GRAPHING DATA USING PLOTLY

Edmond Lascaris and Anne Jessup - Creative Commons – 22 June 2021

OVERVIEW

In this lesson we are going to learn how to **graph data** collected from our environmental sensors. Data will be graphed using a python package called **plotly**. Plotly can produce graphs in **HTML format** so that they can be easily displayed on a web page.

Graphing sensor data is very important because it shows trends and changes in data more clearly.

LEARNING OBJECTIVES

- Learn how to create graphs using plotly
- Learn how to import data from files using pandas
- Learn how to add more formatting options in plotly

CREATING GRAPHS USING PLOTLY

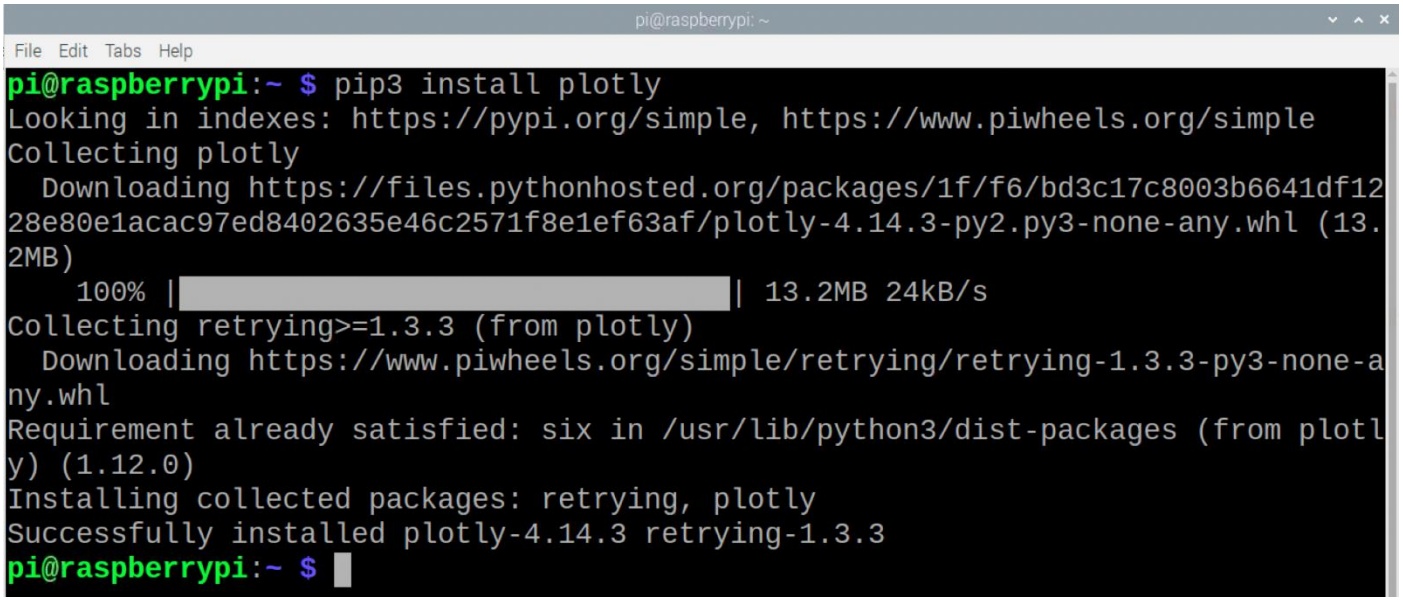
In this example we are going to install the plotly python package and then write a program to create some simple graphs.

1. **Installation of the plotly python package**

- From the Raspberry Pi top bar menu click on the **Terminal** icon.
- Open the Terminal and enter the command **pip3 install plotly**
- **pip** is short for **Pip Installs Python. pip3** installs packages for **python3**

- The installation process will take approximately one minute.

- When plotly has been installed you should see an output like the one below.
- The version of plotly that was installed is **plotly-4.14.3**
- Other python packages may also be installed to support plotly.
- In this instance, another package named **retrying-1.3.3** was also installed.

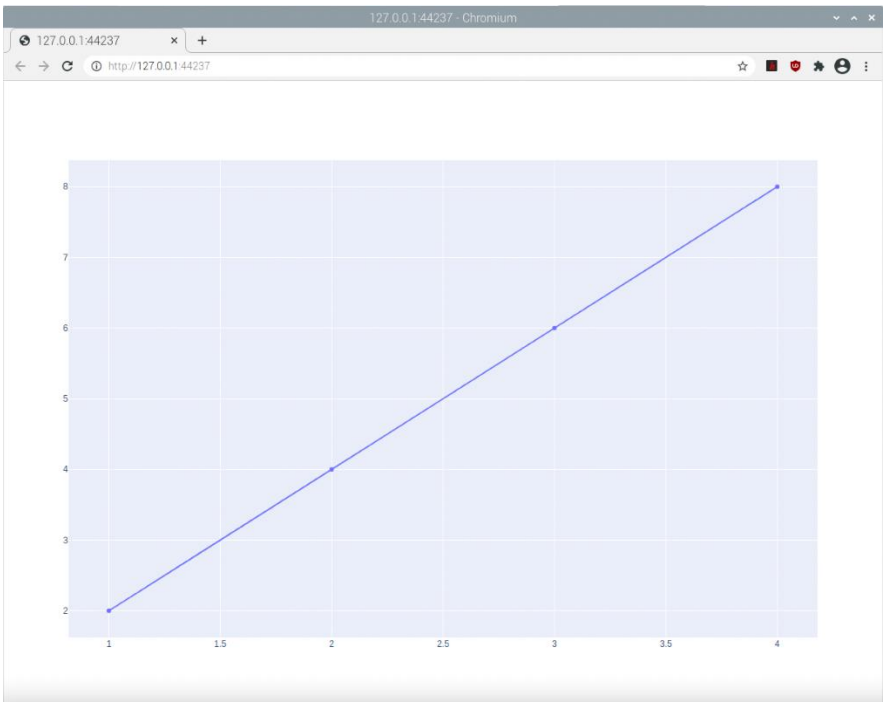


2. First plotly program in Python

- Open **Python3 (IDLE3).**
- From the File dropdown menu select **New File**.
- Enter the python code shown below.
- Save this file as **temp_plot_gen.py** in your project's directory (the file name is short for **temperature plot generator**)

```
File Edit Format Run Options Window Help
import plotly.graph_objects as go

mx = [1, 2, 3, 4]
my = [2, 4, 6, 8]

fig = go.Figure(data = go.Scatter(x=mx, y=my))

fig.show()
```

- **Save** and **Run** the program.
- After approximately 10-20 seconds the web browser Chromium will open and display a **graph** of your data.



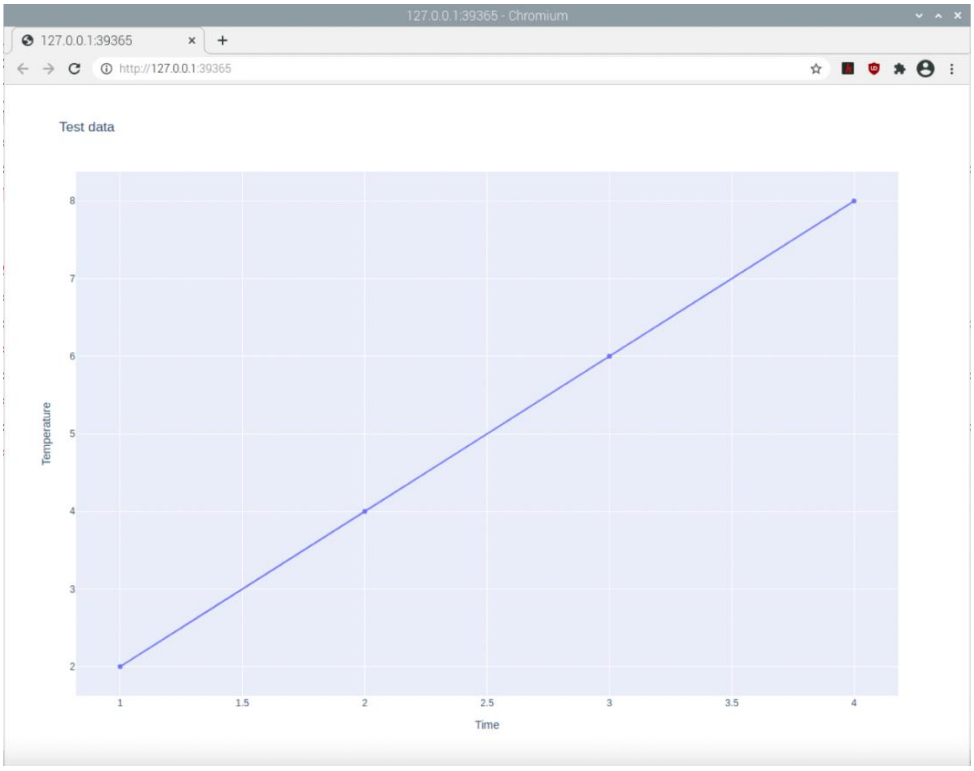
3. Adding title descriptions to your graph

- Update the python code with to add **layout details** in the graph.

```
fig.update_layout(title='Test data',
                  xaxis_title = 'Time',
                  yaxis_title = 'Temperature')
```

```
File  Edit  Format  Run  Options  Window  Help
import plotly.graph_objects as go

mx = [1, 2, 3, 4]
my = [2, 4, 6, 8]

fig = go.Figure(data = go.Scatter(x=mx, y=my))

fig.update_layout(title='Test data',
                  xaxis_title = 'Time',
                  yaxis_title = 'Temperature')

fig.show()
```

- **Save** and **Run** the program. A **new tab** in the web browser will open and display this graph.



IMPORTING AND PLOTTING DATA FROM TEXT FILES

Plotly is good at plotting graphs, but it cannot important data from text files. To prepare data for plotly we need another python package named **pandas**. In the following lesson we will install pandas and demonstrate how to import data from a text file. Before we can start using pandas we also need to **clean** and **label** the data in our text file to make sure it is ready to import.

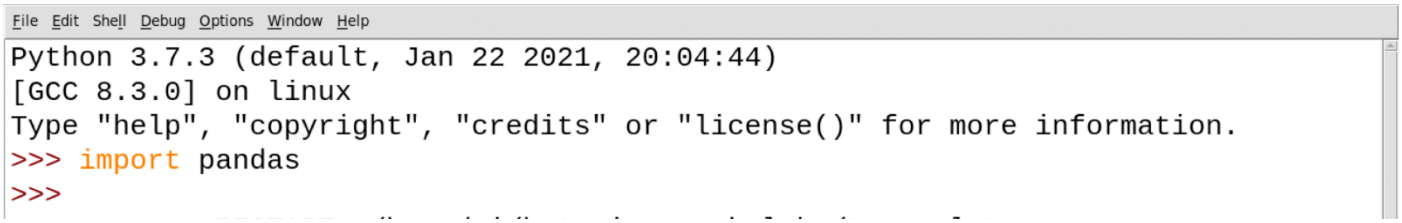
4. Cleaning and adding labels to sensor data

- To import data from our data.txt file need to **clean the data** and give **labels** to the data.
- Click on the **File Manager**.
- Navigate to your project's directory and open the **data.txt** file. By default, it should open in the **Mousepad** text editor.

- The first line of the data file needs to include a **label** for each data set
- **Data labels** for our file are: **Date,Temp,Pres,Volt**
- Make sure there are **no spaces** between the labels. Spelling must be **exactly** as shown.
- **Delete** or **correct** any incomplete data.
- Prepare your **data.txt** file so it looks like the following.

5. **Installing pandas python package to read data from text files**

- To read data from our data.txt file we need to install another python package named **pandas**.
- To install the pandas package open the Terminal and enter the following commands.
    - **pip3 install pandas** – will take 1-2 minutes to install
    - **sudo apt-get update** – update all libraries on the Raspberry Pi
    - **sudo apt-get install libatlas-base-dev** – extra missing software package to support pandas
- To test that pandas has been installed correctly open the **Python Shell** and enter - **import pandas**
- If the pandas python package has installed correctly you should see no errors.



- If the pandas python package has not been installed then you will see the following message in the Python shell.

```
>>> import pandas
Traceback (most recent call last):
  File "<pyshell#0>", line 1, in <module>
    import pandas
ModuleNotFoundError: No module named 'pandas'
>>>
```

6. **Reading data from data.txt using pandas**

- In the temp_plot_gen.py file add the following lines of code.

```
import pandas

df = pandas.read_csv('/home/pi/botanica-park-lake/data.txt')
print(df)
```

```
# temp_plot_gen.py
import plotly.graph_objects as go
import pandas

df = pandas.read_csv('/home/pi/botanica-park-lake/data.txt')
print(df)

mx = [1, 2, 3, 4]
my = [2, 4, 6, 8]

fig = go.Figure(data = go.Scatter(x=mx, y=my))

fig.update_layout(title='Test data',
                  xaxis_title = 'Time',
                  yaxis_title = 'Temperature')

fig.show()
```

- **Save** and **Run** the program.
- The following output will be output in the **Python Shell**.
- The output will show you the number of data entries extracted from the data.txt file.
- All this tabled data is contained within a **python Object** named **df**
- To see the data in the Python Shell we used the statement **print(df)**

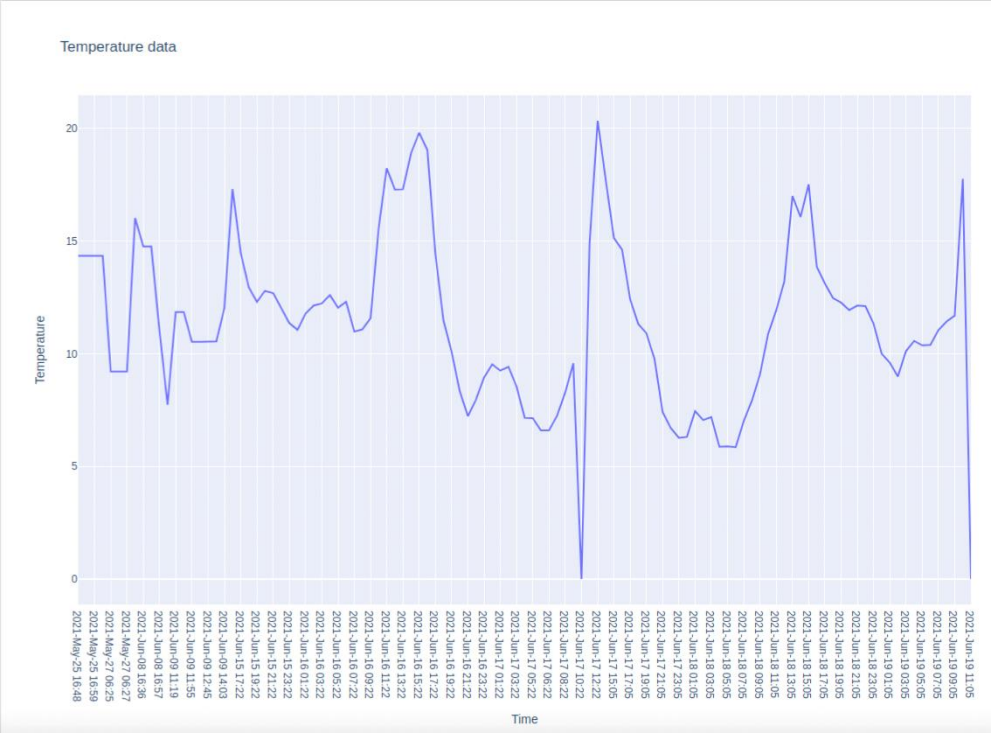
```
=========== RESTART: /home/pi/botanica-park-lake/temp_plot_gen.py ===========
                 Date   Temp     Pres   Volt
0    2021-May-25 16:48  14.34   99.990  4.071
1    2021-May-25 16:57  14.34   99.990  4.071
2    2021-May-25 16:59  14.34   99.990  4.071
3    2021-May-25 17:12  14.34   99.990  4.071
4    2021-May-27 06:25   9.21  101.597  4.055
..                 ...    ...      ...    ...
109  2021-Jun-19 07:05  11.05  101.367  4.063
110  2021-Jun-19 08:05  11.43  101.441  4.059
111  2021-Jun-19 09:05  11.69  101.505  4.060
112  2021-Jun-19 10:05  17.76  101.561  4.062
113  2021-Jun-19 11:05   0.00    0.000  4.065

[114 rows x 4 columns]
>>>
```

7. Graphing data using pandas df Object

- Update your code to include **Date** and **Temp** data from the data.txt file using **pandas df** Object.
    - x=df.Date
    - y=df.Temp
- Update the layout so that the **title** = **'Temperature data'** and **yaxis_title** = **'Temperature'**

```
# temp_plot_gen.py
import plotly.graph_objects as go
import pandas

df = pandas.read_csv('/home/pi/botanica-park-lake/data.txt')
print(df)

fig = go.Figure(data = go.Scatter(x=df.Date,
                                  y=df.Temp))

fig.update_layout(title='Temperature data',
                  xaxis_title = 'Time',
                  yaxis_title = 'Temperature')

fig.show()
```

- In the web browser you should see a plot of the temperature data with data taken from the data.txt file.



ADDING MORE FORMATTING OPTIONS IN PLOTLY

In this example we are going to expand on the formatting properties within plotly.

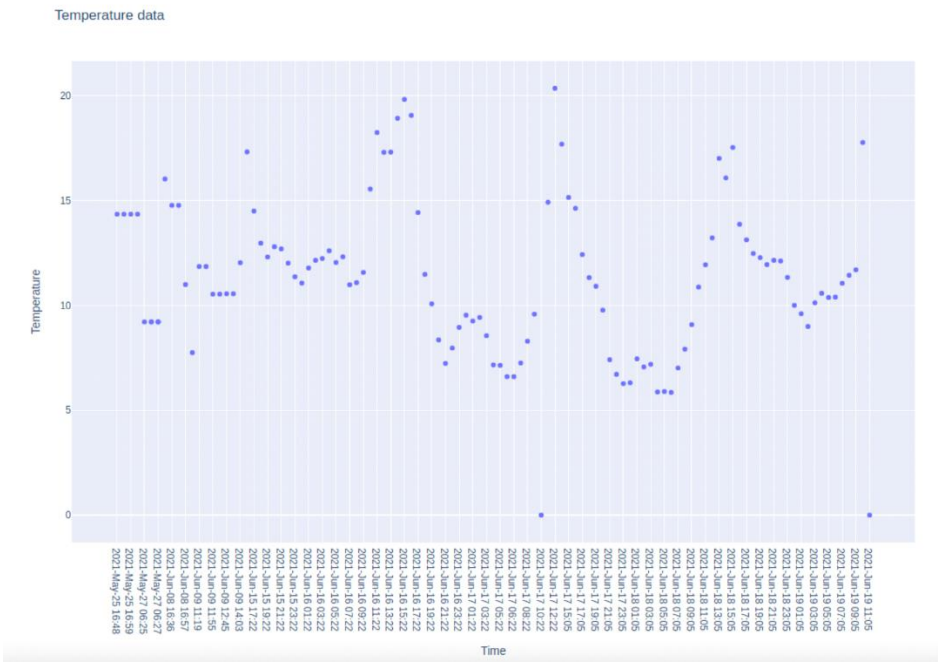
8. Additional formatting options in plotly

- We can replace a line trace with **dots**.
- Update the following **go.Scatter** statement to the following.

```
fig = go.Figure(data = go.Scatter(mode='markers',
                                  x=df.Date,
                                  y=df.Temp))
```

```
# temp_plot_gen.py
import plotly.graph_objects as go
import pandas

df = pandas.read_csv('/home/pi/botanica-park-lake/data.txt')
print(df)

fig = go.Figure(data = go.Scatter(mode='markers',
                                  x=df.Date,
                                  y=df.Temp))

fig.update_layout(title='Temperature data',
                  xaxis_title = 'Time',
                  yaxis_title = 'Temperature')

fig.show()
```

- In the plot below, **lines** have been replaced with **points**.



- We can change the **size** and **colour** of the points.
- Include a **marker dict**ionary with the formatting parameters of **go.Scatter**.

```
fig = go.Figure(data = go.Scatter(mode='markers',
                                  x=df.Date,
                                  y=df.Temp,
                                  marker=dict(
                                      color="LightSkyBlue",
                                      size=20)
                                  )
                )
```

```
# temp_plot_gen.py
import plotly.graph_objects as go
import pandas

df = pandas.read_csv('/home/pi/botanica-park-lake/data.txt')
print(df)

fig = go.Figure(data = go.Scatter(mode='markers',
                                  x=df.Date,
                                  y=df.Temp,
                                  marker=dict(
                                      color="LightSkyBlue",
                                      size=20)
                                  )
                )

fig.update_layout(title='Temperature data',
                  xaxis_title = 'Time',
                  yaxis_title = 'Temperature')

fig.show()
```

- **Save** and **Run** the program. Individual **plotted points** will be **larger** and **blue** in colour.

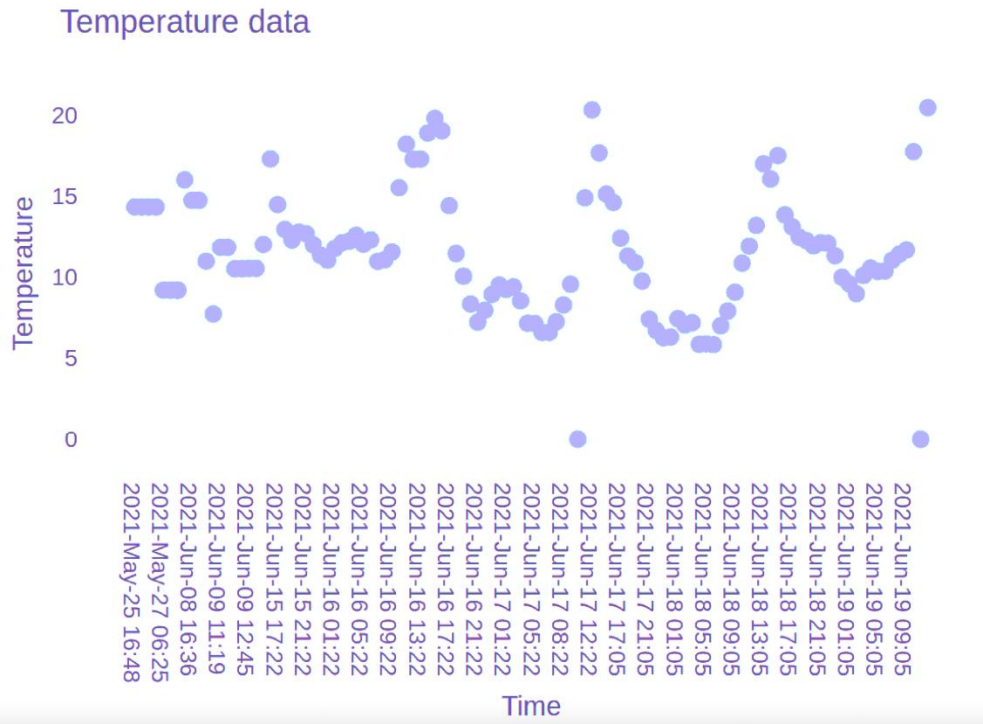
- Now we need to increase the size of the text in the labels for the **title, x-axis and y-axis**.
- Add a **font dict**ionary to the **fig.update_layout** statement.

```
fig.update_layout(title='Temperature data',
                  xaxis_title = 'Time',
                  yaxis_title = 'Temperature',
                  font=dict(
                      size=26,
                      color="RebeccaPurple"
                      )
                  )
```

- The full code is shown below.

```
# temp_plot_gen.py
import plotly.graph_objects as go
import pandas

df = pandas.read_csv('/home/pi/botanica-park-lake/data.txt')
print(df)

fig = go.Figure(data = go.Scatter(mode='markers',
                                  x=df.Date,
                                  y=df.Temp,
                                  marker=dict(
                                      color="LightSkyBlue",
                                      size=20)
                                  )
                )

fig.update_layout(title='Temperature data',
                  xaxis_title = 'Time',
                  yaxis_title = 'Temperature',
                  font=dict(
                      size=26,
                      color="RebeccaPurple"
                      )
                  )
fig.show()
```

- **Save** and **Run** the program. All labels should be **larger** and coloured **purple**.



9. Saving the HTML graph as a file

- Rather than have the graph displayed immediately in a web browser, we can save the graph as a file.
- Once we have a file, we can open it later or link to it on a **web page**.
- Update your code by adding the following statement at the **end of your code**.
- Edit the code to include your **project folder name**.
- The **auto_open=False** statement will prevent the graph from automatically opening up in your Pi Browser.

```
plotly.offline.plot(fig,
                    filename="/home/pi/botanica-park-lake/atm_temp.html",
                    auto_open=False)
```

```
# temp_plot_gen.py
import plotly
import plotly.graph_objects as go
import pandas
df = pandas.read_csv('/home/pi/botanica-park-lake/data.txt')

fig = go.Figure(data = go.Scatter(mode='markers',
                                  x=df.Date,
                                  y=df.Temp,
                                  marker=dict(
                                      color="LightSkyBlue",
                                      size=20)
                                  )
                )
fig.update_layout(title='Temperature data',
                  xaxis_title = 'Time',
                  yaxis_title = 'Temperature',
                  font=dict(
                      size=26,
                      color="RebeccaPurple"
                      )
                  )
plotly.offline.plot(fig,
                    filename="/home/pi/botanica-park-lake/atm_temp.html",
                    auto_open=False)
print("Atmospheric temperature plotted")
```

- To show that the program is complete we have added a **print()** statement at the end of the code.

```
=========== RESTART: /home/pi/botanica-park-lake/temp_plot_gen.py ===========
Atmospheric temperature plotted
>>>
```

- Congratulations. Now you can print all your data, and its also ready to add to your web page.