Farm

Assignment 1

Growing Degree Days

This notebook loads temperature data saved in a CSV file, and analyzes it. It plots the temperatures, shows the highest and lowest value for each day, and calculates the GDD.

To use this notebook:

- Copy the `temperature.csv` file into the same folder as this notebook
- Run all the cells using the ▶ **Run** button above. This will run the selected cell, then move to the next one.

In the cell below, set `base_temperature` to the base temperature of the plant.

```
[1]
base_temperature = 10
```
✓ 0.0s Python

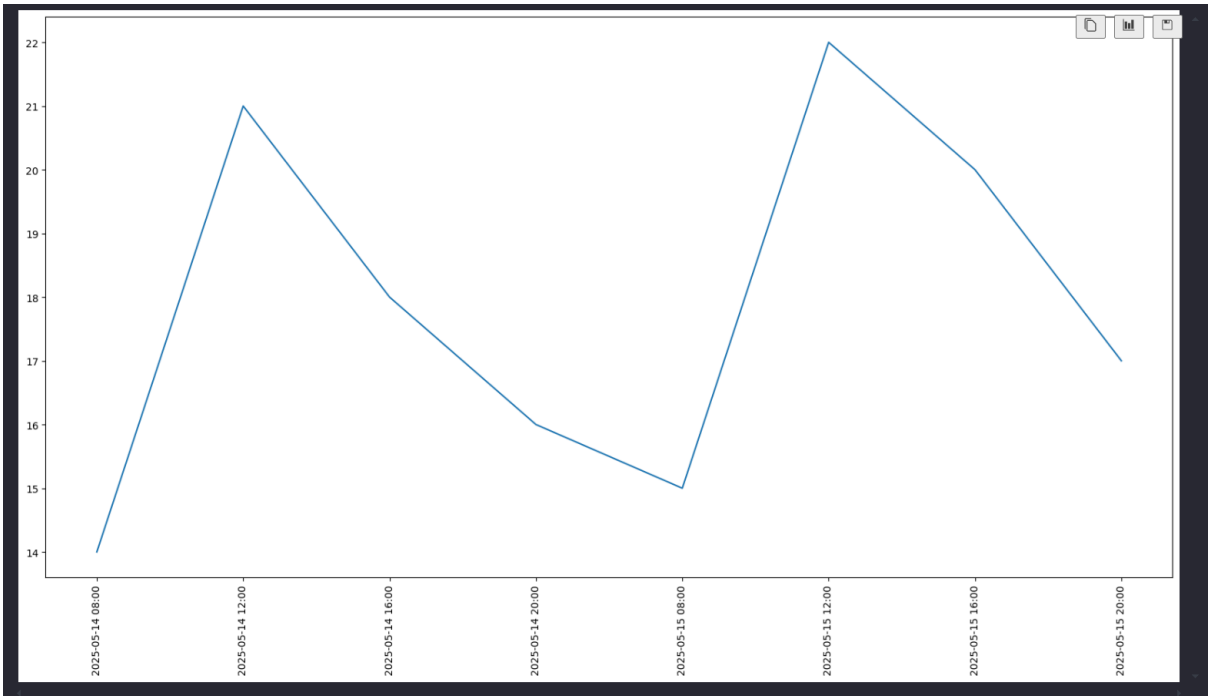
The CSV file now needs to be loaded, using pandas

```
[2]
import pandas as pd
import matplotlib.pyplot as plt

# Read the temperature CSV file
df = pd.read_csv('temperature.csv')
```
✓ 0.8s Python

The temperature can now be plotted on a graph.

```
plt.figure(figsize=(20, 10))
plt.plot(df['date'], df['temperature'])
plt.xticks(rotation='vertical');
```
✓ 0.1s Python



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

```python
# 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()
```

[4] ✓ 0.0s Python

The GDD can be calculated using the standard GDD equation

```python
Tabnine | Edit | Test | Explain | Document
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))
```

[5] ✓ 0.0s Python

```
      date  gdd
2025-05-14  7.5
2025-05-15  8.5
```

Assignment 2

Sample	Wet Weight (g)	Dry Weight (g)	Soil Moisture (%)	Sensor Reading
1	250	220	(250 - 220) / 220 x 100 = 13.6%	630
2	265	225	(265 - 225) / 225 x 100 = 17.8%	710
3	240	215	(240 - 215) / 215 x 100 = 11.6%	590

Assignment 3

Total Pump Time	Soil Moisture	Decrease
Dry	660	0
1s	637	23
2s	615	22
3s	592	23
4s	569	23
5s	547	22
6s	525	22

Total decrease: 23 + 22 + 23 + 23 + 22 + 22 = 135
Number of intervals = 6
Average decrease per second = 135 / 6 = 22.5 units of soil moisture per second of pumping

Assignment 4

Explanation of the Different Cloud Offering:

1. **Infrastructure as a Service (IaaS):**
   This is the most basic type of cloud service. It provides virtualized computing resources over the internet, such as virtual machines, storage, and networks. Developers manage everything from the OS up.
   -> Useful for full control over the environment.
2. **Platform as a Service (PaaS):**
   PaaS offers a ready-made environment for developers to build, run, and manage applications without worrying about the underlying infrastructure.
   -> Helps save time on setup and maintenance.
3. **Serverless Computing:**
   With Serverless, developers write code and deploy it without managing servers. The cloud provider automatically handles the infrastructure and scaling.
   -> Ideal for event-driven applications and microservices.
4. **Software as a Service (SaaS):**
   SaaS delivers software applications over the internet on a subscription basis. Users can access them through a browser without installing or maintaining them.
   -> Perfect for end-users and productivity tools (e.g., Microsoft 365).

Which Offerings Are Relevant for IoT Developers (and Why):

1. **IaaS** is relevant for IoT developers who need to set up custom environments or run virtual machines to manage large data from IoT devices.

-> It gives control over networking and storage needed for processing sensor data.

2. **PaaS** is highly relevant because it allows IoT developers to focus on applications logic while the platform handles scaling and updates.
   -> Good for rapid prototyping and deploying IoT services like device data processing or APIs.
3. **Serverless** is especially useful in IoT for event-driven architectures, such as when devices trigger actions (e.g, an alert when temperature is too high).
   -> It automatically scales and is cost-efficient, especially with many small device interactions.
4. **SaaS** may be less critical for developers, but useful for providing dashboards or analytics tools that IoT systems can integrate with.
   -> Helps users view and manage their IoT data.