Project 1: Performing Analysis of Meteorological Data

~ Ankita Komal

Problem Statement: To transform the raw data into information and then convert it into knowledge.

The dataset has hourly temperature recorded for last 10 years starting from 2006-04-01 00:00:00.000 +0200 to 2016-09-09 23:00:00.000 +0200. It corresponds to Finland, a country in the Northern Europe.

Importing all python libraries

In [30]:

```
%matplotlib inline
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import seaborn as sns
```

Importing weatherhistory dataset

In [13]:

```
data=pd.read_csv('C:/Users/Ankita/Desktop/weatherHistory.csv')
```

In [14]:

```
data.shape
```

Out[14]:

```
(96453, 12)
```

In [15]:

```
data.head() #used to get the 1st n rows
```

Out[15]:

	Formatted Date	Summary	Precip Type	Temperature (C)	Apparent Temperature (C)	Humidity	Wind Speed (km/h)	Wind Bearing (degrees)	Vis
0	2006-04-01 00:00:00.000 +0200	Partly Cloudy	rain	9.472222	7.388889	0.89	14.1197	251.0	15
1	2006-04-01 01:00:00.000 +0200	Partly Cloudy	rain	9.355556	7.227778	0.86	14.2646	259.0	15
2	2006-04-01 02:00:00.000 +0200	Mostly Cloudy	rain	9.377778	9.377778	0.89	3.9284	204.0	14
3	2006-04-01 03:00:00.000 +0200	Partly Cloudy	rain	8.288889	5.944444	0.83	14.1036	269.0	15
4	2006-04-01 04:00:00.000 +0200	Mostly Cloudy	rain	8.755556	6.977778	0.83	11.0446	259.0	15

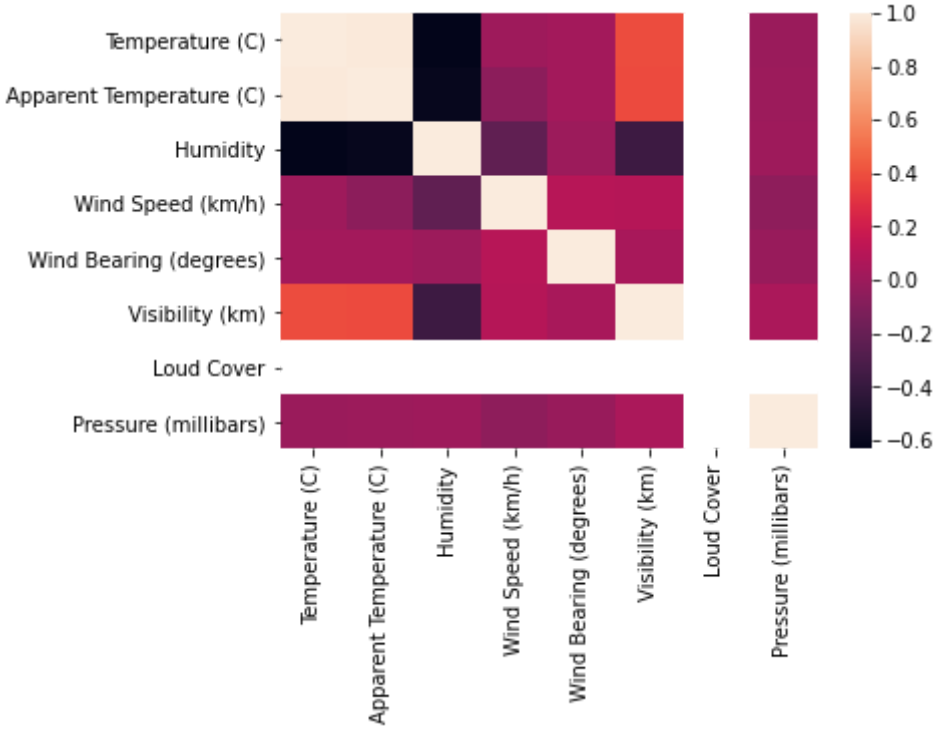
For analysing we are using correlation heatmap as it is ideal for data analysis since it makes patterns easily readable and highlights the differece and vaiation in the data. for mor information about correlation heatmap check [https://www.geeksforgeeks.org/how-to-create-a-seaborn-correlation-heatmap-in-python/#:~:text=The%20data%20here%20has%20to,kaggle.com%20is%20being%20used (https://www.geeksforgeeks.org/how-to-create-a-seaborn-correlation-heatmap-in-python/#:~:text=The%20data%20here%20has%20to,kaggle.com%20is%20being%20used)](https://www.geeksforgeeks.org/how-to-create-a-seaborn-correlation-heatmap-in-python/#:~:text=The%20data%20here%20has%20to,kaggle.com%20is%20being%20used).

In [28]:

```
sns.heatmap(data.corr())
```

Out[28]:

```
<AxesSubplot:>
```

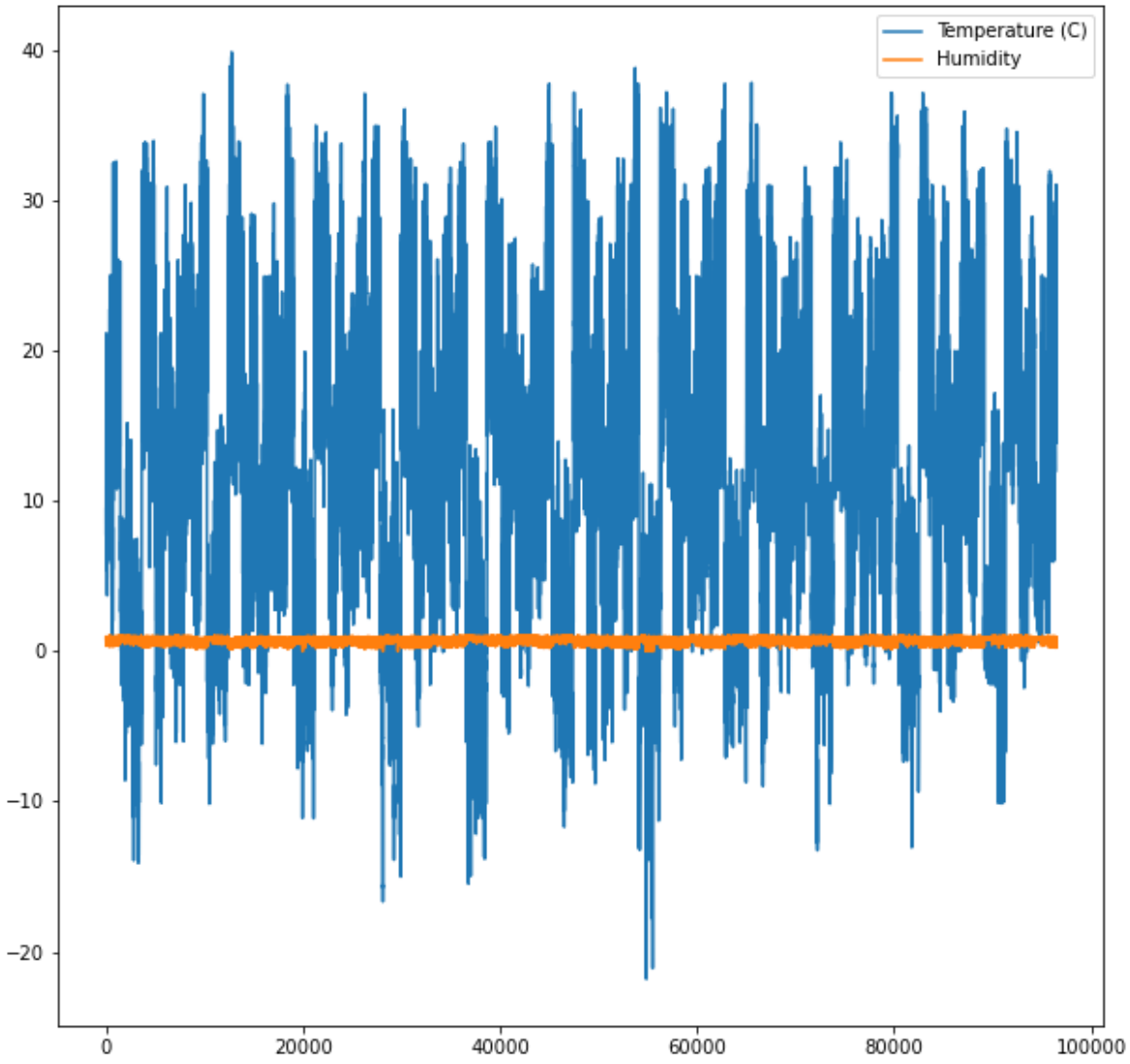


In [18]:

```
plt.figure(figsize=(10,10))
plt.plot(data['Temperature (C)'])
plt.plot(data['Humidity'])
plt.legend(['Temperature (C)','Humidity'])
```

Out[18]:

```
<matplotlib.legend.Legend at 0x2ab44df4d90>
```

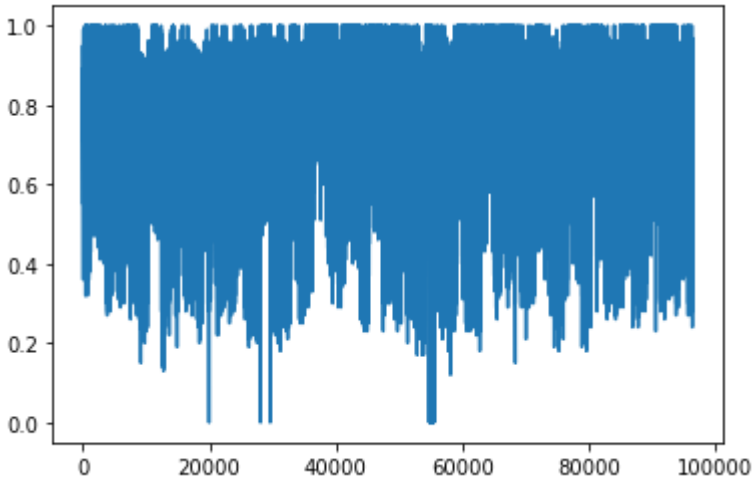


In [19]:

```
plt.figure()
plt.plot(data['Humidity'])
```

Out[19]:

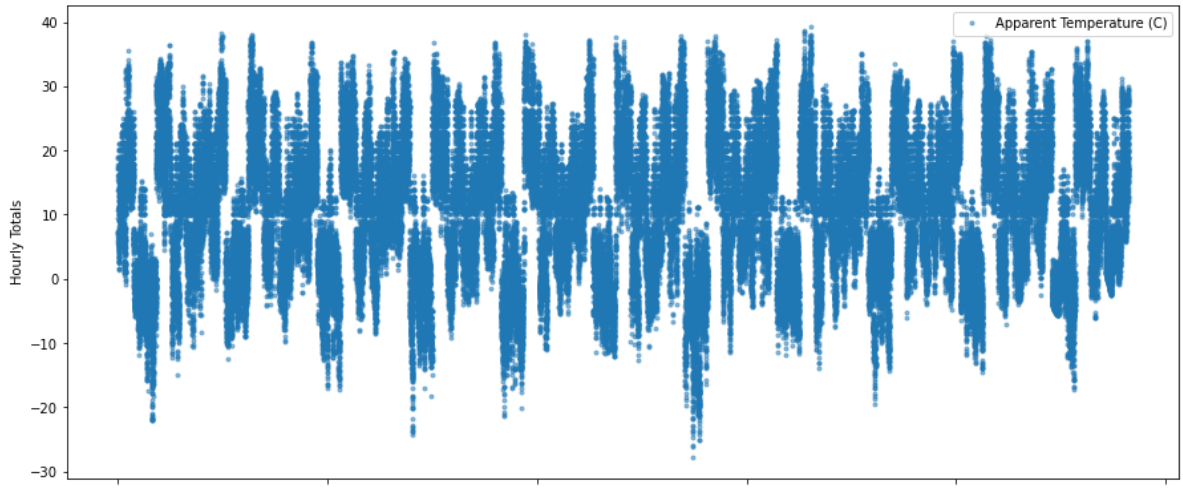
```
[<matplotlib.lines.Line2D at 0x2ab4518bd00>]
```

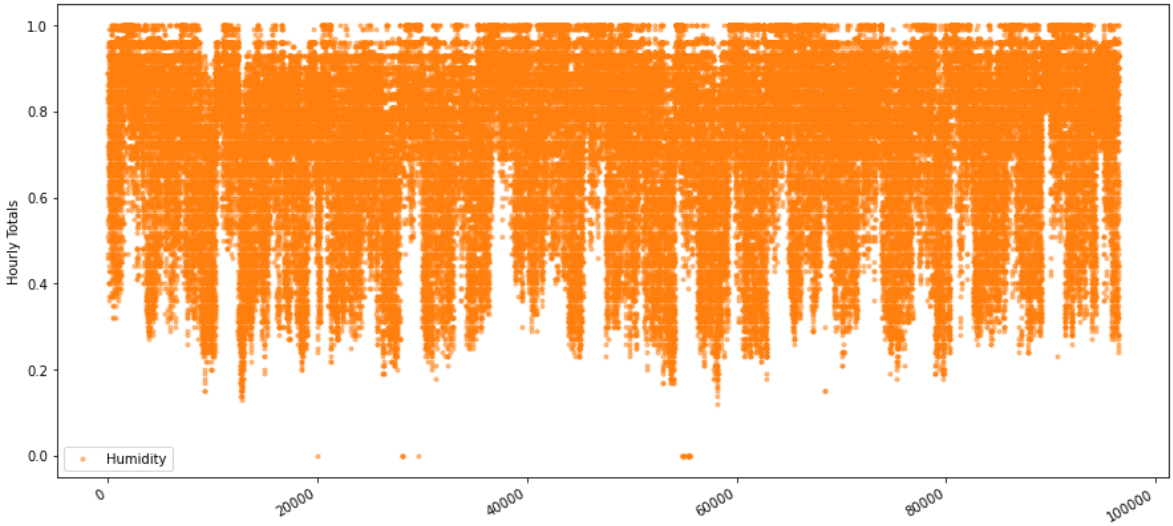


In [20]:

```
cols_plot = ['Apparent Temperature (C)', 'Humidity']

axes = data[cols_plot].plot(marker='.', alpha=0.5, linestyle='None', figsize=(15, 16), subp
for ax in axes:
    ax.set_ylabel('Hourly Totals')
```





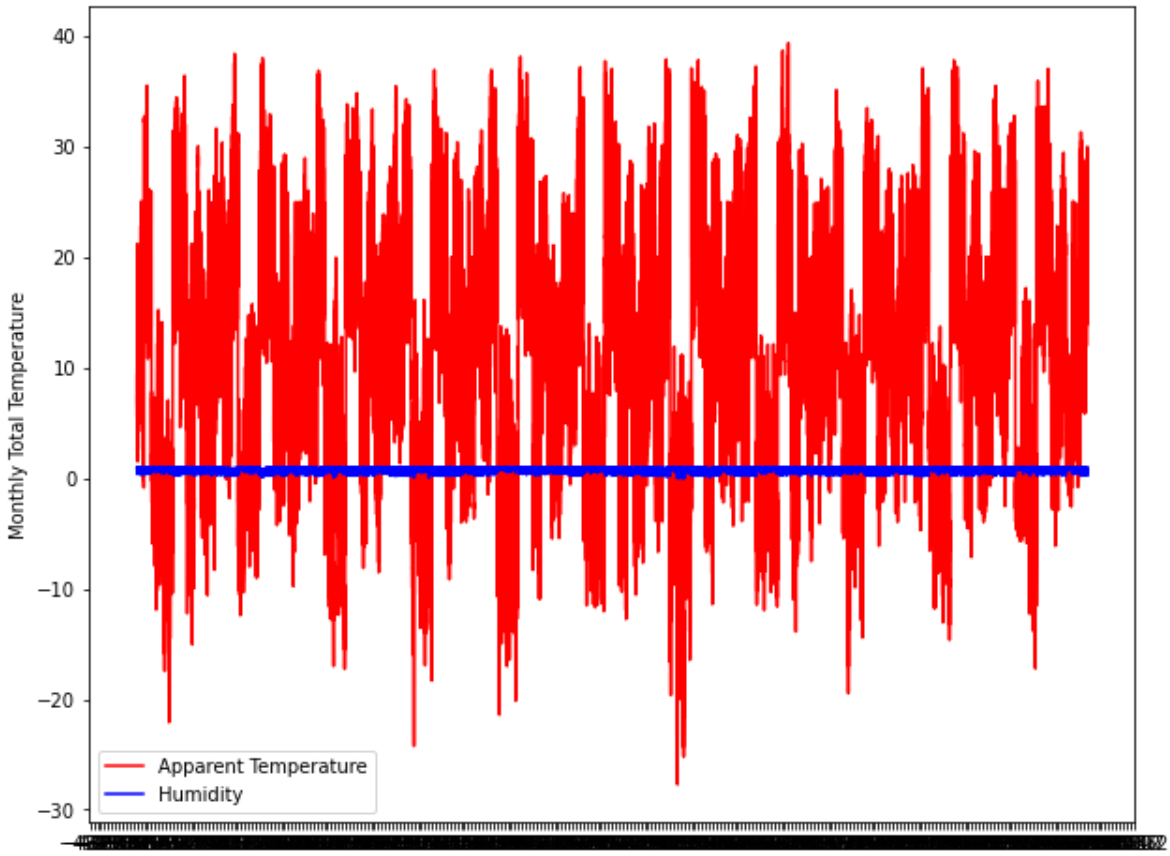
In [21]:

```
import matplotlib.dates as mdates
fig, ax = plt.subplots(figsize=(10,8))
# ax.plot(weather_monthly_mean['Temperature (C)'], color='black', label='Temperature')

ax.plot(data['Apparent Temperature (C)'], color='red', label='Apparent Temperature')

ax.plot(data['Humidity'], color='blue', label='Humidity')

#data[['Apparent Temperature (C)', 'Humidity']].plot.area(ax=ax, linewidth=0)
ax.xaxis.set_major_locator(mdates.YearLocator())
ax.legend()
ax.set_ylabel('Monthly Total Temperature');
```



In [22]:

```
data_columns = ['Temperature (C)', 'Apparent Temperature (C)', 'Humidity']


# Resample to weekly frequency, aggregating with mean
weather_monthly_mean = data[data_columns].mean()
weather_monthly_mean.tail(20)
```

Out[22]:

```
Temperature (C)             11.932678
Apparent Temperature (C)    10.855029
Humidity                     0.734899
dtype: float64
```

In [23]:

```
data.describe()
```

Out[23]:

	Temperature (C)	Apparent Temperature (C)	Humidity	Wind Speed (km/h)	Wind Bearing (degrees)	Visibility (km)	
count	96453.000000	96453.000000	96453.000000	96453.000000	96453.000000	96453.000000	96
mean	11.932678	10.855029	0.734899	10.810640	187.509232	10.347325	
std	9.551546	10.696847	0.195473	6.913571	107.383428	4.192123	
min	-21.822222	-27.716667	0.000000	0.000000	0.000000	0.000000	
25%	4.688889	2.311111	0.600000	5.828200	116.000000	8.339800	
50%	12.000000	12.000000	0.780000	9.965900	180.000000	10.046400	
75%	18.838889	18.838889	0.890000	14.135800	290.000000	14.812000	
max	39.905556	39.344444	1.000000	63.852600	359.000000	16.100000	

In [24]:

```
data[data_columns].head()
```

Out[24]:

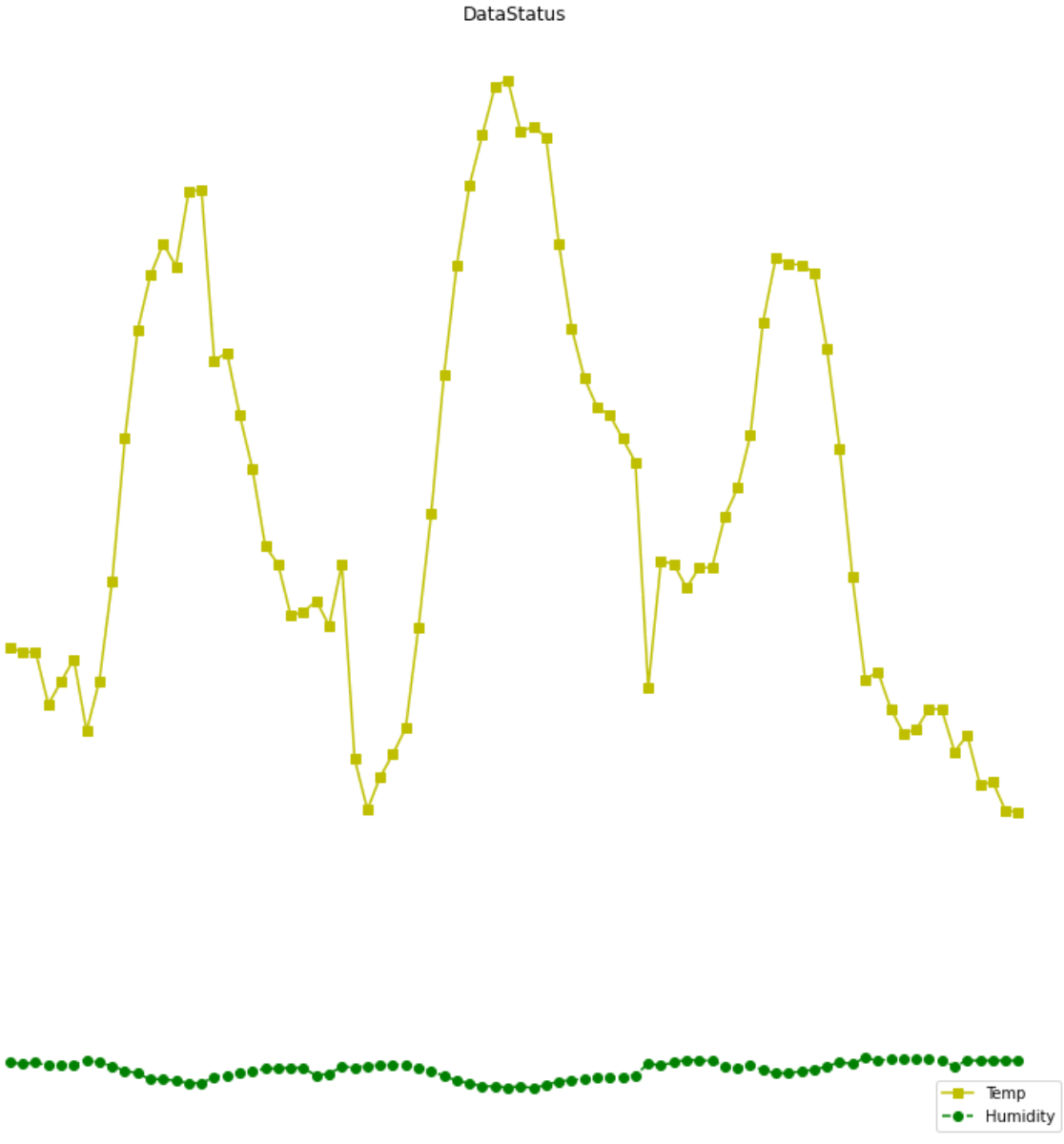
	Temperature (C)	Apparent Temperature (C)	Humidity
0	9.472222	7.388889	0.89
1	9.355556	7.227778	0.86
2	9.377778	9.377778	0.89
3	8.288889	5.944444	0.83
4	8.755556	6.977778	0.83

In [25]:

```
y = data['Formatted Date'][:80]
x1 = data['Temperature (C)'][:80]
x2 = data['Humidity'][:80]
fig = plt.figure(figsize=(10,10))
ax = fig.add_axes([0,0,1,1])
l1 = ax.plot(y,x1,'ys-') # solid line with yellow colour and square marker
l2 = ax.plot(y,x2,'go--') # dash line with green colour and circle marker
ax.legend(labels = ('Temp', 'Humidity'), loc = 'lower right') # legend placed at lower righ
ax.set_title("DataStatus")
ax.set_ylabel('Monthly temp total')
ax.set_xlabel('Date')
ax.axis('off')
```

Out[25]:

```
(-3.95, 82.95, -0.6811666666666664, 22.224499999999992)
```



In [26]:

```
y = data['Formatted Date'][96400:]
x1 = data['Temperature (C)'][96400:]
x2 = data['Humidity'][96400:]
fig = plt.figure(figsize=(10,10))
ax = fig.add_axes([0,0,1,1])
l1 = ax.plot(y,x1,'ys-') # solid line with yellow colour and square marker
l2 = ax.plot(y,x2,'go--') # dash line with green colour and circle marker
ax.legend(labels = ('Temp', 'Humidity'), loc = 'lower right') # legend placed at lower righ
ax.set_title("DataStatus")
ax.set_ylabel('Monthly temp total')
ax.set_xlabel('Date')
ax.axis('off')
```

Out[26]:

```
(-2.6, 54.6, -1.3021666666666667, 32.625499999999995)
```