Python Corner 2026 — Session 1

Environmental Data Analysis with Python

About Me

Tom Lotz

Geographer, data scientist, and Python user based at Jinling Institute of Technology since 2021.

Background

Geography degree from Germany, PhD completed in China

Python Uses

Environmental analysis, modeling, machine learning, and scientific research

What Is Python Corner?

A hands-on, informal space to practice Python on real environmental and societal datasets — together.

Real Datasets

Work with actual environmental and societal data, not toy examples

Practical Skills

Focus on applied data analysis techniques you can use in research

Open Discussion

Informal format that welcomes questions and different skill levels

Prerequisite: Participants should already know basic Python syntax. This corner focuses on applications, not programming instruction from scratch.

Why Learn Data Analysis?

Data analysis is a core skill across many modern disciplines — from climate science to business intelligence.

Find patterns

Identify trends, cycles, and anomalies hidden in raw numbers

Detect extreme events

Spot floods, heatwaves, or market crashes in the record

Build models

Understand and predict behavior using statistical and ML approaches

Today Is a Pilot Session

We're figuring out what works best — together. Your feedback shapes the future of Python Corner.

Is the pace right?

Not too fast, not too slow

Do the exercises work?

Practical and achievable in session

What's the group's Python level?

So we can calibrate going forward

What We're Doing Today

One dataset. One plot. Lots of exploration.

Set Up

Configure your Python environment and install the required libraries

Load Data

Import a real-world river runoff dataset into Python

Explore

Inspect the structure, statistics, and patterns in the data

Visualize

Create clear, labeled plots to communicate what the data shows

The goal is not to cover many topics — it's to get comfortable working with real data.

SETUP

Choose Your Python Environment

Use whatever editor you're most comfortable with — all of them can run Python scripts and install libraries.

Visual Studio Code

Popular, highly extensible, great Python support via extensions

PyCharm

Feature-rich IDE built specifically for Python development

Python IDLE

Lightweight, built-in editor — no installation needed beyond Python itself

SETUP

Required Libraries

Today we use three essential Python libraries. Think of libraries as toolboxes of pre-written code — you import them and use their functions directly.

pandas

Load, inspect, and manipulate tables and datasets

matplotlib

Create graphs, time series plots, and custom visualizations

numpy

Numerical calculations, arrays, and mathematical operations

SETUP

Install the Libraries

If any library is not yet installed, open a terminal and run the following command:

```
pip install pandas
```

```
pip install matplotlib
```

```
pip install numpy
```

Once installed, these libraries will be available to import in any Python script on your machine. You only need to do this once.

Semester Outlook

Python Corner will grow with the group. Here's a preview of where we're heading.

1

Session 1

Load, explore, and plot river runoff data

2

Early Sessions

Exploratory data analysis and visualization techniques

3

Mid Semester

Time series analysis and simple models

4

Later

Independent mini-projects with a dataset of your choice

CONCEPTS

What Is Environmental Data?

Environmental data captures measurements of the natural world — recorded by sensors, satellites, or field instruments over time.

Most environmental datasets are time series — measurements recorded at regular intervals over an extended period.

- **River discharge & rainfall**
- **Temperature & wind speed**
- **Air pollution & ocean data**

CONCEPTS

What Is Societal & Economic Data?

Societal and economic data captures human activities, demographics, and financial trends, offering insights into how societies function and evolve.

Like environmental data, these datasets often form time series, allowing us to track changes, growth, or decline over periods.

- **Population growth & migration**
- **GDP & inflation rates**
- **Employment & income levels**

Today's Dataset: Mississippi River Runoff

What is Runoff?

The volume of water flowing through the river per second, measured in cubic meters per second (m^3/s)

Scale

Large rivers like the Mississippi can carry tens of thousands of m^3/s — especially during flood events

SETUP

Download the Dataset

Get the data file here and save it in the same folder as your Python script:

> https://t-l.earth/teaching/files/python_corner_2026/runoff_data.csv

Keeping your script and data file in the same folder is the simplest way to avoid file path errors when loading the dataset.

CONCEPTS

What Is a CSV File?

CSV = Comma Separated Values. A plain text file that stores tabular data — one row per line, values separated by commas.

```
date,daily_runoff_m3s
2010-01-01,15200
2010-01-02,15800
2010-01-03,14900
```

CSV is one of the most widely used formats for sharing datasets across tools and programming languages.

1 First row

Column names (the header)

2 Each following row

One observation or measurement

CODE

Start Your Python Script

Create a new file called `runoff_analysis.py` and add the following import at the top:

```
import pandas as pd
```

Run the script once to confirm your environment is working and no errors appear. If it runs silently with no output, you're good to go.

CODE

Load the Dataset

Use pandas to read the CSV file into a DataFrame:

```
df = pd.read_csv("runoff_data.csv")
```

pd

The pandas library, imported with the alias `pd`

read_csv()

A pandas function that reads a CSV file and returns a table

df

Short for DataFrame — pandas' core table structure, with rows and columns

CODE

Look at the Data

Before doing anything else, always inspect the first few rows to understand what you're working with:

```
print(df.head())
```

head() returns the first 5 rows by default. You can pass a number — e.g. `df.head(10)` — to see more. This is your first sanity check: does the data look as expected?

CONCEPTS

Data Structure: Rows and Columns

In a tabular dataset:

Columns = Variables

Each column holds one type of measurement (e.g., date or runoff)

Rows = Observations

Each row is one record — in our dataset, one day of river data

Since rows are ordered by date, this dataset forms a time series.

CONCEPTS

What Is a Time Series?

A time series is data recorded at regular intervals over time. The order of the observations matters.

Trends

Long-term increase or decrease in values

Seasonal Cycles

Patterns that repeat at the same time each year

Extreme Events

Sudden spikes caused by floods, droughts, or storms

CODE

Convert the Date Column

When pandas loads a CSV, date columns are usually read as plain text. Convert them to proper datetime objects:

```
df["date"] = pd.to_datetime(df["date"])
```

After conversion, pandas understands this column as time data — enabling date-based filtering, resampling, and correct axis labels on plots.

CODE

Import the Plotting Library

Add the matplotlib import to the top of your script:

```
import matplotlib.pyplot as plt
```

`plt` is the standard alias for `matplotlib.pyplot`. You'll use it to create every plot in this session. With pandas and matplotlib ready, you have everything you need to visualize your data.

CODE

Your First Plot

Start with the simplest possible visualization — just two lines of code:

```
plt.plot(df["date"], df["daily_runoff_m3s"])
plt.show()
```

At this stage the goal is minimal: confirm the data loaded correctly and that the time series appears on screen. Don't worry about labels yet — that comes next.

CODE

Improve the Plot

A good plot communicates clearly. Add a title and axis labels:

```
plt.plot(df["date"], df["daily_runoff_m3s"])
plt.title("Mississippi River Daily Runoff")
plt.xlabel("Date")
plt.ylabel("Runoff (m³/s)")
plt.show()
```

01

Clear title

Tells the reader what the chart shows

02

Labeled x-axis

Indicates the time dimension

03

Labeled y-axis

Includes the unit of measurement

Discussion: What Do You See?

Take a moment to look at your plot and think about the following:

Patterns

Do you notice any regular cycles or repeating behavior throughout the year?

Extreme Peaks

Are there sudden spikes? When do they occur — and how dramatic are they?

Variability

Does discharge appear stable or highly variable from day to day?

CODE

Basic Statistics

Use pandas built-in methods to quickly summarize the runoff values:

```
print(df["daily_runoff_m3s"].min())
print(df["daily_runoff_m3s"].max())
print(df["daily_runoff_m3s"].mean())
```

.min()

The lowest single-day discharge in the record

.max()

The highest single-day discharge — likely a flood event

.mean()

The long-term average daily flow

CODE

Monthly Aggregation

Environmental data is often averaged over longer periods to reveal broader patterns. First, set the date column as the DataFrame index, then resample:

```
df = df.set_index("date")

monthly = df.resample("M").mean()
print(monthly.head())
```

`resample("M")` groups all daily values within each calendar month and computes the mean — reducing noise and making seasonal patterns easier to spot.

CODE

Plot Monthly Runoff

Visualize the monthly averages to compare with the daily plot:

```
plt.plot(monthly.index, monthly["daily_runoff_m3s"])
plt.title("Mississippi River Monthly Average Runoff")
plt.xlabel("Date")
plt.ylabel("Runoff (m³/s)")
plt.show()
```

Monthly aggregation produces a smoother curve, filtering out day-to-day noise. Seasonal patterns — like spring peaks — become much easier to observe.

CODE

Monthly Mean Across All Years

To reveal the typical seasonal pattern, calculate the average runoff for each calendar month across the entire dataset:

```
monthly_means = monthly.groupby(
    monthly.index.month
).mean()

print(monthly_means)
```

This compresses the full record into 12 values — one for each month — showing which months are consistently wet or dry on average.

CODE

Visualize the Seasonal Pattern

```
plt.plot(monthly_means.index,
         monthly_means["daily_runoff_m3s"])
plt.title("Average Monthly Runoff by Calendar Month")
plt.xlabel("Month")
plt.ylabel("Runoff (m³/s)")
plt.show()
```

🔺 Peak months

When is average runoff highest? Spring snowmelt? Rainy season?

🔻 Low months

When is flow at its minimum? Late summer drought?

Student Exploration

Try to answer these questions using the dataset. Use the tools we've covered so far.

1

Flow variability

What is the difference between the highest and the lowest runoff?

2

Average flow

What is the mean daily runoff across the entire dataset?

3

Wettest month

Which calendar month has the highest average flow?

CODE

Flow Variability: Answer

To measure variability, we can compare the highest and lowest runoff values in the dataset.

First, find the maximum runoff:

```
df["daily_runoff_m3s"].max()
```

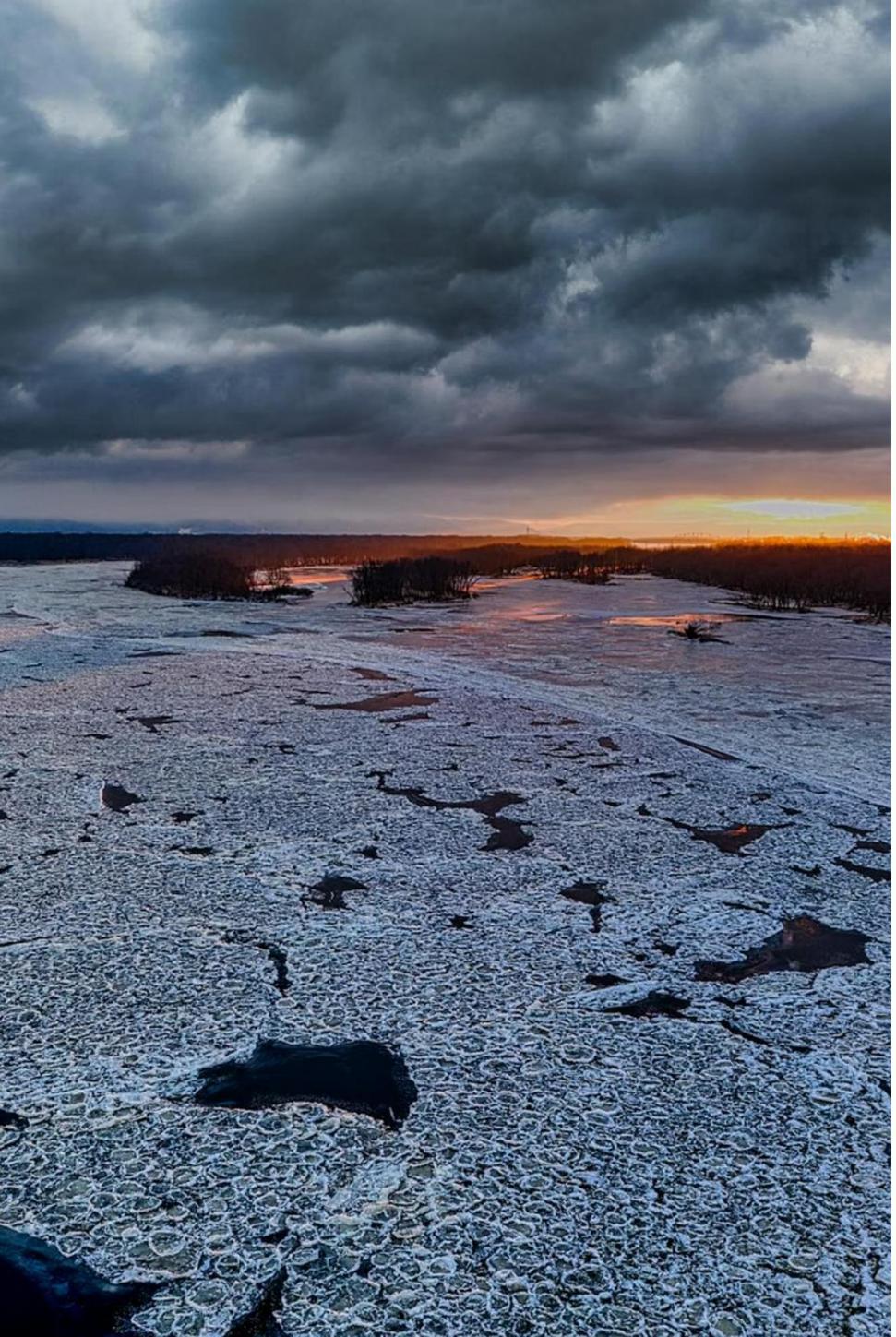
Then find the minimum runoff:

```
df["daily_runoff_m3s"].min()
```

Now calculate the difference:

```
df["daily_runoff_m3s"].max() - df["daily_runoff_m3s"].min()
```

This calculation gives us the range of the dataset, indicating the total span of observed daily flow rates.



INTERPRETATION

Flow Variability: Interpretation

The calculated difference between the highest and lowest runoff values highlights the river's highly dynamic nature. Such significant variability is typically a strong indicator of key hydrological events and seasonal patterns:

Flood Events

Sudden, extreme spikes in discharge caused by intense rainfall, rapid snowmelt, or storm surges, leading to potential flooding.

Seasonal Peaks

Regular, annual increases in flow often linked to predictable climatic patterns, such as spring snowmelt or regional wet seasons.

Low Flow Periods

Extended stretches of significantly reduced water volume, usually during prolonged dry spells or droughts, which can stress ecosystems.

Even this basic statistical summary reveals the complex interplay of factors that shape a river's flow over time.

CODE

Average Flow: Answer

To calculate the average runoff across the entire period, we apply the mean function directly to the daily runoff column:

```
df["daily_runoff_m3s"].mean()
```

`df`

Refers to your loaded pandas DataFrame, which holds all the river data.

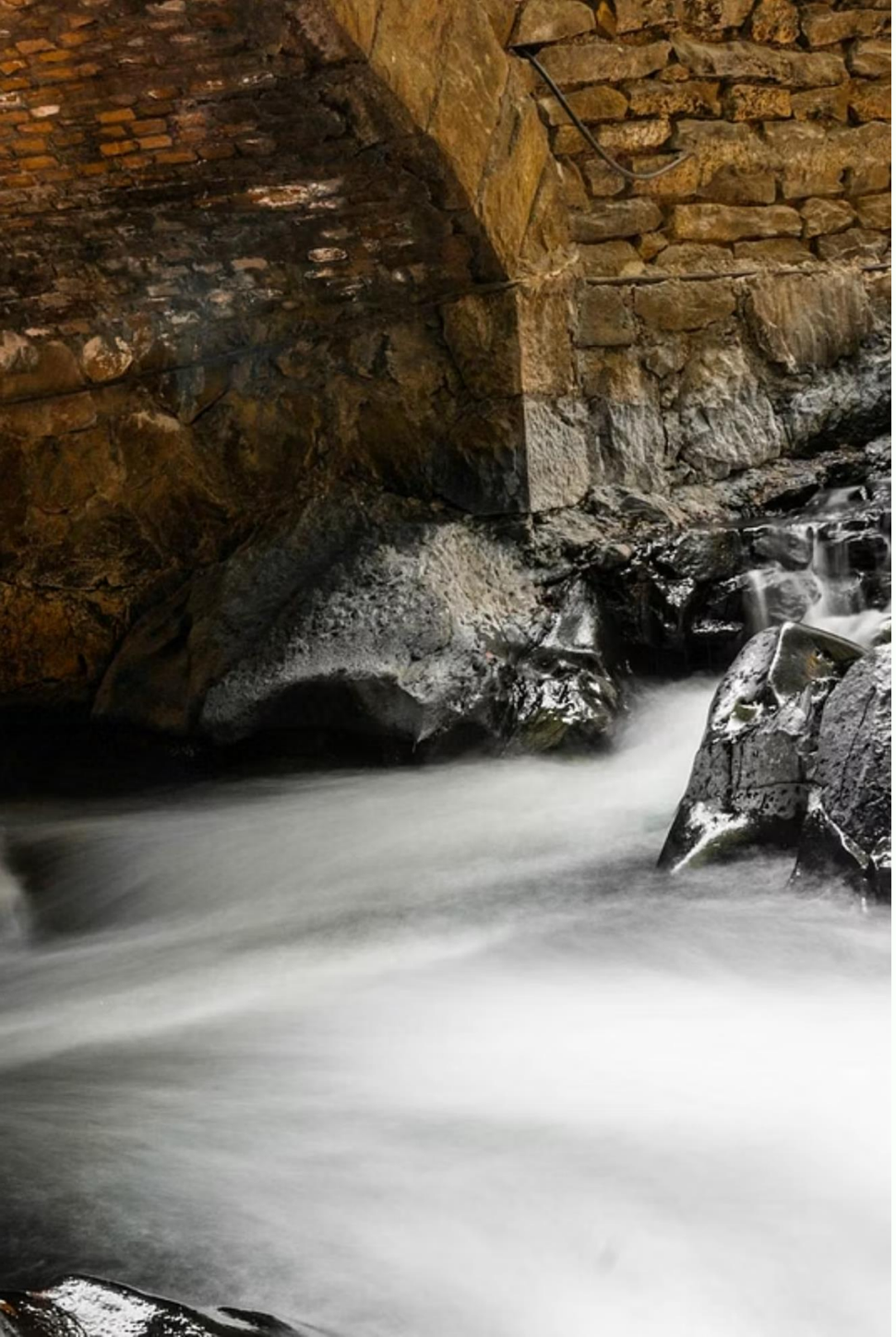
`"daily_runoff_m3s"`

Selects the specific column containing the numerical values for daily runoff.

`.mean()`

A powerful pandas method that computes the arithmetic average of all values in the selected column.

This concise command returns the mean daily runoff across the entire dataset, offering a fundamental measure of the river's long-term average flow rate.



INTERPRETATION

Average Flow: Interpretation

The mean value provides a crucial single number that characterizes the river's typical behavior over the long term.

> The mean gives a typical flow value for the river, establishing a baseline for understanding its hydrological characteristics.

It helps us address fundamental questions about the river's regular state:

- What is the normal discharge of the river?
- How large are typical flows compared to extreme floods or droughts?
- Is the river usually high-flow or low-flow on average?

However, it is vital to remember that the mean alone does not show variability. For a comprehensive understanding, it must always be considered alongside the minimum and maximum values, which we explored earlier.

CODE

Wettest Month: Answer

To determine the wettest month, we'll examine the `monthly_means` DataFrame we previously computed. This table summarizes the average runoff for each calendar month across the entire dataset.

```
print(monthly_means)
```

This output shows a single average runoff value for each month, where `1` represents January and `12` represents December.

To programmatically identify the month with the highest average flow:

```
monthly_means["daily_runoff_m3s"].idxmax()
```

`monthly_means`

Our DataFrame containing the average runoff for each calendar month.

`["daily_runoff_m3s"]`

Selects the specific column holding the average runoff values.

`.idxmax()`

A pandas method that returns the index label (in this case, the month number) of the maximum value in the selected series.

This concise command directly returns the integer corresponding to the month with the highest average runoff, pinpointing the peak period of the river's flow.

CODE

Alternative: Visual Interpretation

Alternatively, we can visually identify the wettest month by inspecting the plot of average monthly runoff:

```
plt.plot(monthly_means.index, monthly_means["daily_runoff_m3s"])
plt.title("Seasonal Pattern of Runoff")
plt.xlabel("Month")
plt.ylabel("Average Runoff (m³/s)")
plt.show()
```

By examining the peak of this graph, we can readily determine which month experiences the highest average runoff.

Observe the Peak

Which month on the x-axis corresponds to the highest point on the runoff curve?

Environmental Interpretation

Numbers alone don't tell the whole story. Think about why these patterns exist.

Rainfall Patterns

Seasonal precipitation cycles drive much of the year-to-year variability in river flow

Snowmelt

Spring warming releases stored winter snowpack, often causing peak discharge in March–May

Seasonal Climate

Temperature, evaporation, and vegetation all modulate how much water reaches the river

WRAP-UP

What We Covered Today

01

Load data

Read a CSV file into a pandas DataFrame

02

Inspect structure

Understand rows, columns, and data types

03

Visualize

Plot a time series with proper labels

04

Statistics

Compute min, max, and mean

05

Aggregate & explore

Resample to monthly data and reveal seasonal patterns

These five steps form the foundation of nearly every real-world data analysis workflow.

Your Feedback Matters

This was our first session — your input directly shapes what comes next.

Was the pace right?

Too fast, too slow, or just about right?

Were the explanations clear?

Any concepts that need more time or a different approach?

What's next for you?

Which topics would you most like to explore in future sessions?

Looking Ahead

Python Corner will evolve with the group. Here's what's on the horizon.

Deeper Time Series

Trend detection, anomaly identification, and rolling statistics

Better Visualizations

Multi-panel figures, custom styles, and interactive plots

Spatial Data

Working with maps and geospatial environmental datasets

Mini Projects

Analyze a dataset of your own choice — your question, your story

Thank you for participating in Session 1. See you next time! 🌿