  data_df = pd.DataFrame(data).T
  #print("Parsed l\n", data_df)
  return data_df
def gp3_parsing(s):
  data = [s["ts"], s["s"], s["gidx"], s["gp3"]]
  data_df = pd.DataFrame(data).T
  #print("Parsed l\n", data_df)
  return data_df
```



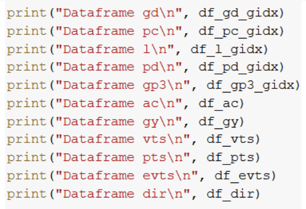
```python
def ac_parsing(s):
  data = [s["ts"], s["s"], s["ac"]]
  data_df = pd.DataFrame(data).T
  #print("Parsed ac\n", data_df)
  return data_df
def gy_parsing(s):
  data = [s["ts"], s["s"], s["gy"]]
  data_df = pd.DataFrame(data).T
  #print("Parsed gy\n", data_df)
  return data_df
```



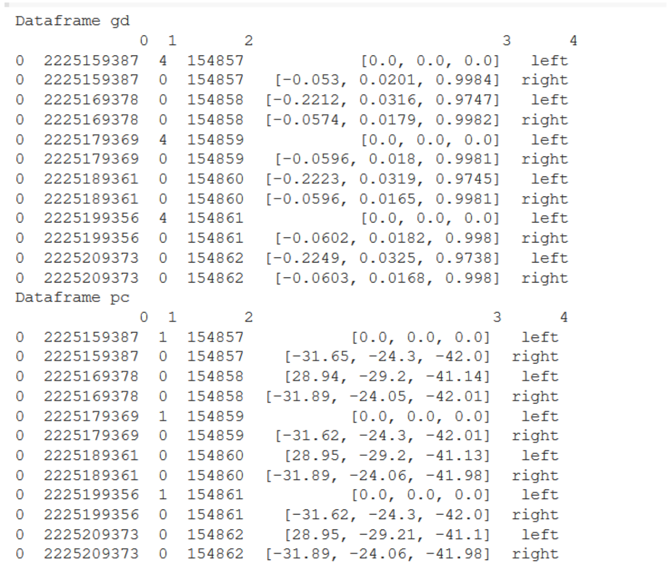
```python
import json as js
import pandas as pd

# dataframes for gidx data
df_gd_gidx = pd.DataFrame()
df_pc_gidx = pd.DataFrame()
df_l_gidx = pd.DataFrame()
df_pd_gidx = pd.DataFrame()
df_gp3_gidx = pd.DataFrame()

# dataframe for ac data
df_ac = pd.DataFrame()
# dataframe for ac data
df_gy = pd.DataFrame()
# dataframe for ac data
df_vts = pd.DataFrame()
# dataframe for ac data
df_pts = pd.DataFrame()
#dataframe for evts data
df_evts = pd.DataFrame()
#dataframe for dir data
df_dir = pd.DataFrame()
```



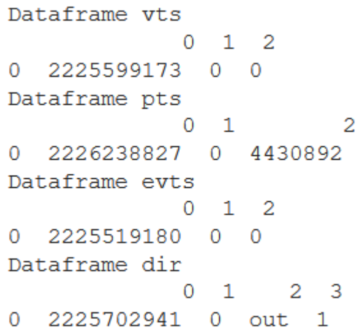
```python
with open('livedata_small.json') as f_livedata:
    for i, line in enumerate(f_livedata):
        #print(line)
        s = js.loads(line)
        #print(s)
        # gidx parsing function
        if "gidx" in s:
          #print("found gdix")
          if "gd" in s:
           df_gd_gidx = df_gd_gidx.append(gd_parsing(s))
          elif "pc" in s:
            df_pc_gidx = df_pc_gidx.append(pc_parsing(s))
          elif "l" in s:
            df_l_gidx = df_l_gidx.append(l_parsing(s))
          elif "pd" in s:
            df_pd_gidx = df_pd_gidx.append(pd_parsing(s))
          elif "gp3" in s:
            df_gp3_gidx = df_gp3_gidx.append(gp3_parsing(s))
        elif "ac" in s:
          #print("found ac")
          df_ac = df_ac.append(ac_parsing(s))
          # parsing ac function
        elif "gy" in s:
          #print("found gy")
          df_gy = df_gy.append(gy_parsing(s))
          # parsing gy function
        elif "vts" in s:
         #print("found vts")
          df_vts = df_vts.append(vts_parsing(s))
          # parsing vts function
```



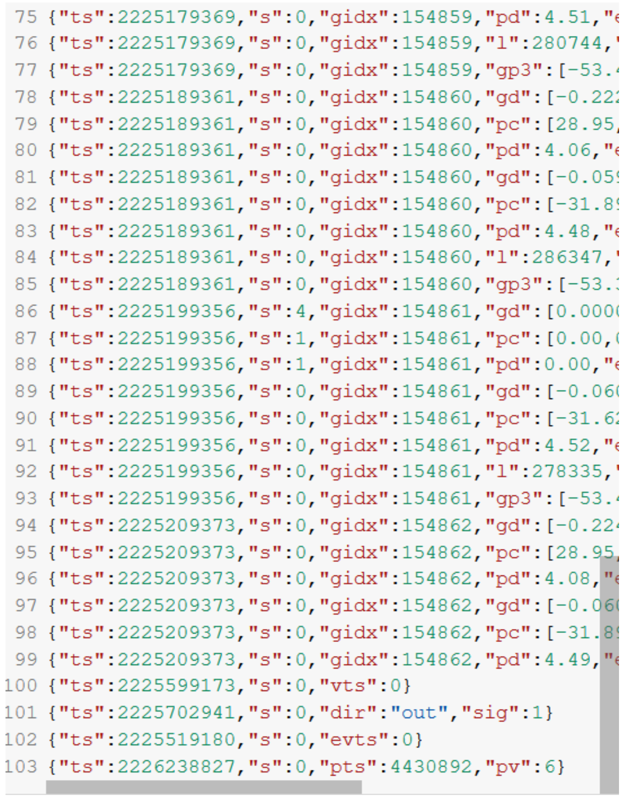
```python
print("Dataframe gd\n", df_gd_gidx)
print("Dataframe pc\n", df_pc_gidx)
print("Dataframe l\n", df_l_gidx)
print("Dataframe pd\n", df_pd_gidx)
print("Dataframe gp3\n", df_gp3_gidx)
print("Dataframe ac\n", df_ac)
print("Dataframe gy\n", df_gy)
print("Dataframe vts\n", df_vts)
print("Dataframe pts\n", df_pts)
print("Dataframe evts\n", df_evts)
print("Dataframe dir\n", df_dir)
```



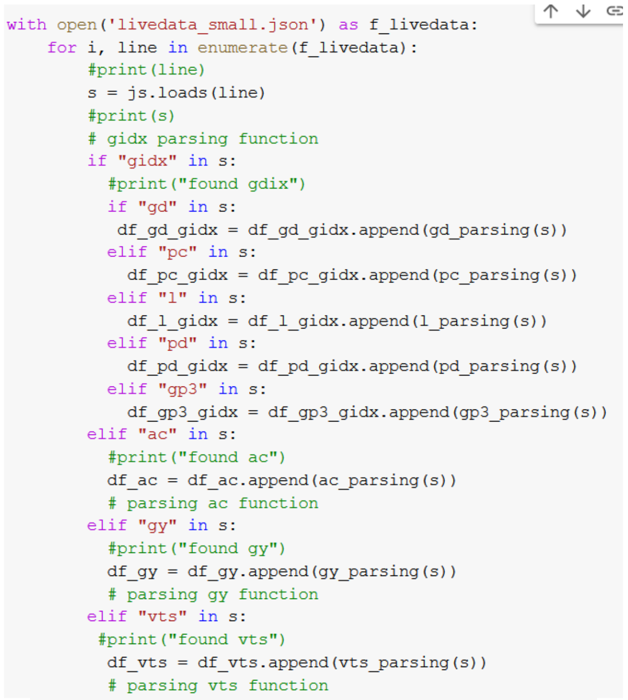
```
Dataframe gd
            0  1       2                          3      4
0  2225159387  4  154857           [0.0, 0.0, 0.0]   left
0  2225159387  0  154857   [-0.053, 0.0201, 0.9984]  right
0  2225169378  0  154858  [-0.2212, 0.0316, 0.9747]   left
0  2225169378  0  154858  [-0.0574, 0.0179, 0.9982]  right
0  2225179369  4  154859           [0.0, 0.0, 0.0]   left
0  2225179369  0  154859   [-0.0596, 0.018, 0.9981]  right
0  2225189361  0  154860  [-0.2223, 0.0319, 0.9745]   left
0  2225189361  0  154860  [-0.0596, 0.0165, 0.9981]  right
0  2225199356  4  154861           [0.0, 0.0, 0.0]   left
0  2225199356  0  154861   [-0.0602, 0.0182, 0.998]  right
0  2225209373  0  154862  [-0.2249, 0.0325, 0.9738]   left
0  2225209373  0  154862   [-0.0603, 0.0168, 0.998]  right
Dataframe pc
            0  1       2                         3      4
0  2225159387  1  154857          [0.0, 0.0, 0.0]   left
0  2225159387  0  154857    [-31.65, -24.3, -42.0]  right
0  2225169378  0  154858    [28.94, -29.2, -41.14]   left
0  2225169378  0  154858  [-31.89, -24.05, -42.01]  right
0  2225179369  1  154859          [0.0, 0.0, 0.0]   left
0  2225179369  0  154859   [-31.62, -24.3, -42.01]  right
0  2225189361  0  154860    [28.95, -29.2, -41.13]   left
0  2225189361  0  154860  [-31.89, -24.06, -41.98]  right
0  2225199356  1  154861          [0.0, 0.0, 0.0]   left
0  2225199356  0  154861    [-31.62, -24.3, -42.0]  right
0  2225209373  0  154862    [28.95, -29.21, -41.1]   left
0  2225209373  0  154862  [-31.89, -24.06, -41.98]  right
```



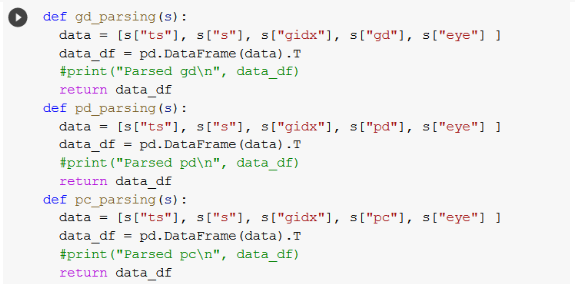
```
Dataframe vts
            0  1  2
0  2225599173  0  0
Dataframe pts
            0  1        2
0  2226238827  0  4430892
Dataframe evts
            0  1  2
0  2225519180  0  0
Dataframe dir
            0  1    2  3
0  2225702941  0  out  1
```



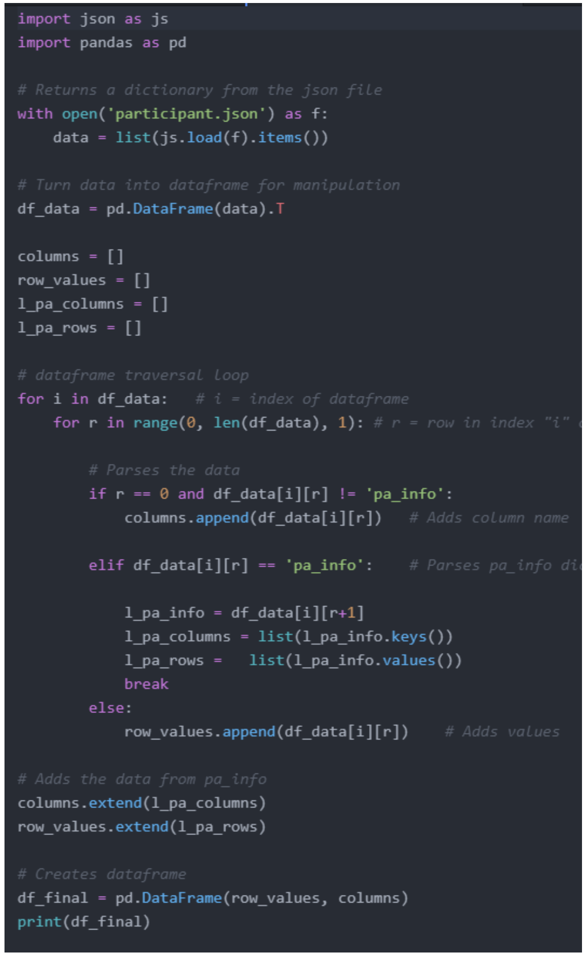
```
{"ts":2225179369,"s":0,"gidx":154859,"pd":4.51,"
{"ts":2225179369,"s":0,"gidx":154859,"l":280744,
{"ts":2225179369,"s":0,"gidx":154859,"gp3":[-53.
{"ts":2225189361,"s":0,"gidx":154860,"gd":[-0.22
{"ts":2225189361,"s":0,"gidx":154860,"pc":[28.95
{"ts":2225189361,"s":0,"gidx":154860,"pd":4.06,"
{"ts":2225189361,"s":0,"gidx":154860,"gd":[-0.05
{"ts":2225189361,"s":0,"gidx":154860,"pc":[-31.8
{"ts":2225189361,"s":0,"gidx":154860,"pd":4.48,"
{"ts":2225189361,"s":0,"gidx":154860,"l":286347,
{"ts":2225189361,"s":0,"gidx":154860,"gp3":[-53.
{"ts":2225199356,"s":4,"gidx":154861,"gd":[0.000
{"ts":2225199356,"s":1,"gidx":154861,"pc":[0.00,
{"ts":2225199356,"s":1,"gidx":154861,"pd":0.00,"
{"ts":2225199356,"s":0,"gidx":154861,"gd":[-0.06
{"ts":2225199356,"s":0,"gidx":154861,"pc":[-31.6
{"ts":2225199356,"s":0,"gidx":154861,"pd":4.52,"
{"ts":2225199356,"s":0,"gidx":154861,"l":278335,
{"ts":2225199356,"s":0,"gidx":154861,"gp3":[-53.
{"ts":2225209373,"s":0,"gidx":154862,"gd":[-0.22
{"ts":2225209373,"s":0,"gidx":154862,"pc":[28.95
{"ts":2225209373,"s":0,"gidx":154862,"pd":4.08,"
{"ts":2225209373,"s":0,"gidx":154862,"gd":[-0.06
{"ts":2225209373,"s":0,"gidx":154862,"pc":[-31.8
{"ts":2225209373,"s":0,"gidx":154862,"pd":4.49,"
{"ts":2225599173,"s":0,"vts":0}
{"ts":2225702941,"s":0,"dir":"out","sig":1}
{"ts":2225519180,"s":0,"evts":0}
{"ts":2226238827,"s":0,"pts":4430892,"pv":6}
```



```python
with open('livedata_small.json') as f_livedata:
    for i, line in enumerate(f_livedata):
        #print(line)
        s = js.loads(line)
        #print(s)
        # gidx parsing function
        if "gidx" in s:
          #print("found gdix")
          if "gd" in s:
           df_gd_gidx = df_gd_gidx.append(gd_parsing(s))
          elif "pc" in s:
            df_pc_gidx = df_pc_gidx.append(pc_parsing(s))
          elif "l" in s:
            df_l_gidx = df_l_gidx.append(l_parsing(s))
          elif "pd" in s:
            df_pd_gidx = df_pd_gidx.append(pd_parsing(s))
          elif "gp3" in s:
            df_gp3_gidx = df_gp3_gidx.append(gp3_parsing(s))
        elif "ac" in s:
          #print("found ac")
          df_ac = df_ac.append(ac_parsing(s))
          # parsing ac function
        elif "gy" in s:
          #print("found gy")
          df_gy = df_gy.append(gy_parsing(s))
          # parsing gy function
        elif "vts" in s:
         #print("found vts")
          df_vts = df_vts.append(vts_parsing(s))
          # parsing vts function
```



```python
def gd_parsing(s):
  data = [s["ts"], s["s"], s["gidx"], s["gd"], s["eye"] ]
  data_df = pd.DataFrame(data).T
  #print("Parsed gd\n", data_df)
  return data_df
def pd_parsing(s):
  data = [s["ts"], s["s"], s["gidx"], s["pd"], s["eye"] ]
  data_df = pd.DataFrame(data).T
  #print("Parsed pd\n", data_df)
  return data_df
def pc_parsing(s):
  data = [s["ts"], s["s"], s["gidx"], s["pc"], s["eye"] ]
  data_df = pd.DataFrame(data).T
  #print("Parsed pc\n", data_df)
  return data_df
```



```python
import json as js
import pandas as pd

# Returns a dictionary from the json file
with open('participant.json') as f:
    data = list(js.load(f).items())

# Turn data into dataframe for manipulation
df_data = pd.DataFrame(data).T

columns = []
row_values = []
l_pa_columns = []
l_pa_rows = []

# dataframe traversal loop
for i in df_data:   # i = index of dataframe
    for r in range(0, len(df_data), 1): # r = row in index "i"

        # Parses the data
        if r == 0 and df_data[i][r] != 'pa_info':
            columns.append(df_data[i][r])   # Adds column name

        elif df_data[i][r] == 'pa_info':    # Parses pa_info di

            l_pa_info = df_data[i][r+1]
            l_pa_columns = list(l_pa_info.keys())
            l_pa_rows =  list(l_pa_info.values())
            break
        else:
            row_values.append(df_data[i][r])    # Adds values

# Adds the data from pa_info
columns.extend(l_pa_columns)
row_values.extend(l_pa_rows)

# Creates dataframe
df_final = pd.DataFrame(row_values, columns)
print(df_final)
```