

## Visualize GDD data using a Jupyter Notebook

In this lesson you gathered GDD data using an IoT sensor. To get good GDD data, you need to gather data for multiple days. To help visualize temperature data and calculate GDD you can use tools like [Jupyter Notebooks](#) to analyze the data.

Start by gathering data for a few days. You will need to ensure your server code is running all the time your IoT device is running, either by adjusting your power management settings or running something like [this keep system active Python script](#).

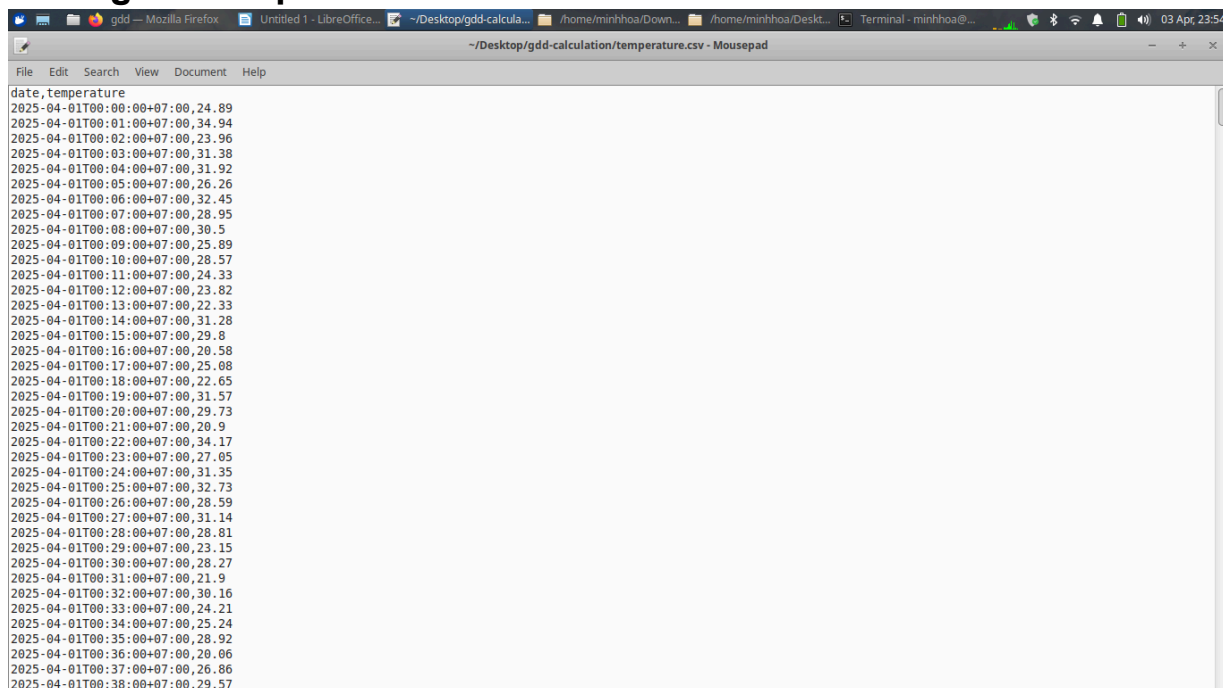
Once you have temperature data, you can use the Jupyter Notebook in this repo to visualize it and calculate GDD. Jupyter notebooks mix code and instructions in blocks called *cells*, often code in Python. You can read the instructions, then run each block of code, block by block. You can also edit the code. In this notebook for example, you can edit the base temperature used to calculate the GDD for your plant.

### Answer:

In this exercise, I collect temperature information for 1 hour. Each measurement is about 1 minute apart.

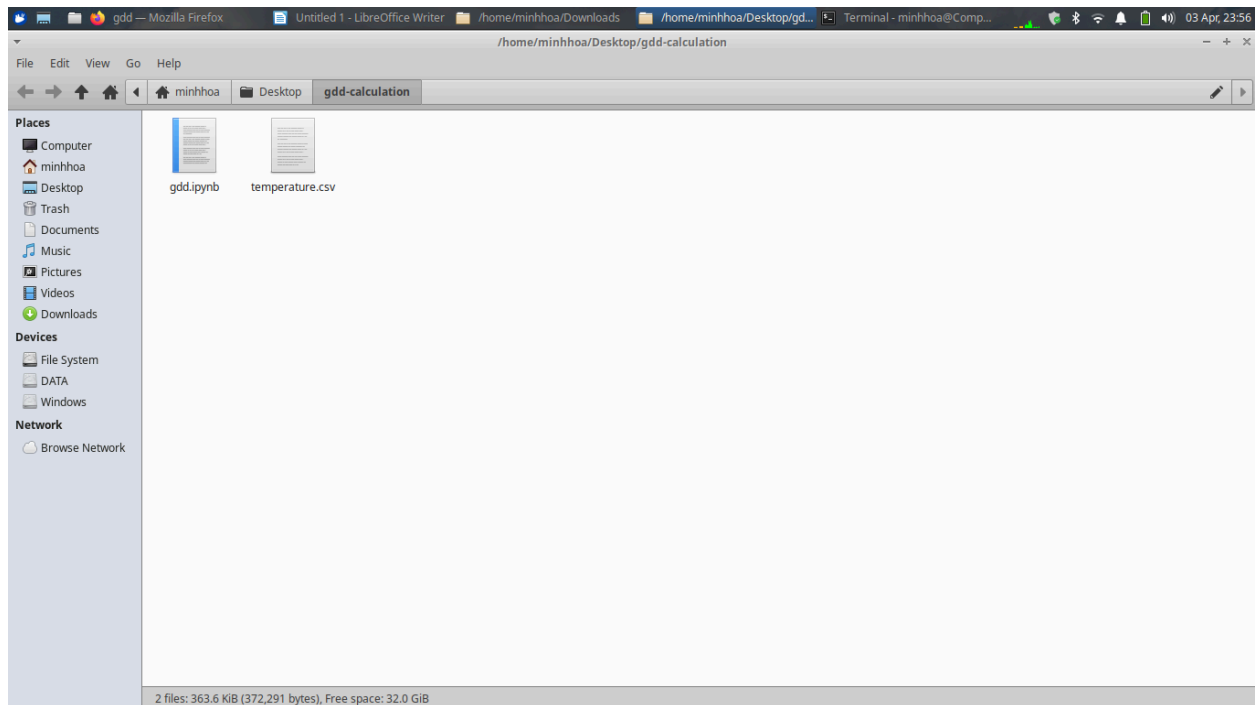
The gdd.ipynb and temperature.csv files are saved in the same folder and that folder is called gdd-calculation.

### 1. Image of temperature.csv file

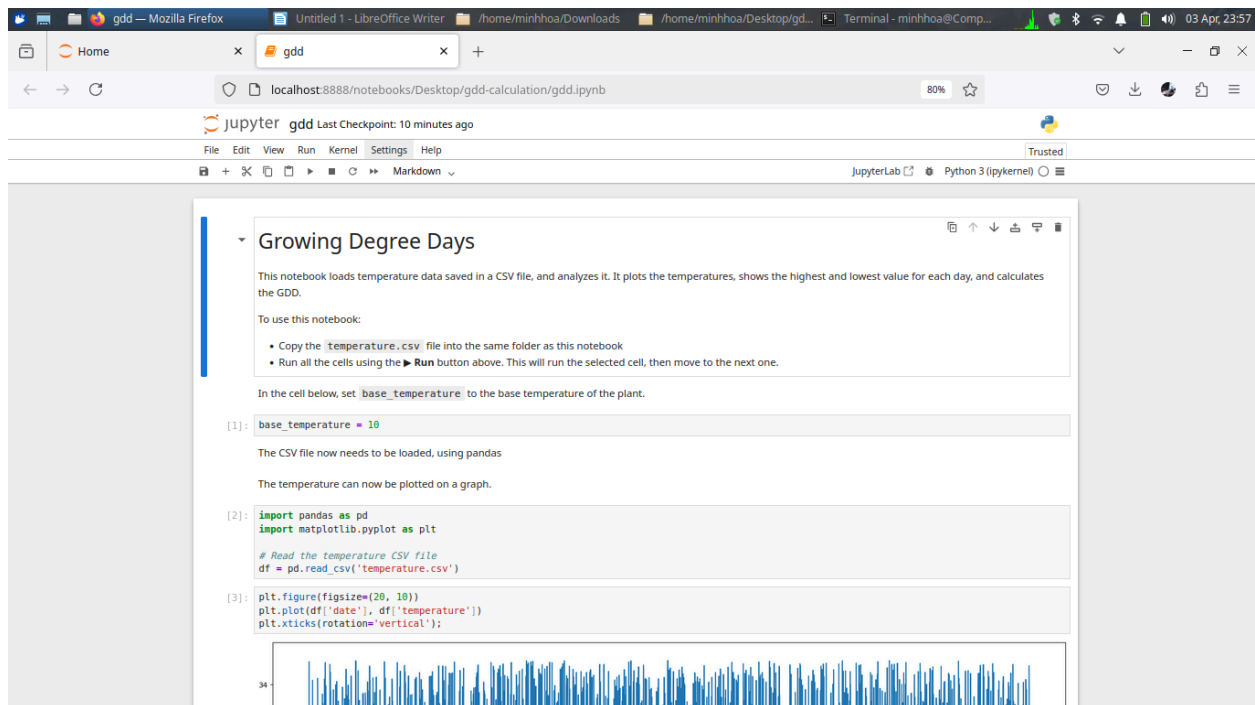


```
date,temperature
2025-04-01T00:00:00+07:00,24.89
2025-04-01T00:01:00+07:00,34.94
2025-04-01T00:02:00+07:00,23.96
2025-04-01T00:03:00+07:00,31.38
2025-04-01T00:04:00+07:00,31.92
2025-04-01T00:05:00+07:00,26.26
2025-04-01T00:06:00+07:00,32.45
2025-04-01T00:07:00+07:00,28.95
2025-04-01T00:08:00+07:00,30.5
2025-04-01T00:09:00+07:00,25.89
2025-04-01T00:10:00+07:00,28.57
2025-04-01T00:11:00+07:00,24.33
2025-04-01T00:12:00+07:00,23.82
2025-04-01T00:13:00+07:00,22.33
2025-04-01T00:14:00+07:00,31.28
2025-04-01T00:15:00+07:00,29.8
2025-04-01T00:16:00+07:00,20.58
2025-04-01T00:17:00+07:00,25.08
2025-04-01T00:18:00+07:00,22.65
2025-04-01T00:19:00+07:00,31.57
2025-04-01T00:20:00+07:00,29.73
2025-04-01T00:21:00+07:00,20.9
2025-04-01T00:22:00+07:00,34.17
2025-04-01T00:23:00+07:00,27.05
2025-04-01T00:24:00+07:00,31.35
2025-04-01T00:25:00+07:00,32.73
2025-04-01T00:26:00+07:00,28.59
2025-04-01T00:27:00+07:00,31.14
2025-04-01T00:28:00+07:00,28.81
2025-04-01T00:29:00+07:00,23.15
2025-04-01T00:30:00+07:00,28.27
2025-04-01T00:31:00+07:00,21.9
2025-04-01T00:32:00+07:00,30.16
2025-04-01T00:33:00+07:00,24.21
2025-04-01T00:34:00+07:00,25.24
2025-04-01T00:35:00+07:00,28.92
2025-04-01T00:36:00+07:00,20.06
2025-04-01T00:37:00+07:00,26.86
2025-04-01T00:38:00+07:00,29.57
```

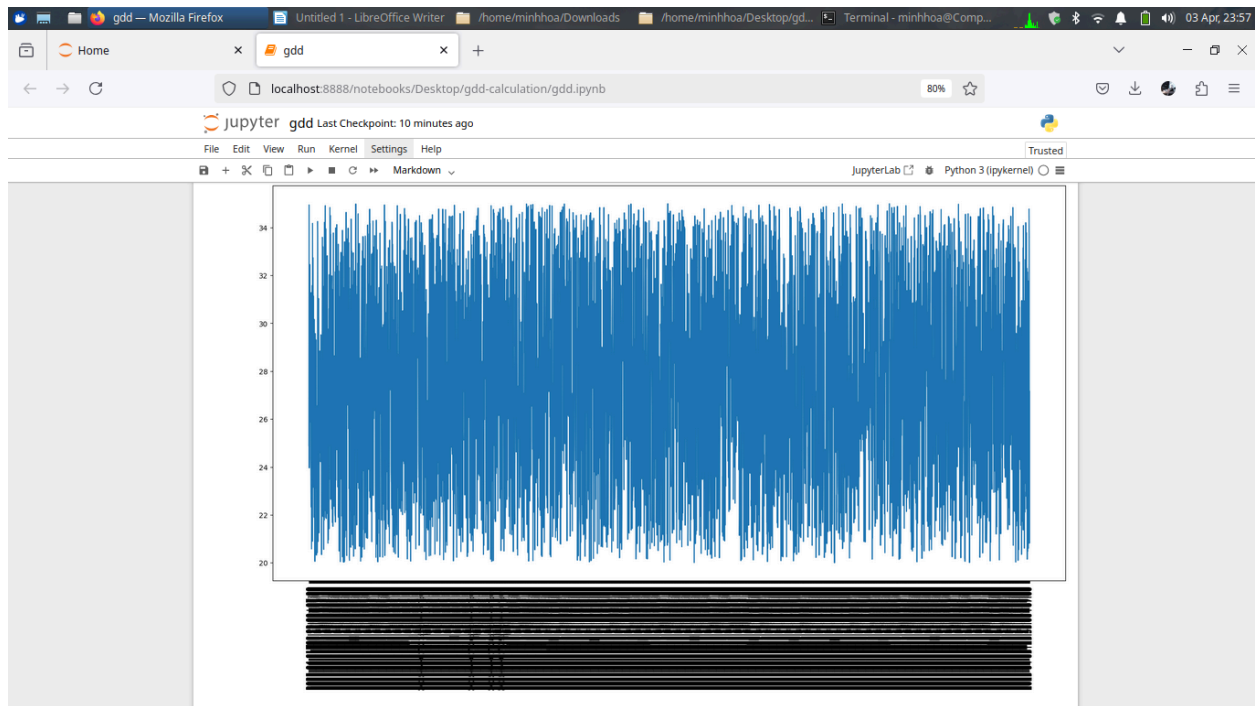
## 2. Image of gdd-calculation folder



## 3. Image when launching gdd.ipynb file



## 4. Temperature and time of day chart



## 5. Gcd calculation

```
Once the data has been read it can be grouped by the date column, and the minimum and maximum temperatures extracted for each date.

[5]: # Convert datetimes to pure dates so we can group by the date
df['date'] = pd.to_datetime(df['date']).dt.date

# Group the data by date so it can be analyzed by date
data_by_date = df.groupby('date')

# Get the minimum and maximum temperatures for each date
min_by_date = data_by_date.min()
max_by_date = data_by_date.max()

# Join the min and max temperatures into one dataframe and flatten it
min_max_by_date = min_by_date.join(max_by_date, on='date', lsuffix='_min', rsuffix='_max')
min_max_by_date = min_max_by_date.reset_index()

The GDD can be calculated using the standard GDD equation

[6]: def calculate_gdd(row):
      return ((row['temperature_max'] + row['temperature_min']) / 2) - base_temperature

# Calculate the GDD for each row
min_max_by_date['gdd'] = min_max_by_date.apply(lambda row: calculate_gdd(row), axis=1)

# Print the results
print(min_max_by_date[['date', 'gdd']].to_string(index=False))

date    gdd
2025-04-01 17.585
2025-04-02 17.500
2025-04-03 17.495

[ ]:
```