Visualizing Volcanoes on Earth

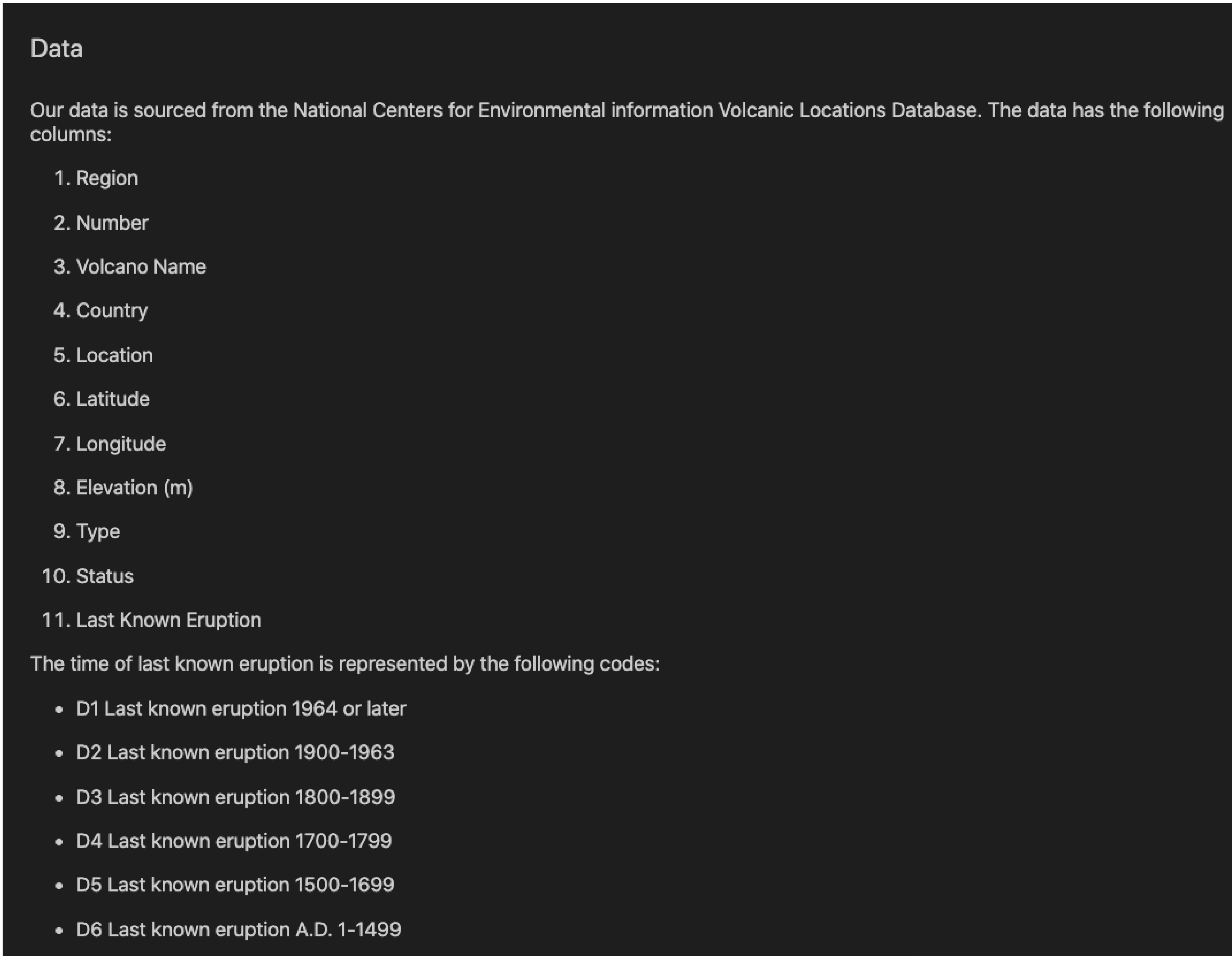
Team Members and Contributions:

Ji Yun Kim: Visualizations 3, 5, 6, 8

Jack Gordon: Visualizations 1, 2, 4, 7, 9

Goal of the Project:

The goal of this project is to understand where volcanoes are mainly populated, the differences between types of volcanoes, and the correlation between the elevation and classification. Volcanoes are occupied near the shoreline or within islands which can cause differences in the elevation and can affect the time when the volcano last erupted. Data visualization can assist with understanding these correlations, help with natural disaster prevention, and increase scientific knowledge and understanding of Earth's geology and tectonic activities.



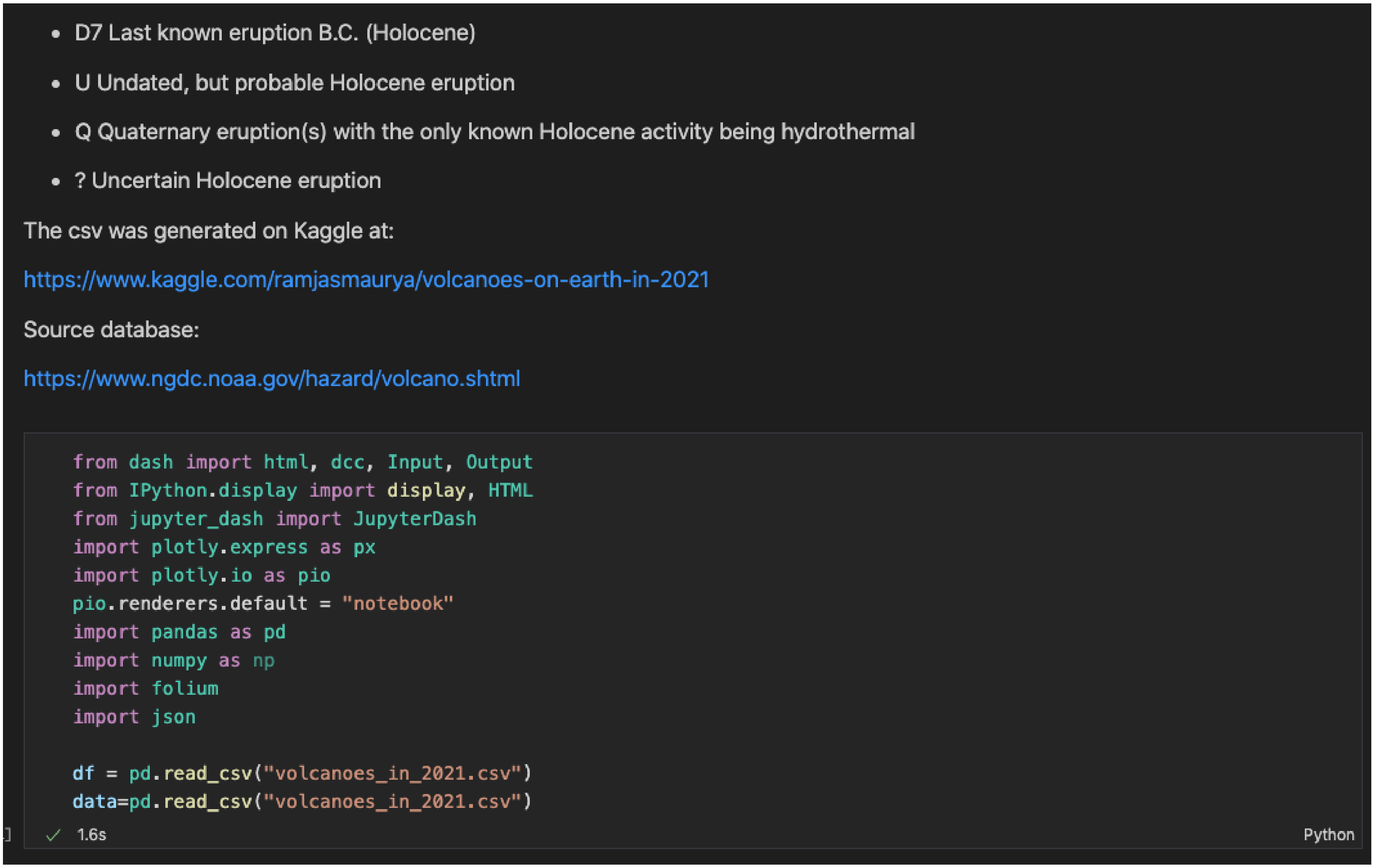
Data

Our data is sourced from the National Centers for Environmental information Volcanic Locations Database. The data has the following columns:

1. Region
2. Number
3. Volcano Name
4. Country
5. Location
6. Latitude
7. Longitude
8. Elevation (m)
9. Type
10. Status
11. Last Known Eruption

The time of last known eruption is represented by the following codes:

- D1 Last known eruption 1964 or later
- D2 Last known eruption 1900-1963
- D3 Last known eruption 1800-1899
- D4 Last known eruption 1700-1799
- D5 Last known eruption 1500-1699
- D6 Last known eruption A.D. 1-1499



- D7 Last known eruption B.C. (Holocene)
- U Undated, but probable Holocene eruption
- Q Quaternary eruption(s) with the only known Holocene activity being hydrothermal
- ? Uncertain Holocene eruption

The csv was generated on Kaggle at:

https://www.kaggle.com/ramjasmaurya/volcanoes-on-earth-in-2021

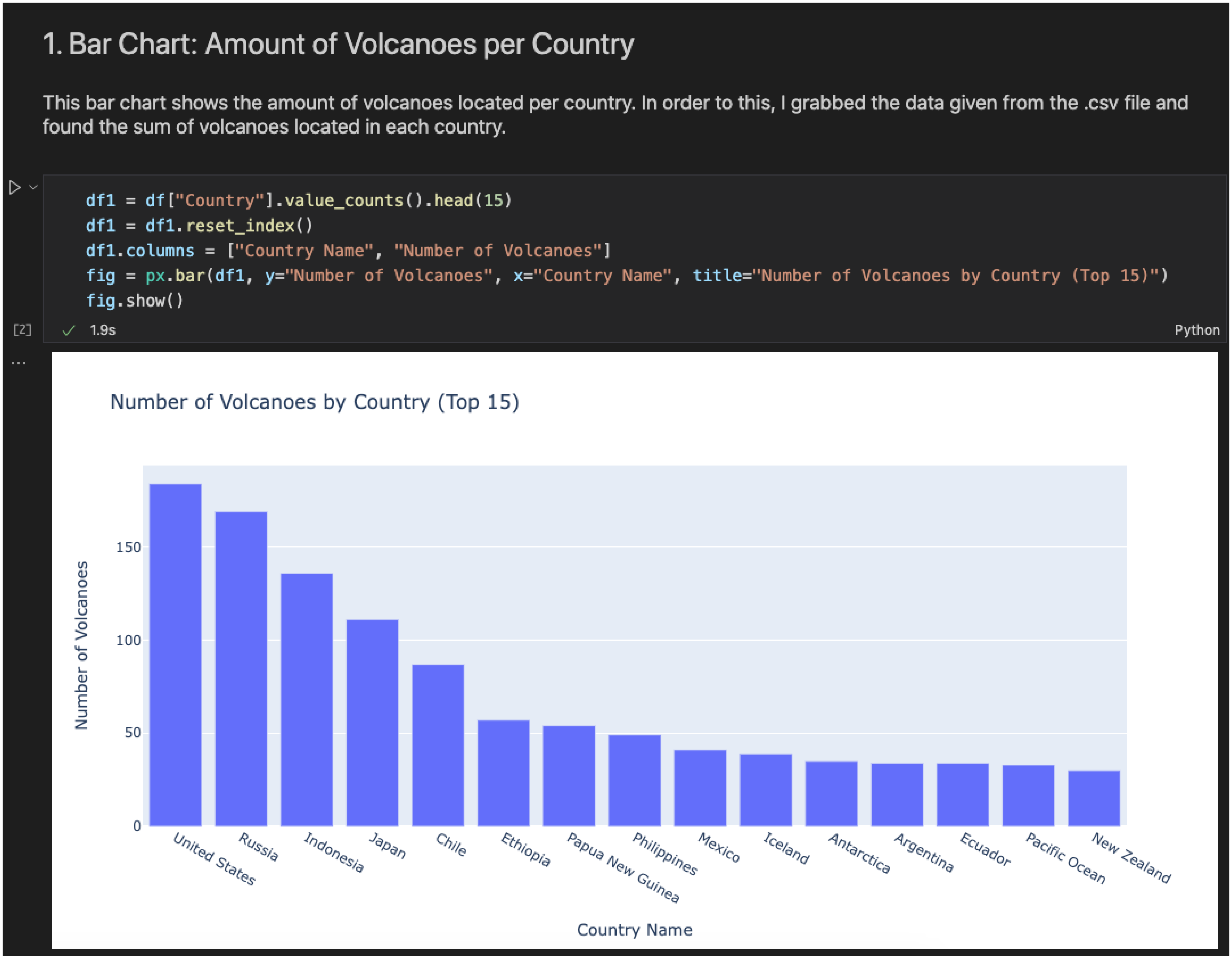
Source database:

https://www.ngdc.noaa.gov/hazard/volcano.shtml

```
from dash import html, dcc, Input, Output
from IPython.display import display, HTML
from jupyter_dash import JupyterDash
import plotly.express as px
import plotly.io as pio
pio.renderers.default = "notebook"
import pandas as pd
import numpy as np
import folium
import json

df = pd.read_csv("volcanoes_in_2021.csv")
data=pd.read_csv("volcanoes_in_2021.csv")
```

✓ 1.6s

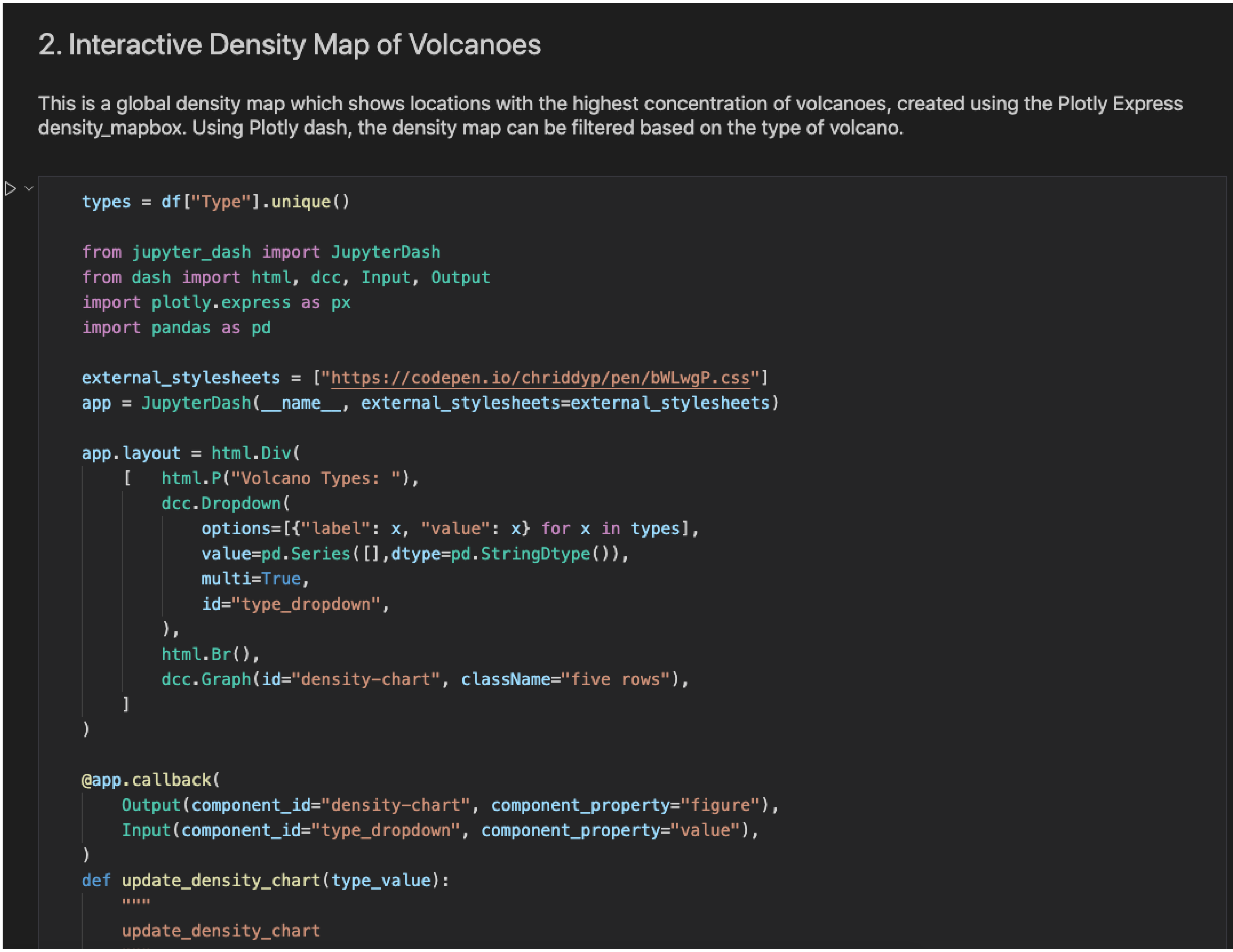
Python



1. Bar Chart: Amount of Volcanoes per Country

This bar chart shows the amount of volcanoes located per country. In order to this, I grabbed the data given from the .csv file and found the sum of volcanoes located in each country.

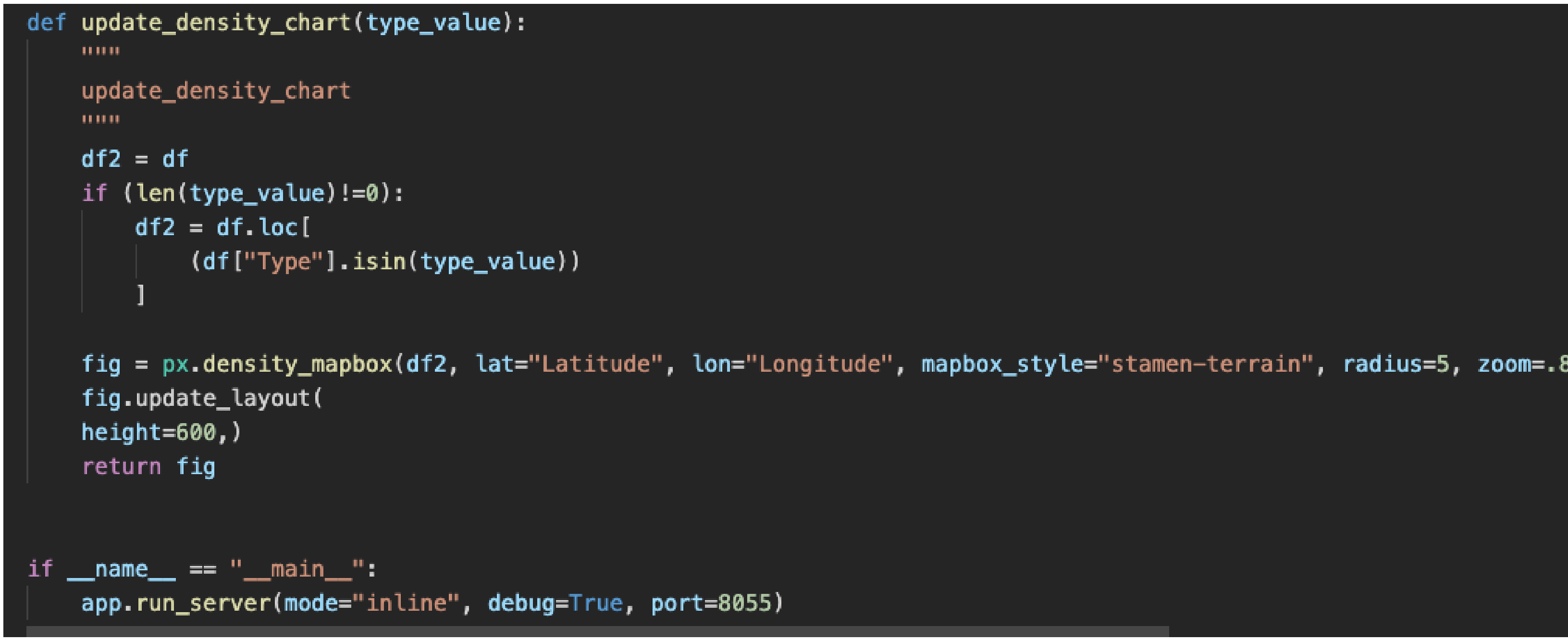
```
df1 = df["Country"].value_counts().head(15)
df1 = df1.reset_index()
df1.columns = ["Country Name", "Number of Volcanoes"]
fig = px.bar(df1, y="Number of Volcanoes", x="Country Name", title="Number of Volcanoes by Country (Top 15)")
fig.show()
```



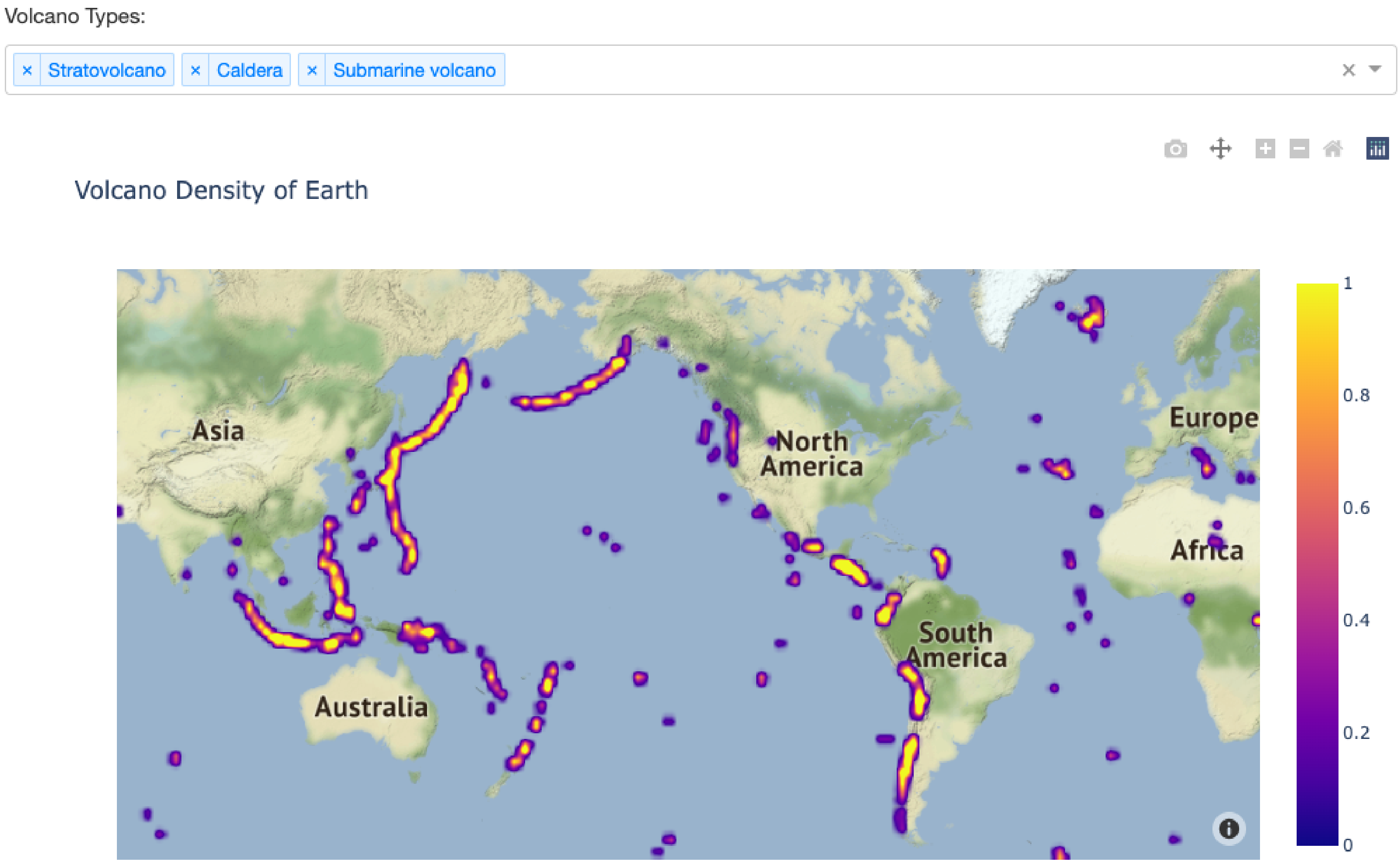
2. Interactive Density Map of Volcanoes

This is a global density map which shows locations with the highest concentration of volcanoes, created using the Plotly Express density_mapbox. Using Plotly dash, the density map can be filtered based on the type of volcano.

```
types = df["Type"].unique()

from jupyter_dash import JupyterDash
from dash import html, dcc, Input, Output
import plotly.express as px
import pandas as pd

external_stylesheets = ["https://codepen.io/chriddyp/pen/bWLwgP.css"]
app = JupyterDash(__name__, external_stylesheets=external_stylesheets)

app.layout = html.Div(
    [   html.P("Volcano Types: "),
        dcc.Dropdown(
            options=[{"label": x, "value": x} for x in types],
            value=pd.Series([],dtype=pd.StringDtype()),
            multi=True,
            id="type_dropdown",
        ),
        html.Br(),
        dcc.Graph(id="density-chart", className="five rows"),
    ]
)

@app.callback(
    Output(component_id="density-chart", component_property="figure"),
    Input(component_id="type_dropdown", component_property="value"),
)
def update_density_chart(type_value):
    """
    update_density_chart
```



```
def update_density_chart(type_value):
    """
    update_density_chart
    """
    df2 = df
    if (len(type_value)!=0):
        df2 = df.loc[
            (df["Type"].isin(type_value))
        ]

    fig = px.density_mapbox(df2, lat="Latitude", lon="Longitude", mapbox_style="stamen-terrain", radius=5, zoom=.8
    fig.update_layout(
    height=600,)
    return fig


if __name__ == "__main__":
    app.run_server(mode="inline", debug=True, port=8055)
```



Volcano Types:

× Stratovolcano × Caldera × Submarine volcano

Volcano Density of Earth

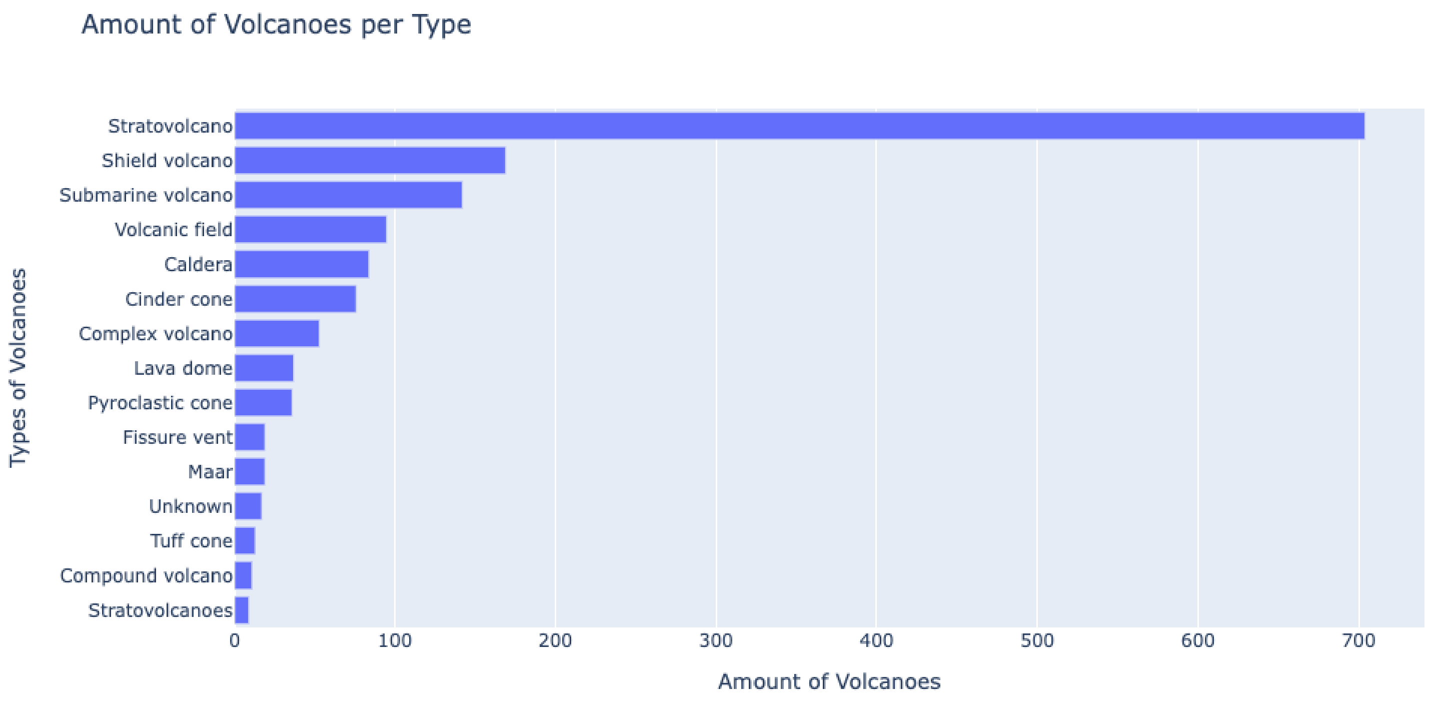
3. Bar Chart: Amount of Volcanoes per Type

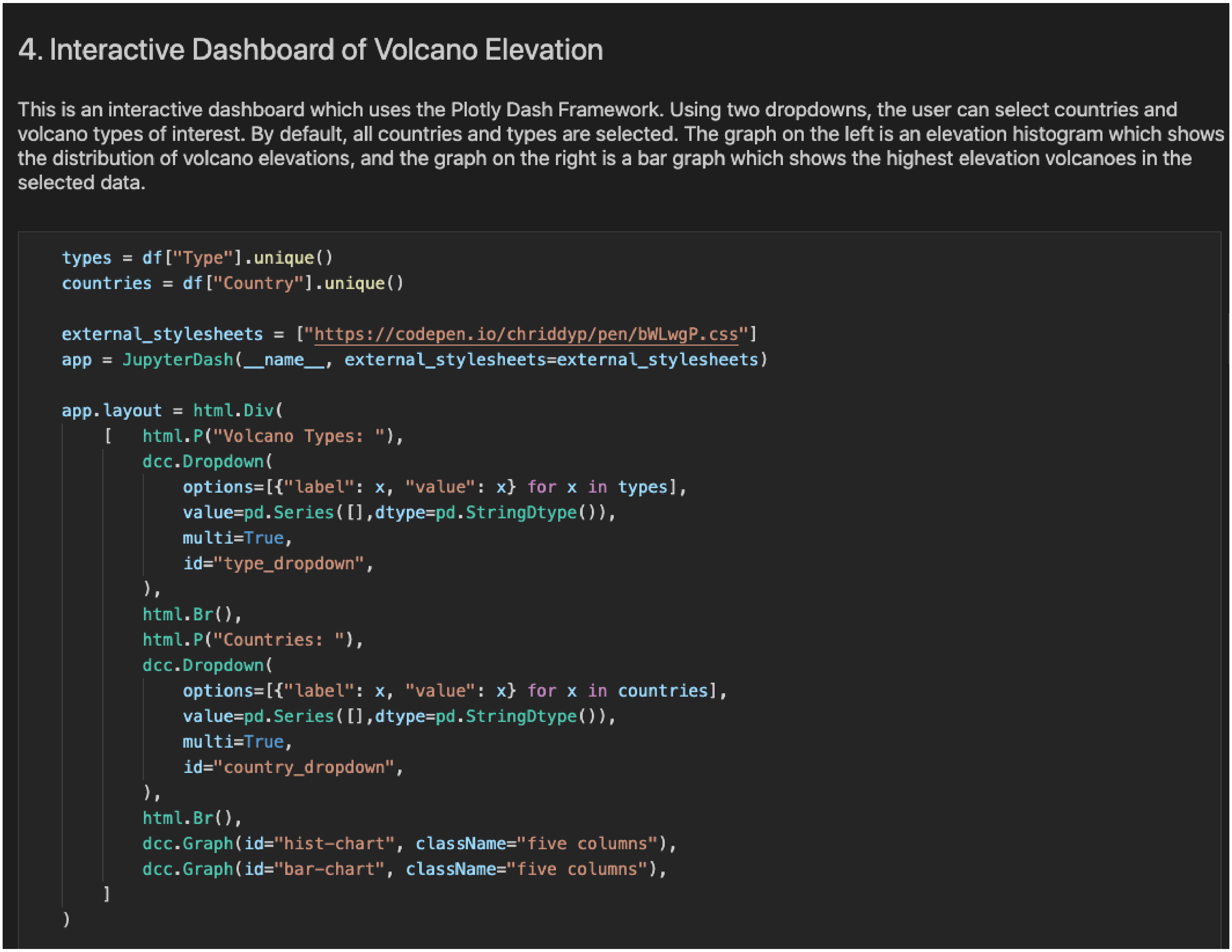
This bar chart shows the amount of volcanoes per volcano type. I used the data provided by NGDC and found the sum for volcanoes per type.

```
value = data['Type'].value_counts().head(15).sort_values()
x = value.index
y = value.values
fig = px.bar(value, x=y, y=x, title='Amount of Volcanoes per Type')
fig.update_layout(xaxis_title='Amount of Volcanoes', yaxis_title='Types of Volcanoes')
fig.show()
```

[4] ✓ 0.1s Python

...

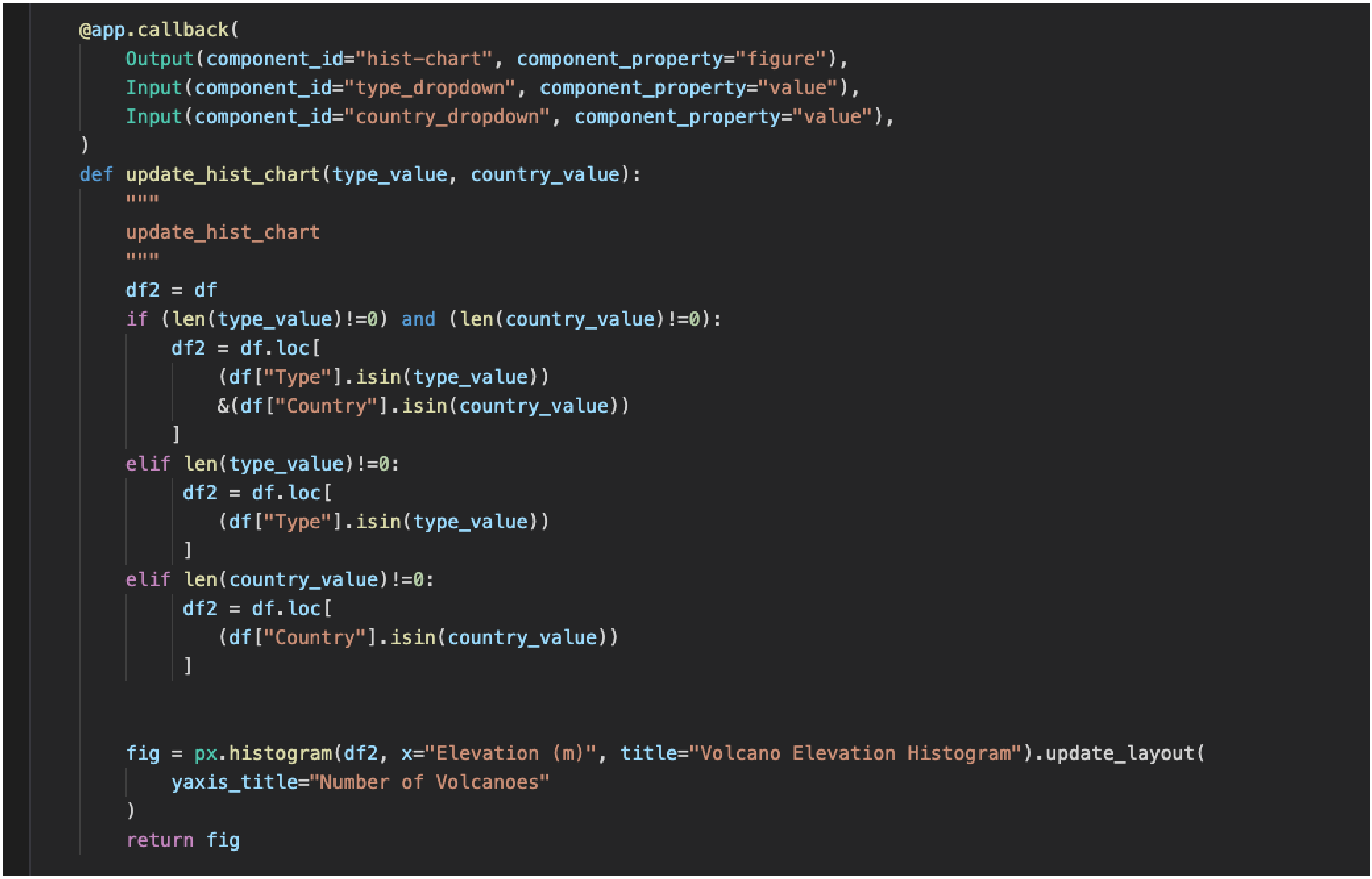




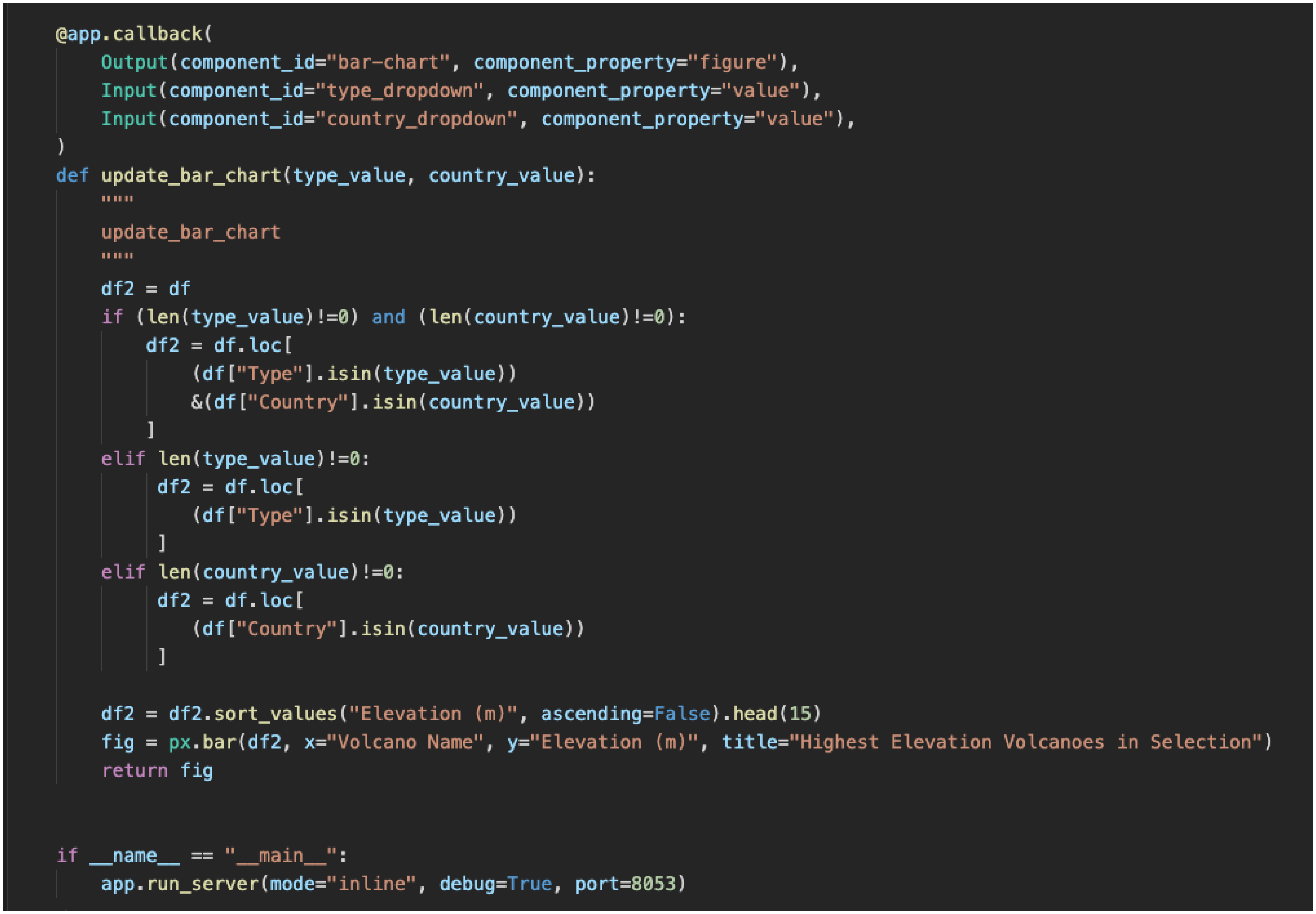
4. Interactive Dashboard of Volcano Elevation

This is an interactive dashboard which uses the Plotly Dash Framework. Using two dropdowns, the user can select countries and volcano types of interest. By default, all countries and types are selected. The graph on the left is an elevation histogram which shows the distribution of volcano elevations, and the graph on the right is a bar graph which shows the highest elevation volcanoes in the selected data.

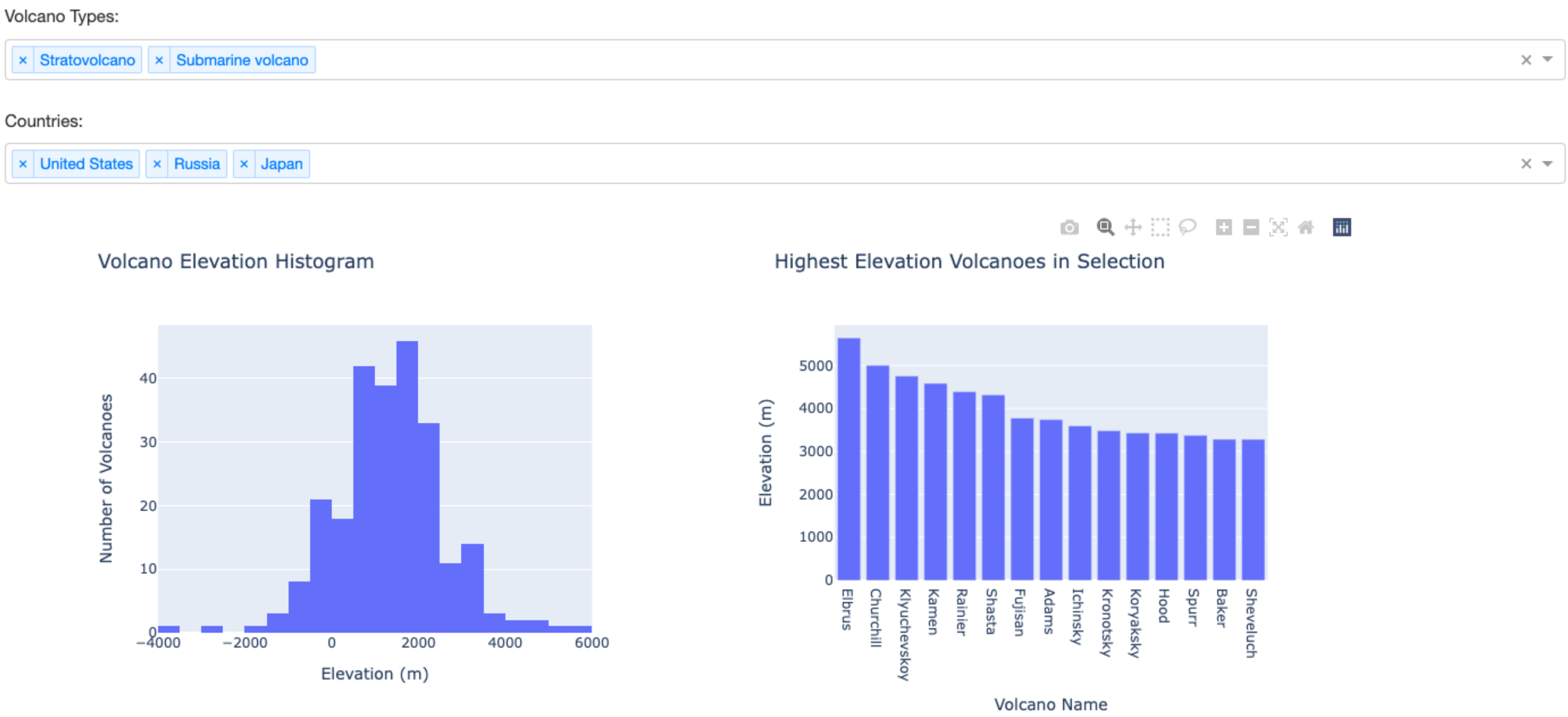
```python
types = df["Type"].unique()
countries = df["Country"].unique()

external_stylesheets = ["https://codepen.io/chriddyp/pen/bWLwgP.css"]
app = JupyterDash(__name__, external_stylesheets=external_stylesheets)

app.layout = html.Div(
    [   html.P("Volcano Types: "),
        dcc.Dropdown(
            options=[{"label": x, "value": x} for x in types],
            value=pd.Series([],dtype=pd.StringDtype()),
            multi=True,
            id="type_dropdown",
        ),
        html.Br(),
        html.P("Countries: "),
        dcc.Dropdown(
            options=[{"label": x, "value": x} for x in countries],
            value=pd.Series([],dtype=pd.StringDtype()),
            multi=True,
            id="country_dropdown",
        ),
        html.Br(),
        dcc.Graph(id="hist-chart", className="five columns"),
        dcc.Graph(id="bar-chart", className="five columns"),
    ]
)
```



```
@app.callback(
    Output(component_id="hist-chart", component_property="figure"),
    Input(component_id="type_dropdown", component_property="value"),
    Input(component_id="country_dropdown", component_property="value"),
)
def update_hist_chart(type_value, country_value):
    """
    update_hist_chart
    """
    df2 = df
    if (len(type_value)!=0) and (len(country_value)!=0):
        df2 = df.loc[
            (df["Type"].isin(type_value))
            &(df["Country"].isin(country_value))
        ]
    elif len(type_value)!=0:
         df2 = df.loc[
            (df["Type"].isin(type_value))
         ]
    elif len(country_value)!=0:
         df2 = df.loc[
            (df["Country"].isin(country_value))
         ]


    fig = px.histogram(df2, x="Elevation (m)", title="Volcano Elevation Histogram").update_layout(
        yaxis_title="Number of Volcanoes"
    )
    return fig
```



```
@app.callback(
    Output(component_id="bar-chart", component_property="figure"),
    Input(component_id="type_dropdown", component_property="value"),
    Input(component_id="country_dropdown", component_property="value"),
)
def update_bar_chart(type_value, country_value):
    """
    update_bar_chart
    """
    df2 = df
    if (len(type_value)!=0) and (len(country_value)!=0):
        df2 = df.loc[
            (df["Type"].isin(type_value))
            &(df["Country"].isin(country_value))
        ]
    elif len(type_value)!=0:
         df2 = df.loc[
            (df["Type"].isin(type_value))
         ]
    elif len(country_value)!=0:
         df2 = df.loc[
            (df["Country"].isin(country_value))
         ]

    df2 = df2.sort_values("Elevation (m)", ascending=False).head(15)
    fig = px.bar(df2, x="Volcano Name", y="Elevation (m)", title="Highest Elevation Volcanoes in Selection")
    return fig


if __name__ == "__main__":
    app.run_server(mode="inline", debug=True, port=8053)
```



Volcano Types:

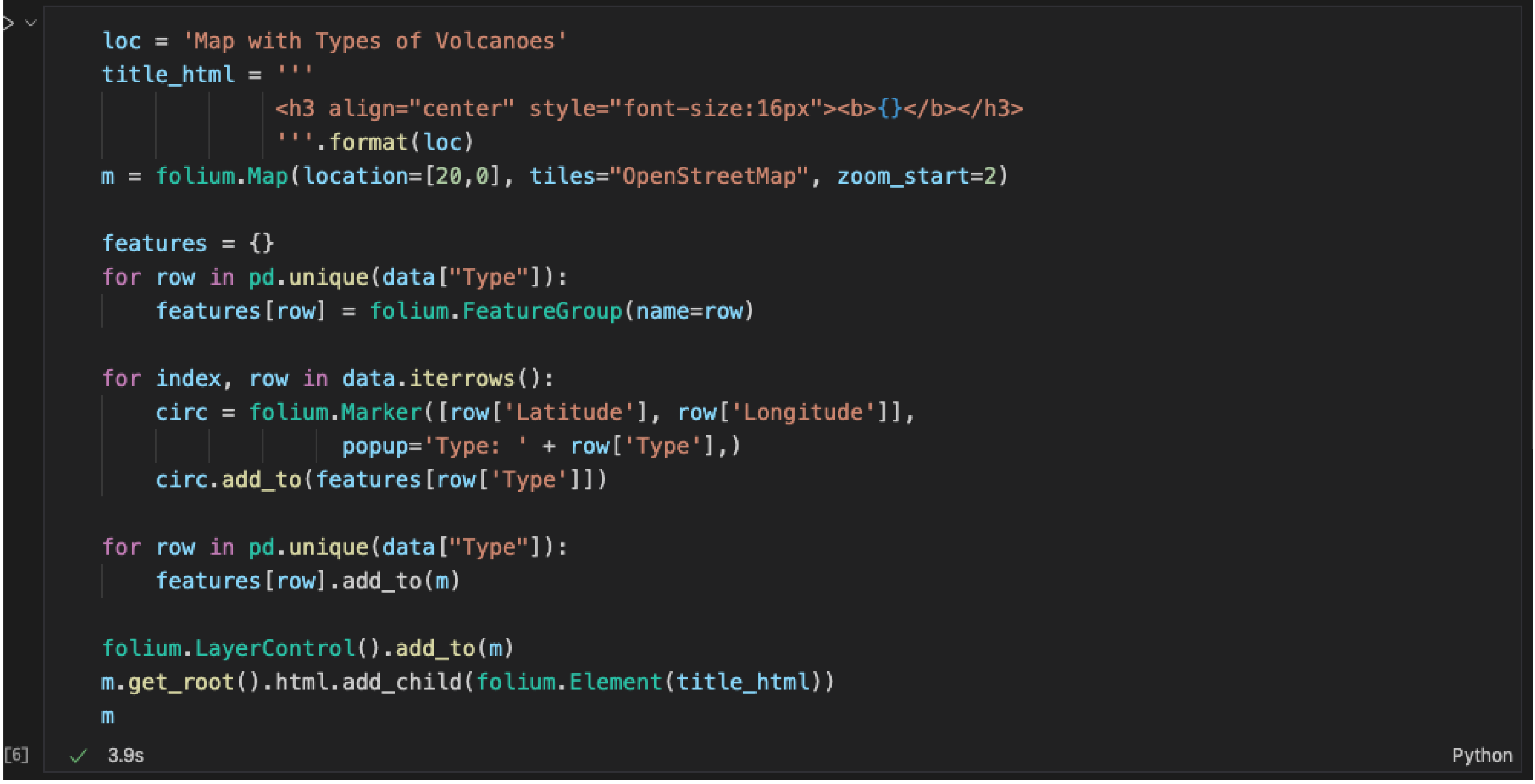
× Stratovolcano × Submarine volcano

Countries:

× United States × Russia × Japan

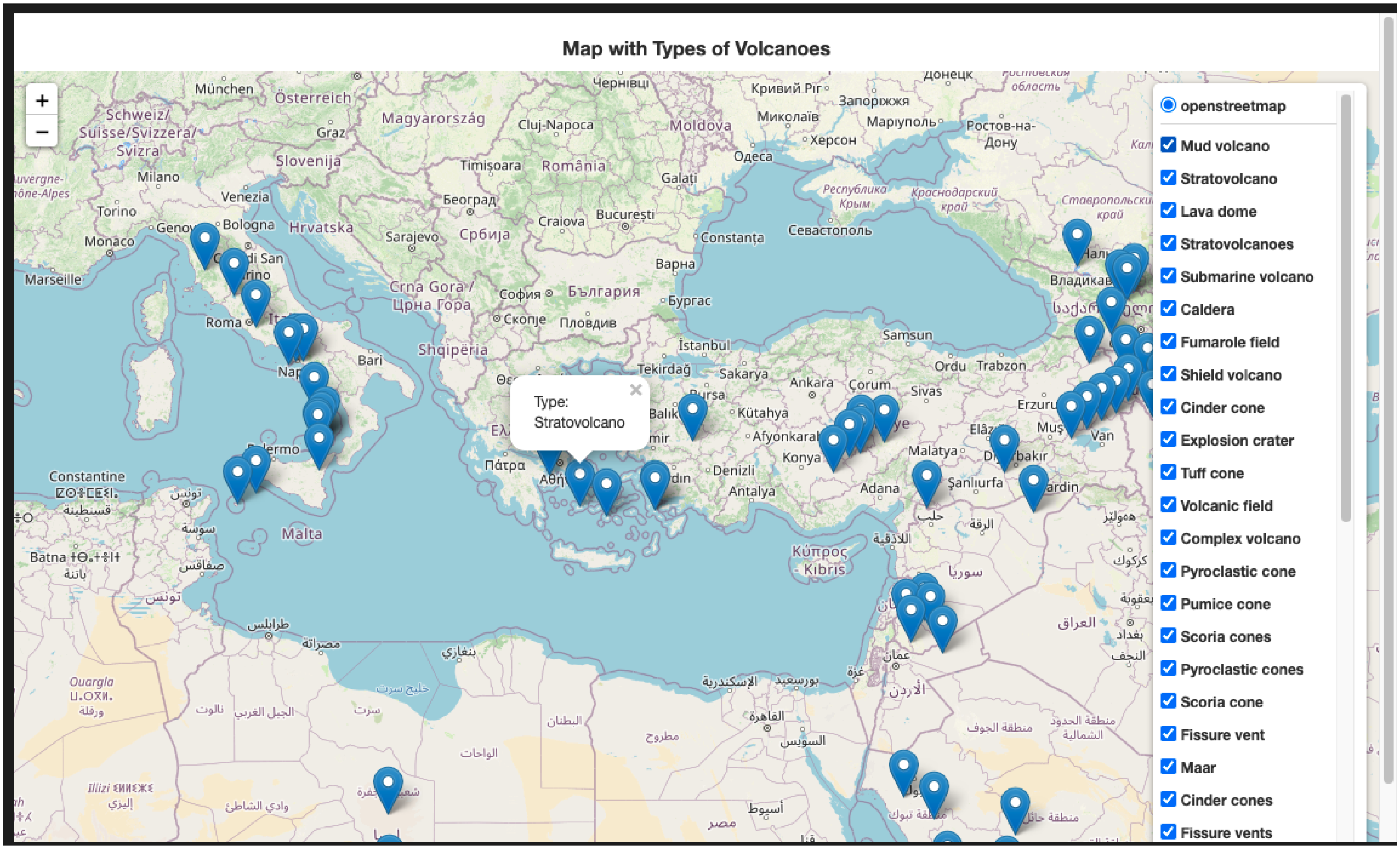
5. Map: Map with Types of Volcanoes

This Map shows all the volcanoes located on Earth and it has a marker with a pop-up that shows the type of volcano. It also provides a filter so for example, the user will be able to see the types they would want to see.



```
loc = 'Map with Types of Volcanoes'
title_html = '''
             <h3 align="center" style="font-size:16px"><b>{}</b></h3>
             '''.format(loc)
m = folium.Map(location=[20,0], tiles="OpenStreetMap", zoom_start=2)

features = {}
for row in pd.unique(data["Type"]):
    features[row] = folium.FeatureGroup(name=row)

for index, row in data.iterrows():
    circ = folium.Marker([row['Latitude'], row['Longitude']],
                         popup='Type: ' + row['Type'],)
    circ.add_to(features[row['Type']])

for row in pd.unique(data["Type"]):
    features[row].add_to(m)

folium.LayerControl().add_to(m)
m.get_root().html.add_child(folium.Element(title_html))
m
```

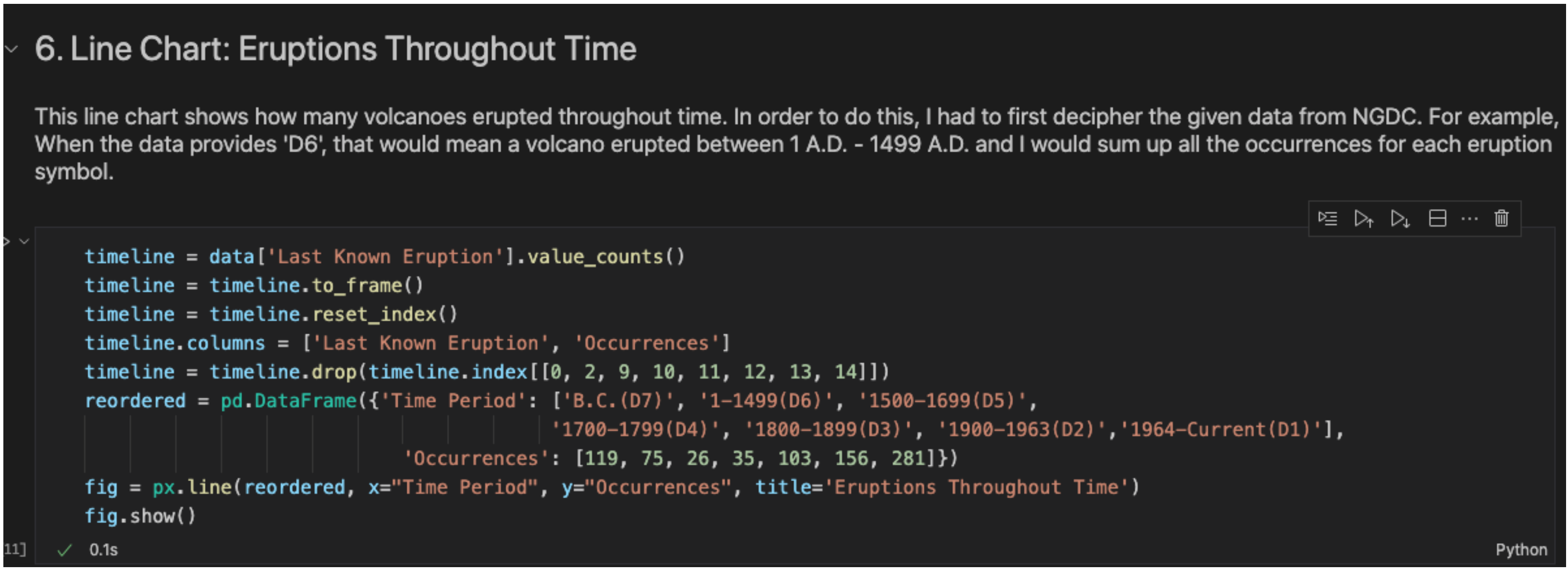
[6] ✓ 3.9s Python



Map with Types of Volcanoes

Type: Stratovolcano

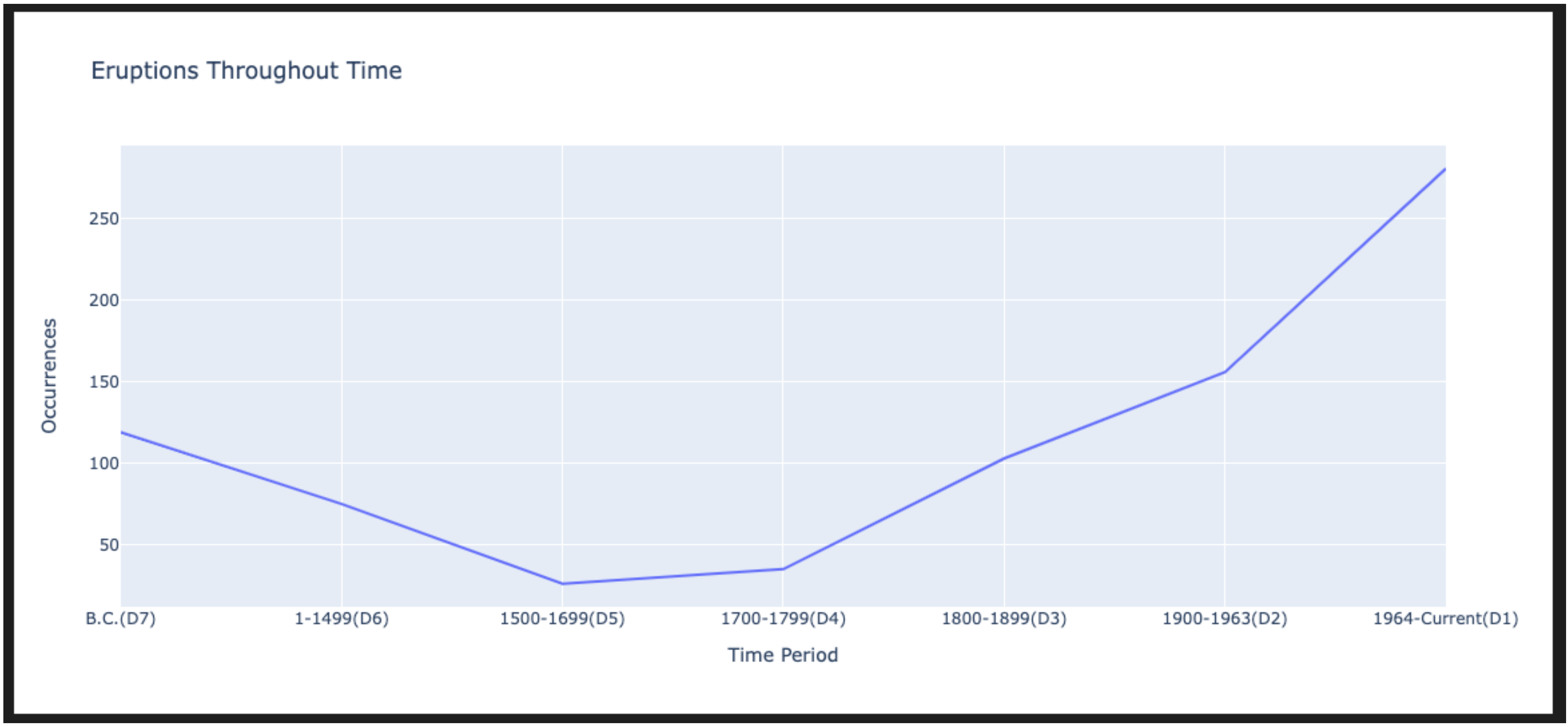
● openstreetmap
☑ Mud volcano
☑ Stratovolcano
☑ Lava dome
☑ Stratovolcanoes
☑ Submarine volcano
☑ Caldera
☑ Fumarole field
☑ Shield volcano
☑ Cinder cone
☑ Explosion crater
☑ Tuff cone
☑ Volcanic field
☑ Complex volcano
☑ Pyroclastic cone
☑ Pumice cone
☑ Scoria cones
☑ Pyroclastic cones
☑ Scoria cone
☑ Fissure vent
☑ Maar
☑ Cinder cones
☑ Fissure vents

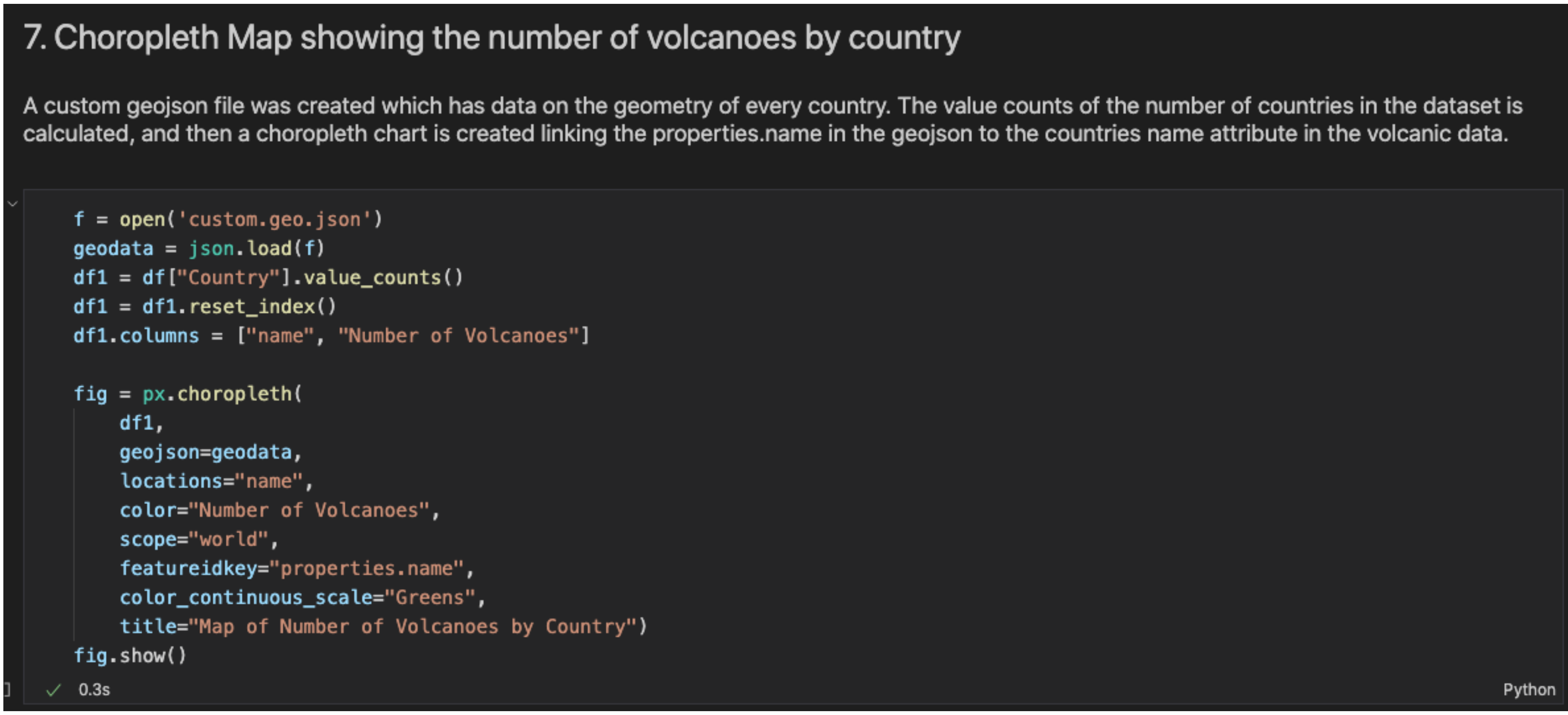


6. Line Chart: Eruptions Throughout Time

This line chart shows how many volcanoes erupted throughout time. In order to do this, I had to first decipher the given data from NGDC. For example, When the data provides 'D6', that would mean a volcano erupted between 1 A.D. - 1499 A.D. and I would sum up all the occurrences for each eruption symbol.

```python
timeline = data['Last Known Eruption'].value_counts()
timeline = timeline.to_frame()
timeline = timeline.reset_index()
timeline.columns = ['Last Known Eruption', 'Occurrences']
timeline = timeline.drop(timeline.index[[0, 2, 9, 10, 11, 12, 13, 14]])
reordered = pd.DataFrame({'Time Period': ['B.C.(D7)', '1-1499(D6)', '1500-1699(D5)',
                                         '1700-1799(D4)', '1800-1899(D3)', '1900-1963(D2)','1964-Current(D1)'],
                            'Occurrences': [119, 75, 26, 35, 103, 156, 281]})
fig = px.line(reordered, x="Time Period", y="Occurrences", title='Eruptions Throughout Time')
fig.show()
```





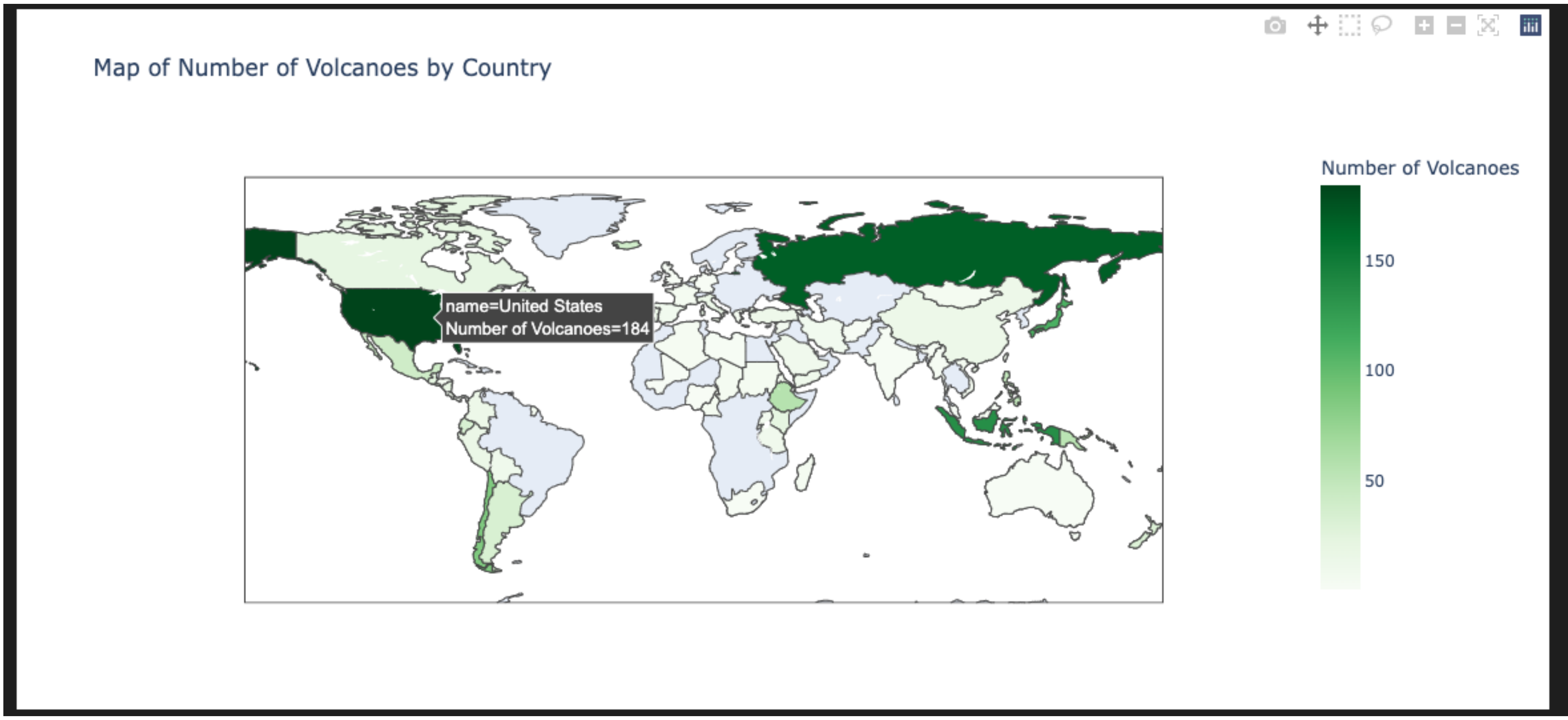
7. Choropleth Map showing the number of volcanoes by country

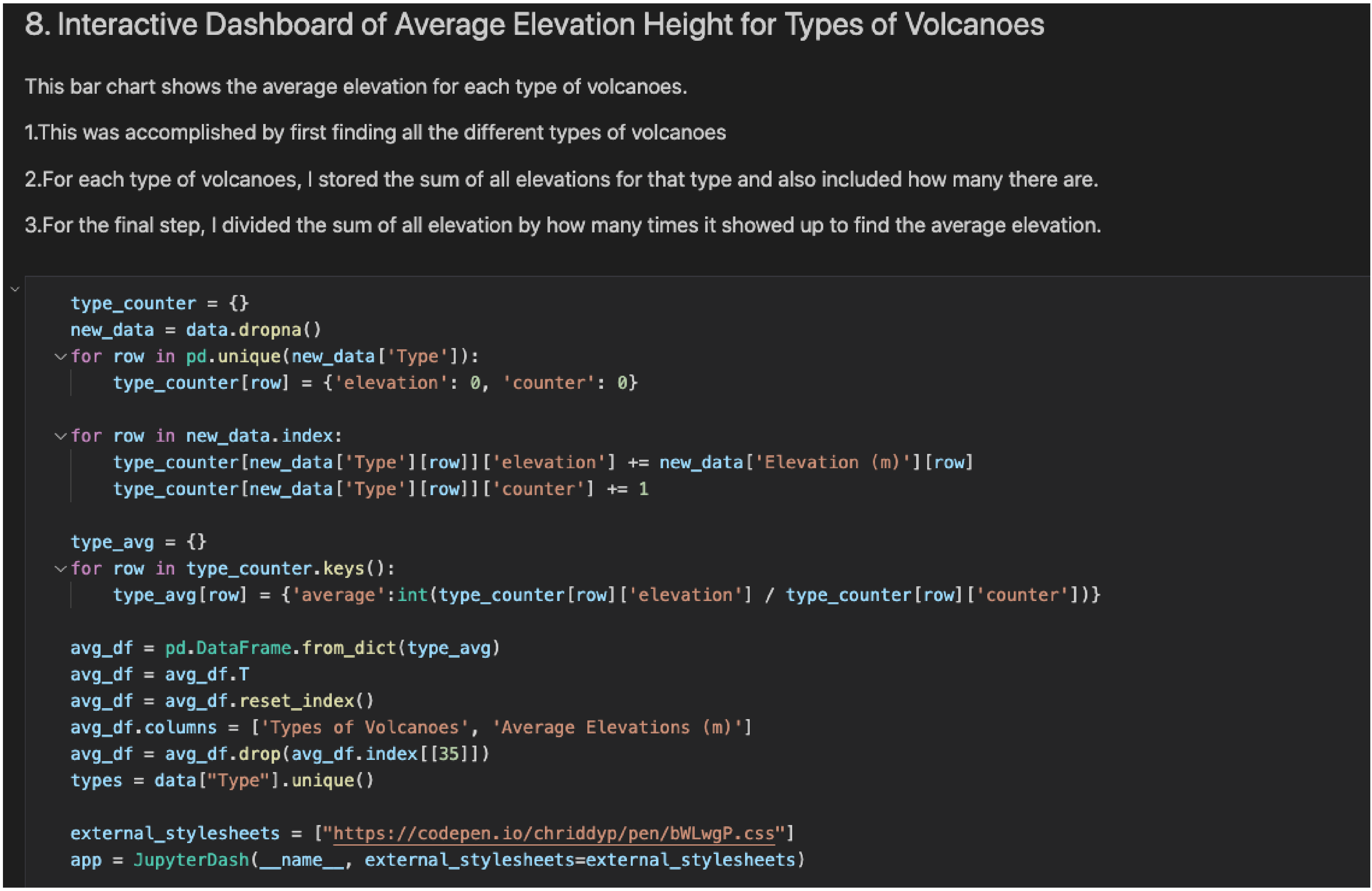
A custom geojson file was created which has data on the geometry of every country. The value counts of the number of countries in the dataset is calculated, and then a choropleth chart is created linking the properties.name in the geojson to the countries name attribute in the volcanic data.

```python
f = open('custom.geo.json')
geodata = json.load(f)
df1 = df["Country"].value_counts()
df1 = df1.reset_index()
df1.columns = ["name", "Number of Volcanoes"]

fig = px.choropleth(
    df1,
    geojson=geodata,
    locations="name",
    color="Number of Volcanoes",
    scope="world",
    featureidkey="properties.name",
    color_continuous_scale="Greens",
    title="Map of Number of Volcanoes by Country")
fig.show()
```

✓ 0.3s

Python





8. Interactive Dashboard of Average Elevation Height for Types of Volcanoes

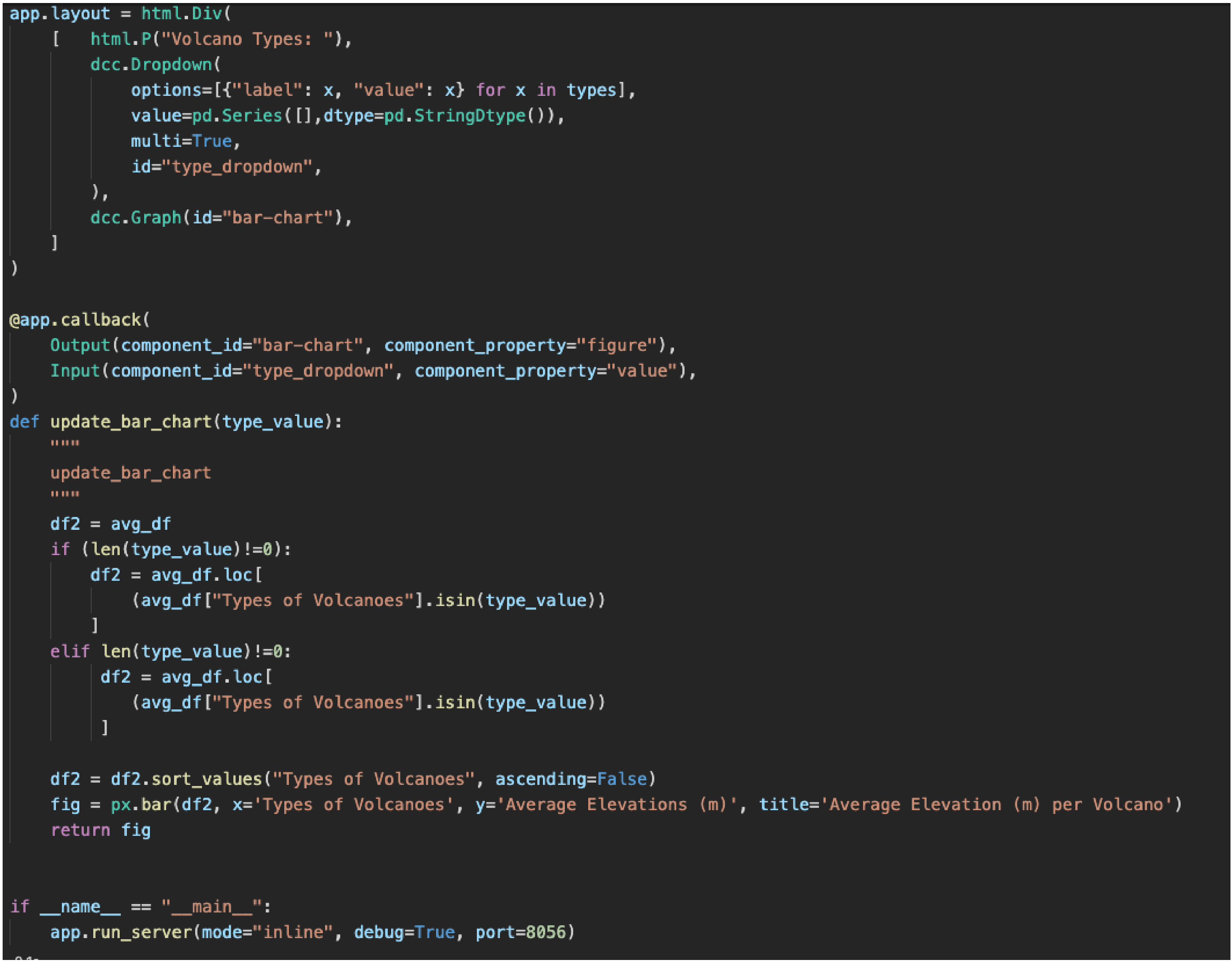
This bar chart shows the average elevation for each type of volcanoes.

1.This was accomplished by first finding all the different types of volcanoes

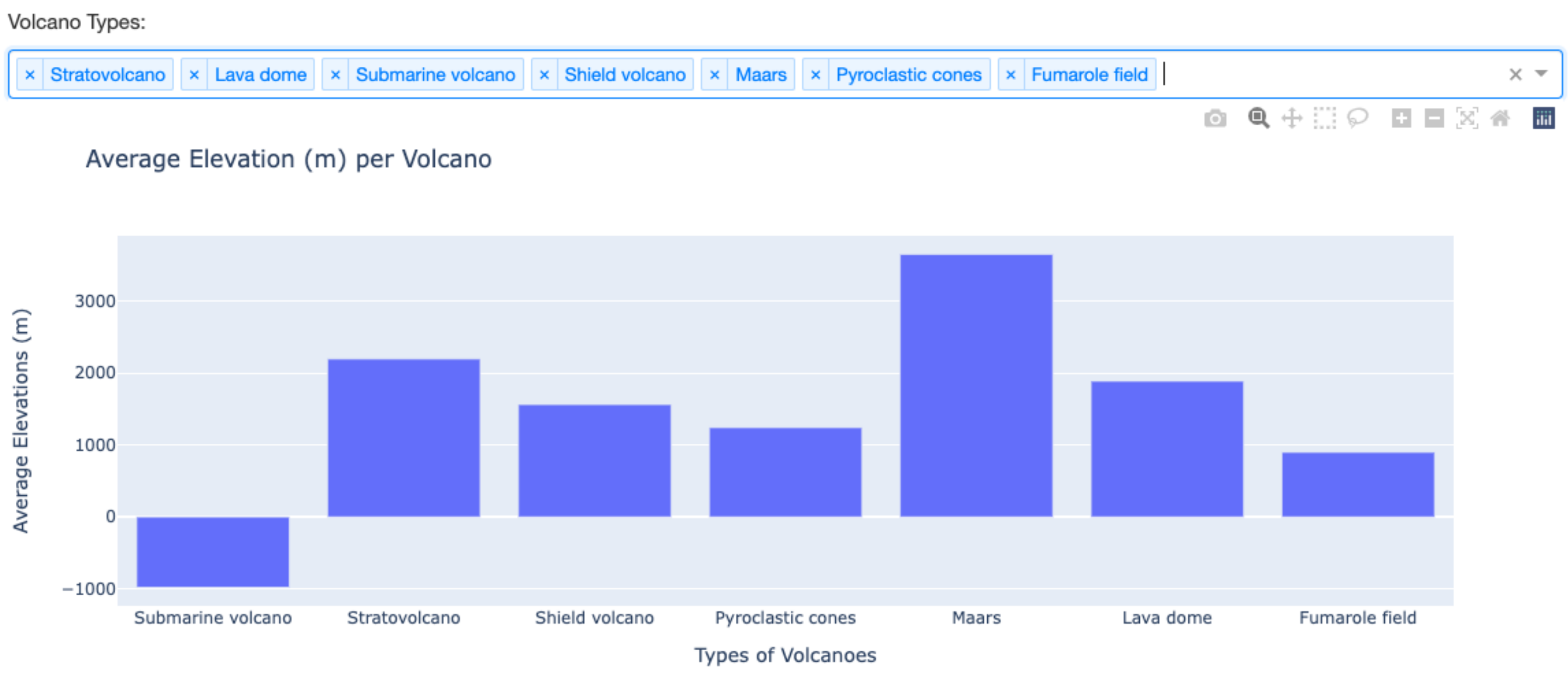
2.For each type of volcanoes, I stored the sum of all elevations for that type and also included how many there are.

3.For the final step, I divided the sum of all elevation by how many times it showed up to find the average elevation.

```python
type_counter = {}
new_data = data.dropna()
for row in pd.unique(new_data['Type']):
    type_counter[row] = {'elevation': 0, 'counter': 0}

for row in new_data.index:
    type_counter[new_data['Type'][row]]['elevation'] += new_data['Elevation (m)'][row]
    type_counter[new_data['Type'][row]]['counter'] += 1

type_avg = {}
for row in type_counter.keys():
    type_avg[row] = {'average':int(type_counter[row]['elevation'] / type_counter[row]['counter'])}

avg_df = pd.DataFrame.from_dict(type_avg)
avg_df = avg_df.T
avg_df = avg_df.reset_index()
avg_df.columns = ['Types of Volcanoes', 'Average Elevations (m)']
avg_df = avg_df.drop(avg_df.index[[35]])
types = data["Type"].unique()

external_stylesheets = ["https://codepen.io/chriddyp/pen/bWLwgP.css"]
app = JupyterDash(__name__, external_stylesheets=external_stylesheets)
```

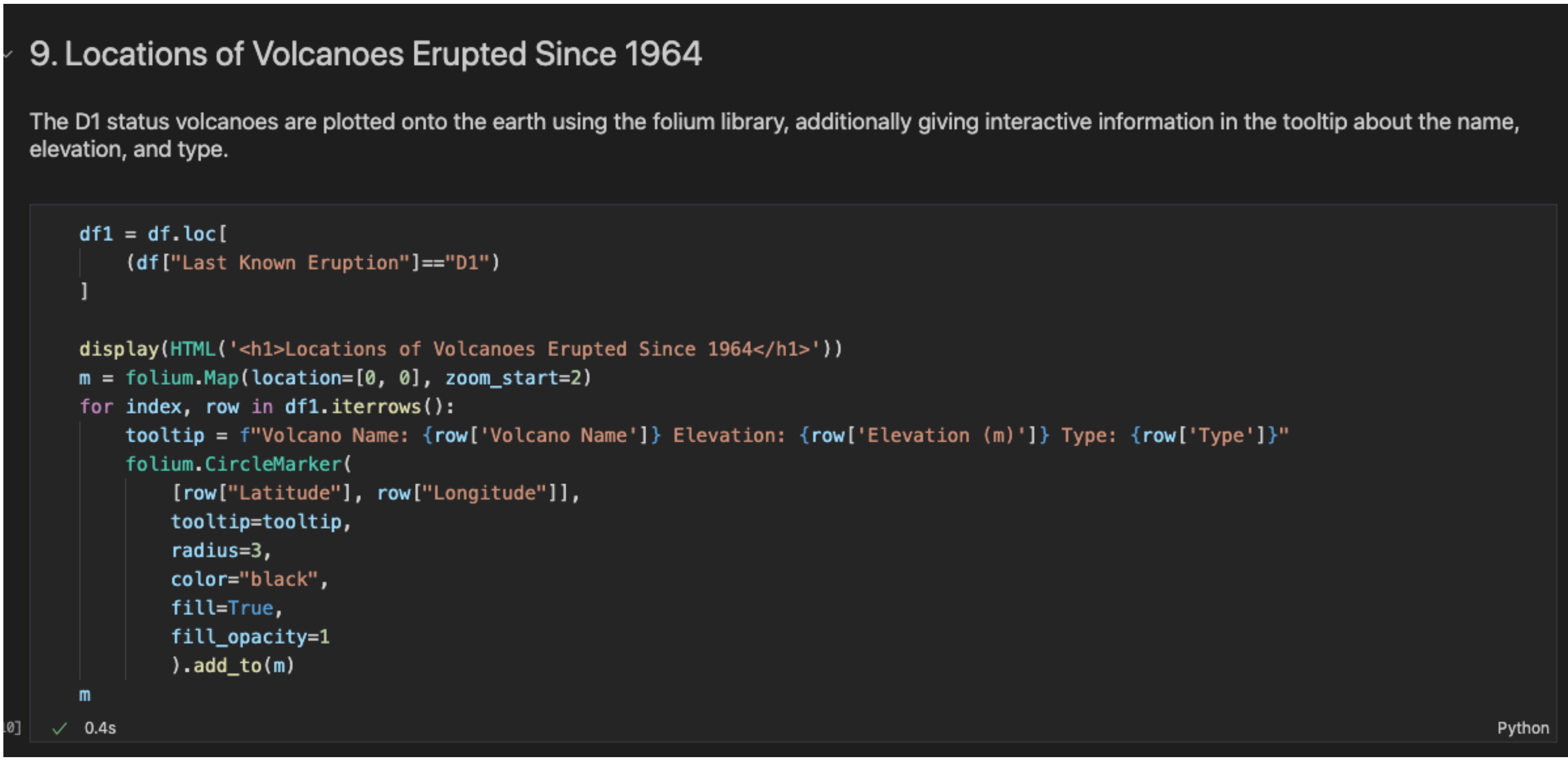


```
app.layout = html.Div(
    [   html.P("Volcano Types: "),
        dcc.Dropdown(
            options=[{"label": x, "value": x} for x in types],
            value=pd.Series([],dtype=pd.StringDtype()),
            multi=True,
            id="type_dropdown",
        ),
        dcc.Graph(id="bar-chart"),
    ]
)

@app.callback(
    Output(component_id="bar-chart", component_property="figure"),
    Input(component_id="type_dropdown", component_property="value"),
)
def update_bar_chart(type_value):
    """
    update_bar_chart
    """
    df2 = avg_df
    if (len(type_value)!=0):
        df2 = avg_df.loc[
            (avg_df["Types of Volcanoes"].isin(type_value))
        ]
    elif len(type_value)!=0:
         df2 = avg_df.loc[
            (avg_df["Types of Volcanoes"].isin(type_value))
         ]

    df2 = df2.sort_values("Types of Volcanoes", ascending=False)
    fig = px.bar(df2, x='Types of Volcanoes', y='Average Elevations (m)', title='Average Elevation (m) per Volcano')
    return fig


if __name__ == "__main__":
    app.run_server(mode="inline", debug=True, port=8056)
```



Volcano Types:

× Stratovolcano × Lava dome × Submarine volcano × Shield volcano × Maars × Pyroclastic cones × Fumarole field



9. Locations of Volcanoes Erupted Since 1964

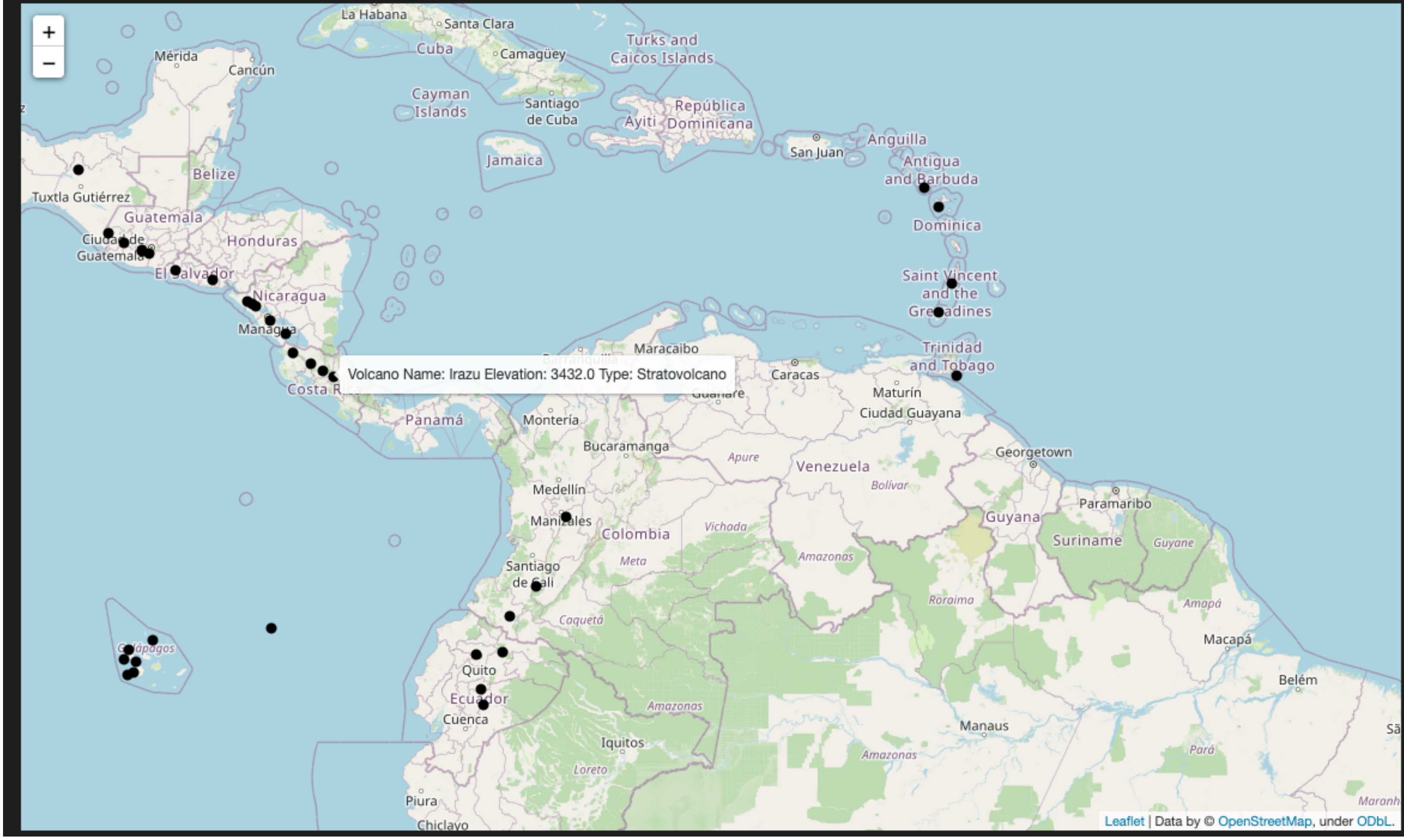
The D1 status volcanoes are plotted onto the earth using the folium library, additionally giving interactive information in the tooltip about the name, elevation, and type.

```python
df1 = df.loc[
    (df["Last Known Eruption"]=="D1")
]

display(HTML('<h1>Locations of Volcanoes Erupted Since 1964</h1>'))
m = folium.Map(location=[0, 0], zoom_start=2)
for index, row in df1.iterrows():
    tooltip = f"Volcano Name: {row['Volcano Name']} Elevation: {row['Elevation (m)']} Type: {row['Type']}"
    folium.CircleMarker(
        [row["Latitude"], row["Longitude"]],
        tooltip=tooltip,
        radius=3,
        color="black",
        fill=True,
        fill_opacity=1
        ).add_to(m)
m
```

✓ 0.4s

Python

Locations of Volcanoes Erupted Since 1964



Volcano Name: Irazu Elevation: 3432.0 Type: Stratovolcano

Leaflet | Data by © OpenStreetMap, under ODbL.

Conclusions

- Certain countries have hundreds of volcanoes (USA, Russia, Indonesia, Japan), while other countries have zero.
- Additionally, there seems to be a strong correlation between the locations of volcanoes and the fault lines of tectonic plates.
- However, there are certain types of volcanoes that may not be near tectonic activity, such as volcanic fields.
- Stratovolcanoes are the most overwhelmingly common type of volcano in the world.
- The elevation distribution of the world's volcanoes tend to follow a normal distribution.
- Most of the highest elevation volcanoes in the world are in the Andes mountains region of South America.
- The amount of recorded eruptions seems to be increasing, but this is likely explained by the fact that it is easier to determine when more recent volcanic eruptions happened compared to ancient eruptions.
- Despite Russia having so many volcanoes, most of them are in the mostly unpopulated Kamchatka region on the Pacific rim, meaning that most of the country is quite inactive.
- The United States has a large amount of volcanoes on the West coast and Alaska, with volcanoes much closer to population centers.
- Countries like Japan, Philippines, and Indonesia have highly active volcanoes near very densely populated regions, which could cause dangerous situations in the near future.