DES: CODE DOCUMENTATION

Dashboard

Project: Simulate Sensor Data and Dashboard

Team Number: SP2 #28

CLIENT: DR.GEORG.GROSSMANN

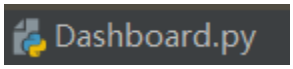
TEAM MEMBER: YUXUAN LI, MUNYARADZI MAGURA, YIFEI RAN.

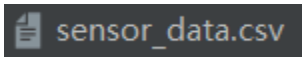
Contents

Overview:

This document includes two big parts: How to run and Code explanation, in Code explanation it has imports: show all the imports I used, functions: what can each function do, and full code. This document is used to describe and introduce the functions in the code of the Dashboard section.

HOW TO RUN:

0) Run code: open Dashboard.py in PyCharm and run.
1) 

2) Sensor_Data.csv get from DES part.
3) 

Code explanation:

Imports

```
import dash
import dash_table
import dash_core_components as dcc
import dash_html_components as html
import pandas as pd
import csv
```

dash: is used to data analysis and visualization
dash_table: is used to make dash table
dash_core_components: is used to make web-based interactive components
dash_html_components: is used to make html components
pandas: is used to get and manipulate data in csv files
csv: is used to manipulate data in csv files

Functions

a) Open the csv file generated in the DES section

```
df = pd.read_csv('sensor_data.csv')
```

b) This function is used to get the number of sensorID, because if we want to make the code dynamic not hard code, my plan is get the number of sensorID, then use for loop to traversal each sensor's value.

```
sensorID = df['sensorID']
sensorID_list = sensorID.values.tolist()  # change the value of sensorID
to list to operate on it
list_num = []
for i in sensorID_list:  # use for loop to get all different value in
sensorID
    if i in list_num:
        continue
    else:
        list_num.append(i)  # put the value into list
sensorID_num = list_num[len(list_num) - 1]  # get the last number of
list_num
sensorID_num = int(sensorID_num)  # change it to int format
```

c) This function use dictionary to storage data, in this way, we can use it during making line graph and bar chart.

```
csv_dictionary = {}
with open('sensor_data.csv', 'r') as file:
    reader = csv.reader(file)
    for row in reader:
        # print(row[1])
        time = row[1]
        temp = row[2]
        vib = row[3]
        if str(row[0]) in csv_dictionary:
            # add time
            csv_dictionary[str(row[0])][0].append(time)
            # add temperature
            csv_dictionary[str(row[0])][1].append(temp)
            # add vibration
            csv_dictionary[str(row[0])][2].append(vib)
        else:
            csv_dictionary[str(row[0])] = [[time], [temp], [vib]]
```

d) This function is used to set figure outside, so just need to calling this function in dcc.graph. In this way, we use for loop to set x,y value and name for each line we will get. Whatever how many data we will get, all the data will show on the line graph.

```
data_line = []
for i in range(0, sensorID_num + 1):
    name_temp = 'sensor_' + str(i + 1) + '_temp'  # because i is 0 at
begin, so use i+1. Set the name for each temp line
    name_vib = 'sensor_' + str(i + 1) + '_vib'  # set the name for each
temp line
    # set x,y value and name for each line we will get
    data_line.append({'x': csv_dictionary[str(i)][0], 'y':
csv_dictionary[str(i)][1],
                      'type': 'line', 'name': name_temp})
    data_line.append({'x': csv_dictionary[str(i)][0], 'y':
csv_dictionary[str(i)][2],
                      'type': 'line', 'name': name_vib})
figure_line_graph = {'data': data_line,
                     'layout': {
                         'title': 'Line Graph',  # the title of the line
graph
                     },
                     }
```

e) Same as last function, this function is used to set x,y value and name for each bar we will get.

```
data_bar = []
for i in range(0, sensorID_num + 1):
    name_temp = 'sensor_' + str(i + 1) + '_temp'  # set the name for each
temp bar
    name_vib = 'sensor_' + str(i + 1) + '_vib'  # set the name for each
vib bar
    # set x,y value and name for each bar we will get
    data_bar.append({'x': csv_dictionary[str(i)][0], 'y':
csv_dictionary[str(i)][1],
                     'type': 'bar', 'name': name_temp})
    data_bar.append({'x': csv_dictionary[str(i)][0], 'y':
csv_dictionary[str(i)][2],
                     'type': 'bar', 'name': name_vib})
```

```
figure_barchart = {'data': data_bar,
                   'layout': {
                       'title': 'Bar Chart',  # the title of the bar chart
                   },
                   }
```

f) Make a title 'Dashboard' for html

```
app.layout = html.Div([
html.Div([
    html.H3('Dashboard')
]),
```

g) Set line graph and bar chart

```
html.Div([
    html.Br(),

    dcc.Graph(id='line garph',
              figure=figure_line_graph  # calling figure_line_graph
              ),

    dcc.Graph(id='bar chart',
              figure=figure_barchart,  # calling figure_barchart
              ),
```

h) Define the detail of the datable, include editing the cells, all data is passed to the table up-front or not ('none'), number of rows visible per page, filtering by column, align text columns to left. By default they are aligned to right.
Because the title of some columns are so long, so I make the width big.

```
dash_table.DataTable(
    id='datatable-advanced-filtering',
    columns=[
        {'name': i, 'id': i, 'deletable': True} for i in df.columns
        # omit the id column
        if i != 'id'
    ],
    data=df.to_dict('records'),
    editable=True, #editing the cells
    page_action='native', # all data is passed to the table up-front or
not ('none')
    page_size=10,   # number of rows visible per page
    filter_action="native", #filtering by column
    style_cell={
        'textAlign':'left', # align text columns to left. By default they
are aligned to right
        'minWidth': 210,'maxWidth': 210,'Width': 210,
        # couse some of the texts are too long so I make the width big
        'backgroundColor':'white'
    },
),
```

Full code

```
import dash
import dash_table
import dash_core_components as dcc
import dash_html_components as html
import pandas as pd
import csv

df = pd.read_csv('sensor_data.csv')  # open the csv file to get data
# this is used to get the final number of sensorID, in this way can get
the whole number of sensors that used
sensorID = df['sensorID']
sensorID_list = sensorID.values.tolist()  # change the value of sensorID
to list to operate on it
list_num = []
for i in sensorID_list:  # use for loop to get all different value in
sensorID
    if i in list_num:
        continue
    else:
        list_num.append(i)  # put the value into list
sensorID_num = list_num[len(list_num) - 1]  # get the last number of
list_num
sensorID_num = int(sensorID_num)  # change it to int format

# use dictionary to storage data in this way can use it during making line
graph and bar chart
csv_dictionary = {}
with open('sensor_data.csv', 'r') as file:
    reader = csv.reader(file)
    for row in reader:
        # print(row[1])
        time = row[1]
        temp = row[2]
        vib = row[3]
        if str(row[0]) in csv_dictionary:
            # add time
            csv_dictionary[str(row[0])][0].append(time)
            # add temperature
            csv_dictionary[str(row[0])][1].append(temp)
            # add vibration
            csv_dictionary[str(row[0])][2].append(vib)
        else:
            csv_dictionary[str(row[0])] = [[time], [temp], [vib]]

app = dash.Dash(__name__)

# write the figure outside, so just need to calling this function in
dcc.graph
# figure for line graph
data_line = []
for i in range(0, sensorID_num + 1):
    name_temp = 'sensor_' + str(i + 1) + '_temp'  # because i is 0 at
```

```
begin, so use i+1. Set the name for each temp line
    name_vib = 'sensor_' + str(i + 1) + '_vib'  # set the name for each
temp line
    # set x,y value and name for each line we will get
    data_line.append({'x': csv_dictionary[str(i)][0], 'y':
csv_dictionary[str(i)][1],
                      'type': 'line', 'name': name_temp})
    data_line.append({'x': csv_dictionary[str(i)][0], 'y':
csv_dictionary[str(i)][2],
                      'type': 'line', 'name': name_vib})
figure_line_graph = {'data': data_line,
                     'layout': {
                         'title': 'Line Graph',  # the title of the line
graph
                     },
                     }

# figure for bar chart
data_bar = []
for i in range(0, sensorID_num + 1):
    name_temp = 'sensor_' + str(i + 1) + '_temp'  # set the name for each
temp bar
    name_vib = 'sensor_' + str(i + 1) + '_vib'  # set the name for each
vib bar
    # set x,y value and name for each bar we will get
    data_bar.append({'x': csv_dictionary[str(i)][0], 'y':
csv_dictionary[str(i)][1],
                     'type': 'bar', 'name': name_temp})
    data_bar.append({'x': csv_dictionary[str(i)][0], 'y':
csv_dictionary[str(i)][2],
                     'type': 'bar', 'name': name_vib})
figure_barchart = {'data': data_bar,
                   'layout': {
                       'title': 'Bar Chart',  # the title of the bar chart
                   },
                   }

app.layout = html.Div([
    html.Div([
        html.H3('Dashboard')
    ]),
    # make a title for dashboard

    html.Div([
        html.Br(),

        dcc.Graph(id='line garph',
                  figure=figure_line_graph  # calling figure_line_graph
                  ),

        dcc.Graph(id='bar chart',
                  figure=figure_barchart,  # calling figure_barchart
                  ),

        # define the detail of datatable
        dash_table.DataTable(
            id='datatable-advanced-filtering',
```

```
            columns=[
                {'name': i, 'id': i, 'deletable': True} for i in
df.columns
                # omit the id column
                if i != 'id'
            ],
            data=df.to_dict('records'),
            editable=True,  # editing the cells
            page_action='native',  # all data is passed to the table up-
front or not ('none')
            page_size=10,  # number of rows visible per page
            filter_action="native",  # filtering by column
            style_cell={
                'textAlign': 'left',  # align text columns to left. By
default they are aligned to right
                'minWidth': 210, 'maxWidth': 210, 'Width': 210,
                # cause some of the texts are too long so I make the width
big
                'backgroundColor': 'white'
            },
        ),
        html.Hr()
    ])
])

if __name__ == '__main__':
    app.run_server(debug=True)
```