Welcome to my IND320 Jupyter notebook

Introduction

In this notebook, I work with the dataset `open-meteo-subset.csv` in order to explore different visualization methods. The objectives are:

- Read and print the dataset
- Plot each column separately
- Plot all columns together in a meaningful way
- Prepare for the Streamlit app

Reading the supplied CSV file using Pandas and printing its contents in a relevant way.

In [18]:
```
import pandas as pd
import matplotlib.pyplot as plt

df = pd.read_csv("../data/open-meteo-subset.csv")
df.head()
```

Out[18]:

	time	temperature_2m (°C)	precipitation (mm)	wind_speed_10m (m/s)	wind_gusts_10m (m/s)	wind_direction_10m (°)
0	2020-01-01T00:00	-2.2	0.1	9.6	21.3	284
1	2020-01-01T01:00	-2.2	0.0	10.6	23.0	282
2	2020-01-01T02:00	-2.3	0.0	11.0	23.5	284
3	2020-01-01T03:00	-2.3	0.0	10.6	23.3	284
4	2020-01-01T04:00	-2.7	0.0	10.6	22.8	284

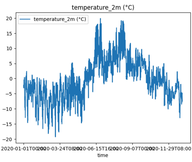
I showed here all the columns

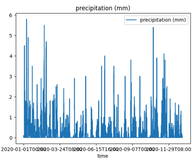
In [19]:
```
for i, col in enumerate(df.columns):
    print(f"{i}: {col}")
```

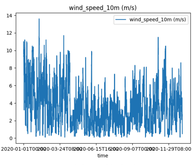
```
0: time
1: temperature_2m (°C)
2: precipitation (mm)
3: wind_speed_10m (m/s)
4: wind_gusts_10m (m/s)
5: wind_direction_10m (°)
```

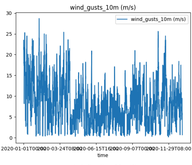
I plot each column separately.

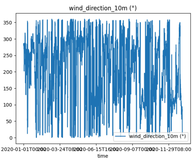
In [20]:
```
for col in df.columns[1:]:  # without the "time" column
    df.plot(x="time", y=col, title=col)
    plt.show()
```











Ploting all columns together.

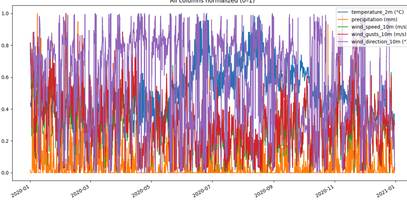
In [21]:
```
from sklearn.preprocessing import MinMaxScaler

# Convert "time" to datetime for better handling of the x-axis
df["time"] = pd.to_datetime(df["time"])

# Normalize all columns (except time) so that variables with different units (°C, mm, m/s, °)
# can be plotted together on the same scale (0-1).
# This allows comparing trends rather than absolute values.
scaler = MinMaxScaler()
scaled = scaler.fit_transform(df.iloc[:, 1:])
scaled_df = pd.DataFrame(scaled, columns=df.columns[1:])
scaled_df["time"] = df["time"]

# Plot all normalized columns
plt.figure(figsize=(12, 6))
for col in scaled_df.columns[:-1]:
    plt.plot(scaled_df["time"], scaled_df[col], label=col)

plt.gcf().autofmt_xdate()  # auto-format x-axis labels for dates
plt.legend()
plt.title("All columns normalized (0-1)")
plt.tight_layout()
plt.show()
```



Observation

When plotting all columns together over the entire time period, the figure becomes overcrowded and difficult to interpret. Although normalization puts the variables on the same scale (0–1), the large number of data points and the different dynamics of each variable make the plot unreadable.

Therefore, it is useful to consider alternative visualizations that preserve the idea of showing all variables together but in a more interpretable way. Examples include:

Focusing on a shorter time window (e.g., one month instead of the full year) to reduce clutter and highlight short-term relationships.

Applying smoothing (rolling averages) to reduce noise and reveal clearer trends.

Using correlation plots to summarize relationships numerically rather than plotting every single data point.

These approaches make it possible to compare variables "together" while also providing insights that are not visible in the raw, overcrowded plot.

Correlation matrix

In [112]:
```
corr = df.drop(columns="time").corr()

plt.figure(figsize=(8,6))
plt.imshow(corr, cmap="coolwarm", vmin=-1, vmax=1)
plt.colorbar(label="Correlation")
plt.xticks(range(len(corr.columns)), corr.columns, rotation=45, ha="right")
plt.yticks(range(len(corr.columns)), corr.columns)
plt.title("Correlation matrix of variables")
plt.tight_layout()
plt.show()
```

Observation

The correlation matrix shows that wind speed and wind gusts are almost perfectly correlated, which is expected as gusts are short-term peaks of wind speed. Precipitation is moderately correlated with both wind speed and gusts, suggesting that windy conditions often come with rain. Temperature, however, shows almost no linear correlation with the other variables, and wind direction appears largely independent of the other measurements.

In [ ]:

Focusing on a shorter time window

In [113]:
```
fig, ax1 = plt.subplots(figsize=(12,6))

# Temperature, precipitation, wind speed, gusts
for col in ["temperature_2m (°C)", "precipitation (mm)", "wind_speed_10m (m/s)", "wind_gusts_10m (m/s)"]:
    ax1.plot(subset["time"], subset[col], label=col)
ax1.set_ylabel("Temperature / Wind / Precipitation")

# Wind direction on secondary axis
ax2 = ax1.twinx()
ax2.plot(subset["time"], subset["wind_direction_10m (°)"], color="purple", label="wind_direction_10m (°)")
ax2.set_ylabel("Wind direction (°)")

fig.legend(loc="upper right")
plt.title("Variables – June 2020 with dual axis")
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

Project Log – IND320: Data to Decision

This first part of the IND320 project focused on learning how to handle a dataset in Jupyter Notebook, visualize it in different ways, and prepare for the Streamlit application.

I started by importing the supplied CSV file using pandas. The dataset contained several columns: time, temperature, precipitation, wind speed, wind gusts, and wind direction. At first, I printed the contents of the DataFrame to verify the data was loaded correctly. Then, I generated individual plots for each variable. For continuous variables like temperature and wind speed, line plots were most suitable, while for precipitation I tested bar plots since rainfall is often represented as discrete events.

The next challenge was plotting all columns together. Because the variables have different scales and units, the raw combined plot was unreadable. I tried two different strategies: (1) normalization of all values between 0 and 1 to compare trends, and (2) using a dual axis for wind direction so that the other variables were still visible in their original units. Both approaches improved readability, and this process taught me the importance of choosing the right visualization depending on the question I want to answer.

Additionally, I experimented with correlation matrices to explore linear relationships between variables. As expected, wind speed and wind gusts showed very strong correlation, while temperature had little direct relationship with other variables. This confirmed that some visualizations are better at highlighting patterns that are hidden in raw plots.

Regarding the use of AI, I worked with GitHub Copilot in VS Code and ChatGPT to support my coding and documentation. Copilot helped me quickly write plotting loops and boilerplate code, while ChatGPT was useful for explaining why some visualization choices (e.g., normalization vs. multiple axes) are more appropriate than others. The AI tools did not replace my work but accelerated my understanding and gave me alternative approaches when I was stuck. For example, I would not have thought about applying rolling averages or correlation matrices without this guidance.

So far, I have learned how to structure my notebook with clear headings, comment code properly, and export results. The next step is to integrate these insights into a Streamlit application that will allow interactive exploration of the data.

GitHub Repository: https://github.com/JulesSyMJAMBA/IND320_DataTo_Decision
Streamlit App: https://share.streamlit.io/user/julessymumba

In [ ]: