Python netCDF and xarray

[STARTER KIT]

Marco Miani

TETHYS Summer School (Hands-on Session)

2 September, 2025

How to retrieve API key for Copernicus Data Store (CDS)

Steps to obtain your API key and endpoint URL

1. Go to: https://cds.climate.copernicus.eu/how-to-api
2. Create a user account, if not already done so – validate via e-mail.
✓ Accept Privacy statements.
3. Go to the link above and log in.
4. Now, after logging in, you will see key and url.
5. Copy the `url` and `key` parameters, they look like[1]:

```
url : https://cds.climate.copernicus.eu/api
key : abcd1234-ef56-7890-gh12-ijklmnopqrst
```

5. Have them ready – you'll need them during hands-on session!

[1] This key is made up, only pattern is realistic

Run the Code, Learn the Science!

Everything you need is in the GitHub repo

- **Open the repository:** via GitHub link or QR code
  - ☛ Start with: `1-start-here.ipynb`
- **Launch in Colab:** use the badge on the README
- **Execute:** run each cell in Google Colab
- No setup needed – it runs entirely in the cloud

/geacomputing/UCY2Sept

Prerequisites		You can run the code		You will need	
coding	basic	Cloud	☛ colab	Copernicus Account	free ✓
scientific data	basic	Locally	(git clone) ✓	Google Account	free ✓

Advanced Launchpad – only if you want to run locally (instead of on the cloud)

Advanced users: Set up the full Python environment locally with Conda

- **1. Clone the repository**

  ```
  git clone https://github.com/geacomputing/UCY2Sept.git
  cd UCY2Sept
  ```

- **2. Create a Conda environment from the .yml file**

  ```
  conda env create -f environment.yml
  conda activate your-env-name
  ```

- **3. Launch Jupyter Notebook or Lab**

  ```
  jupyter notebook
  ```

- You're now running the notebooks on your local machine!

Handling Climate Data in Python

NetCDF Processing and Xarray Techniques

Marco Miani

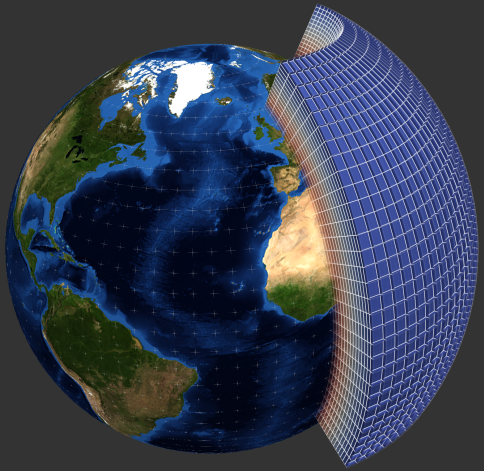
TETHYS Summer School (Hands-on Session)

2 September, 2025 – University of Cyprus in Nicosia

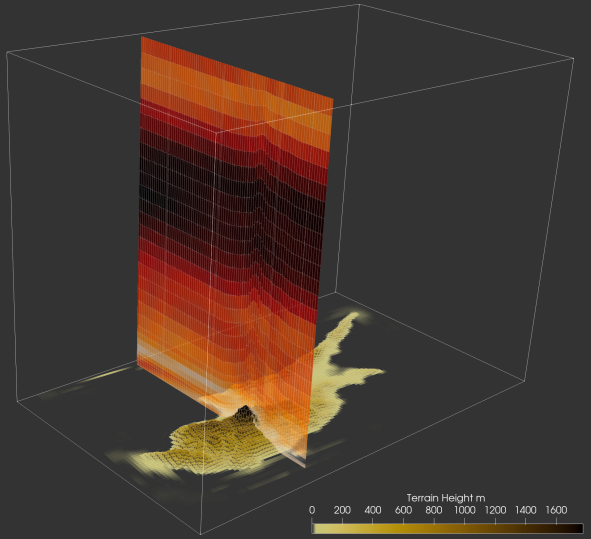
Welcome!

- Name?
- Affiliation?
- Domain of expertise or study?

3D Radial and Angular Grid Discretization of a Spherical Sector
for Numerical Weather Prediction Simulations - global scale



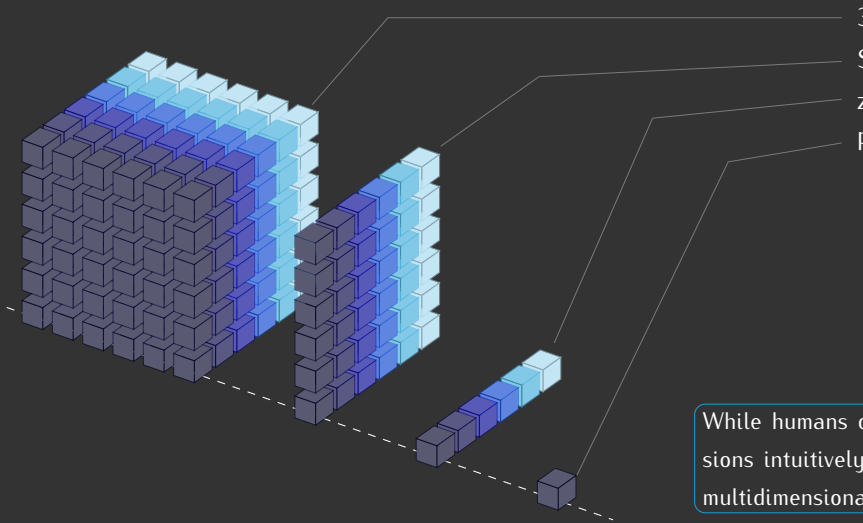
Weather Research Model (WRF), high resolution run for Cyprus (0.02 degrees)
Model gird discretization (Terrain following)



384 variables, 24 timesteps, 31 vertical levels,
85 latitudes, 115 longitudes - daily!

Multidimensional Arrays: Dimensions and Coordinates

A powerful container encapsulating scientific data across multiple dimensions



While humans only grasp 3 spatial dimensions intuitively, (netcdf) arrays encapsulate multidimensional data structures.

netCDF library

Inspect from command line

`ncdump -h <file.nc>`

- Dims, variables, and attributes
- Displays data types and sizes
- Reveals global and variable-specific metadata
- No data
- quick file structure inspection

```
netcdf t2m {
dimensions:
        time = UNLIMITED ; // (365 currently)
        latitude = 721 ;
        longitude = 1440 ;
variables:
        float t2m(time, latitude, longitude) ;
                t2m:long_name = "2 metre temperature" ;
                t2m:units = "K" ;
                t2m:standard_name = "air_temperature" ;
                t2m:missing_value = -9999.f ;
                t2m:_FillValue = -9999.f ;
        double time(time) ;
                time:standard_name = "time" ;
                time:units = "hours since 19000101";
                time:calendar = "gregorian" ;
        float longitude(longitude) ;
                longitude:units = "degrees_east" ;
                longitude:standard_name = "longitude" ;
// global attributes:
                :title = "ERA5 Reanalysis, worldwide" ;
                :institution = "ECMWF" ;
```

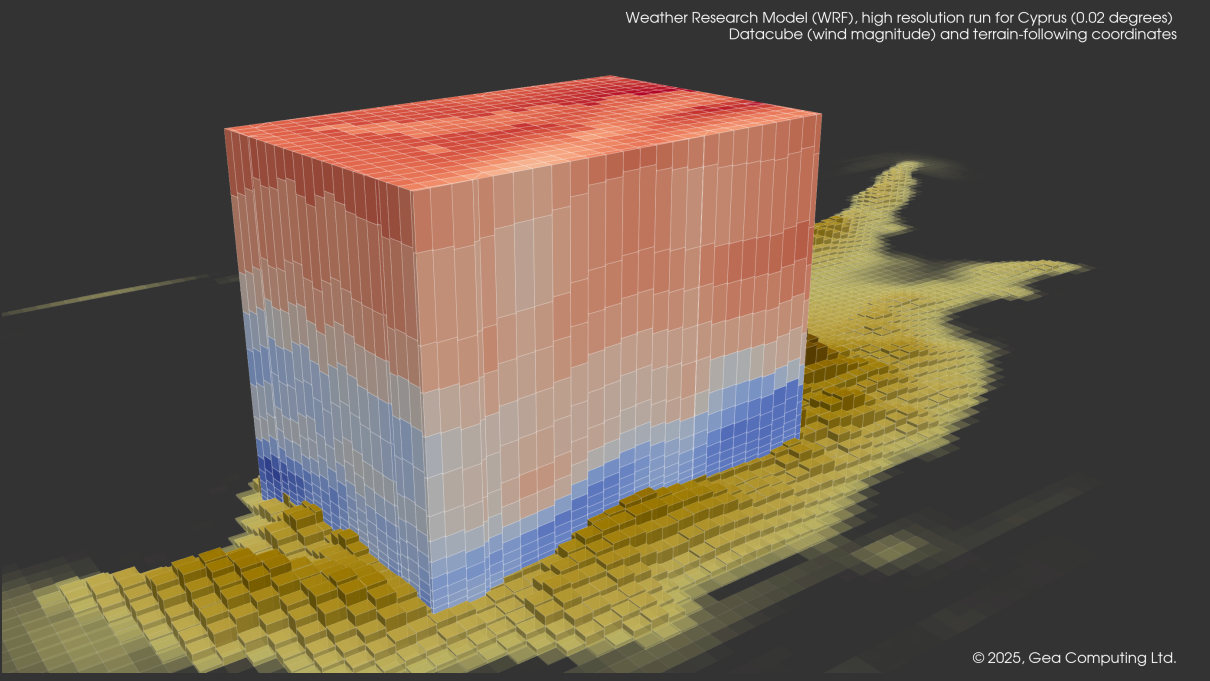
Binary Data Formats in Science

All formats are: open, binary encoded, machine-independent, and widely supported standards

	BUFR	grib	netCDF
Domain	Observations	Gridded Model Data	Gridded Model/General
Data Type	Point	Grid	Grid + Point ✓
Self-describing	No	No	Yes ✓
Common Usage	Meteorological obs.	Weather (short-term)	Climate (long-term)
Avail. Versions	-	grib, grib2	netCDF3, netCDF4
Groups Support	No	Limited	Yes, hierarchical ✓
Compression	lossless[1]	simple lossy[2]	lossless + lossy ✓
Unstructured Grid	-	No	Possible (e.g., WRF) ✓
Binary readers	ecCodes, bufr_lib	ecCodes, wgrib2	ncdump, CDO, NCO

[1] reduces the file size by encoding data more efficiently without losing any information, so the original data can be perfectly reconstructed without errors.

[2] achieves higher compression rates by selectively discarding some data; may introduce small inaccuracies or quality loss – often acceptable depending on the application.


Weather Research Model (WRF), high resolution run for Cyprus (0.02 degrees)
Datacube (wind magnitude) and terrain-following coordinates
© 2025, Gea Computing Ltd.

Why xarray

(and how?)

- Native support for multi-D netCDF data
- Intuitive, labeled coordinates & dimensions
- Easy netCDF read/write — no hassle!
- Metadata & attributes preserved automatically
- Fast, flexible indexing & masking
- Powerful group-by & aggregation tools
- Scales from laptop to cluster with dask
- Excellent documentation
- Designed to evolve

```
ds = xr.tutorial.open_dataset("air_temperature")
```

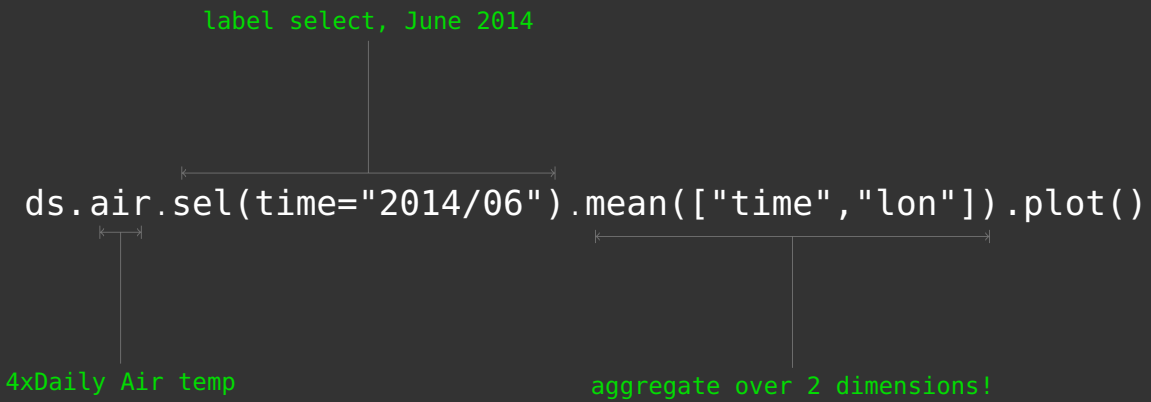
Equivalent of **ncdump**

```
print(ds)         # Use this to print on screen
```

```
<xarray.Dataset>
Dimensions :  (lat:  25, time:  2920, lon:  53)
Coordinates:
   * lat   (lat) float32 75.0 72.5 70.0 67.5 65.0 ...  25.0 22.5 20.0 17.5 15.0
   * lon   (lon) float32 200.0 202.5 205.0 207.5  ...  322.5 325.0 327.5 330.0
   * time (time) datetime64[ns] 2013-01-01  ... 2014-12-31T18:00:00
Data variables:
      air (time, lat, lon) float32 ...
Attributes:
   Conventions :  COARDS
   title       :  4x daily NMC reanalysis (1948)
   description :  4x/day, from 1948 to present.  Arranged on a 2.5-degree grid.
   platform    :  Model
```

```
ds = xr.tutorial.open_dataset("air_temperature")
```

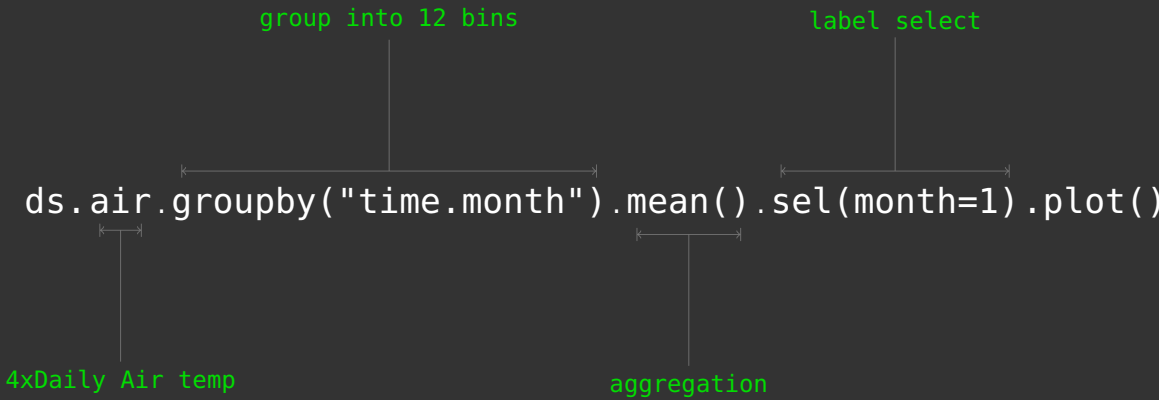
Feature: **aggregation**



- Given input dimensions (`lon, lat, time`), can you guess the result?

```
ds = xr.tutorial.open_dataset("air_temperature")
```

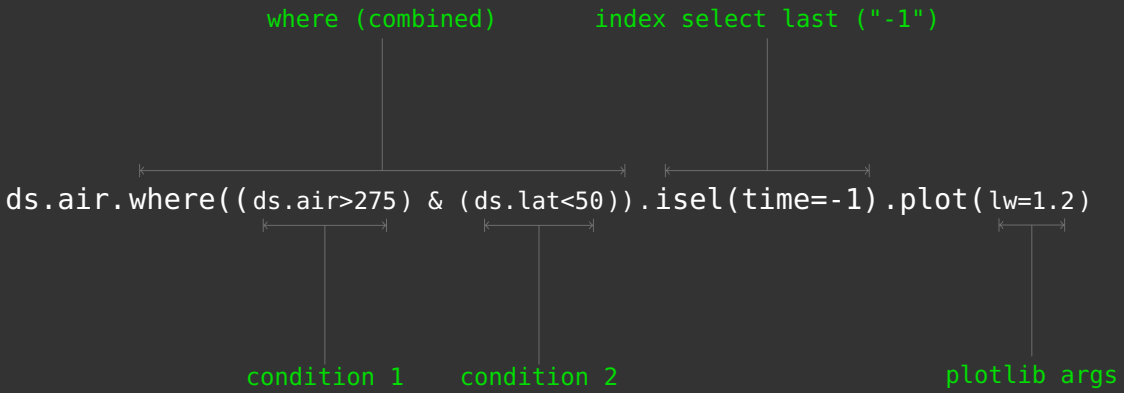
Feature: **grouping**



- Given input dimensions (`lon, lat, time`), can you guess the result?

```
ds = xr.tutorial.open_dataset("air_temperature")
```

Feature **masking**



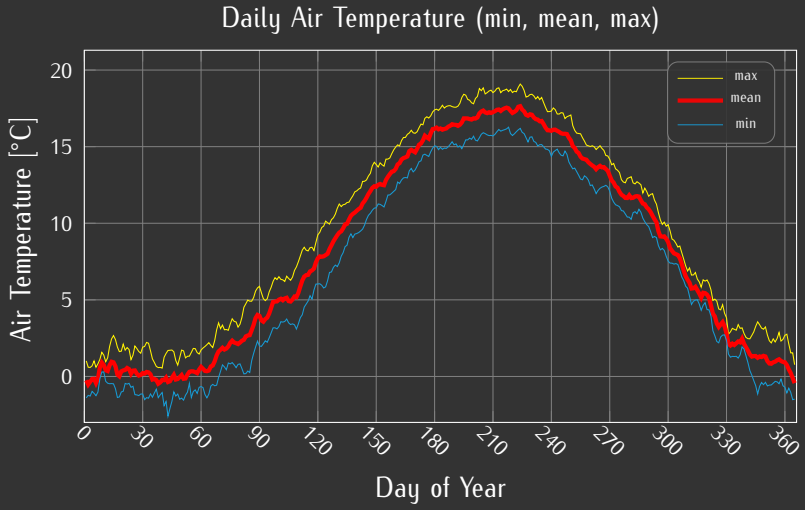
- Given input dimensions (`lon, lat, time`), can you guess the result?

Available datasets: `xarray.tutorial.open_dataset`

	dimensions	description
air_temperature	(lat: 25, time: 2920, lon: 53)	4x daily NMC reanalysis (1948)
air_temperature_gradient	(lat: 25, time: 2920, lon: 53)	Approximate x,y gradients of air temperature
basin_mask	(X: 360, Y: 180, Z: 33)	Ocean basins marked using integers
ASE_ice_velocity	(ny: 800, nx: 500)	Ice Velocity of the Amundsen Sea Embayment, Antarctica
rasm	(time: 36, y: 205, x: 275)	Regional Arctic System Model output (daily averaged Tair)
ROMS_example	(ocean_time: 2, s_rho: 30, eta_rho: 191, xi_rho: 371)	Regional Ocean Model System hindcast run with dyes and oxygen
tiny	(dim_0: 5)	Small synthetic dataset with 1D integer variable
era5-2mt-2019-03-uk.grib	(time: 744, latitude: 33, longitude: 49)	ERA5 2m temperature data over UK, March 2019
eraint_uvz	(longitude: 480, latitude: 241, level: 3, month: 2)	Monthly ERA-Interim upper-level data (u, v, z)
ersstv5	(lat: 89, lon: 180, time: 624, nbnds: 2)	NOAA ERSST.v5 monthly sea surface temperature

```
ds = xr.tutorial.open_dataset("air_temperature")
```

Challenge: compute **daily mean climatology** using xarray



Remarks:

- Select a lat, lon or
- aggregate globally
- Create a dictionary with:
- names, method & styles;
- Iterate using a `for` loop

- Pro challenge:
- use `.rolling()` to filter
- Run this challenge on colab

Run the Code, Learn the Science!

Everything you need is in the GitHub repo

- **Open the repository:** via GitHub link or QR code
  - ☛ Start with: `1-start-here.ipynb`
- **Launch in Colab:** use the badge on the README
- **Execute:** run each cell in Google Colab
- No setup needed – it runs entirely in the cloud

/geacomputing/UCY2Sept

Prerequisites		You can run the code		You will need	
coding	basic	Cloud	☛ colab	Copernicus Account	free ✓
scientific data	basic	Locally	(git clone) ✓	Google Account	free ✓

References I

[3] xarray developers. *Indexing and selecting (xarray)*. https://docs.xarray.dev/en/latest/user-guide/indexing.html. Accessed: 2025-08-08. 2025.

[4] xarray developers. *toy weather data (xarray)*. https://docs.xarray.dev/en/latest/examples/weather-data.html. Accessed: 2025-08-08. 2025.

[5] Copernicus Climate Data Store. *API set up, and keys*. https://cds.climate.copernicus.eu/how-to-api. Accessed: 2025-08-08. 2025.

[6] Copernicus Climate Data Store. *Copernicus learning resources*. https://cds.climate.copernicus.eu/training. Accessed: 2025-08-08. 2025.

[7] ECMWF. *ECMWF E-learning modules*. https://learning.ecmwf.int/. Accessed: 2025-08-08. 2025.

References II

[8] ECMWF. *jupyter notebooks from ECMWF*. https://ecmwf-projects.github.io/copernicus-training-c3s/intro.html. Accessed: 2025-08-08. 2025.

[9] Copernicus Atmosphere Monitoring Service. *Atmospheric Data Store*. https://ads.atmosphere.copernicus.eu/. Accessed: 2025-08-08. 2025.

[10] Copernicus Marine Service. *Copernicus marine, Ocean products, download netcdf, satellite, model, indicators and measurements*. https://data.marine.copernicus.eu/products. Accessed: 2025-08-08. 2025.

[11] Copernicus Marine Service. *Ocean visualization tool, from Copernicus Marine*. https://marine.copernicus.eu/access-data/ocean-visualisation-tools. Accessed: 2025-08-08. 2025.

References III

[12] Copernicus Marine Service. *Copernicus Marine Toolbox - API access*. https://help.marine.copernicus.eu/en/articles/7949409-copernicus-marine-toolbox-introduction. Accessed: 2025-08-08. 2025.

[13] CF Conventions Committee. *Climate and Forecast CF Convention for netcdf data*. https://cfconventions.org/Data/cf-conventions/cf-conventions-1.12/cf-conventions.html. Accessed: 2025-08-08. 2025.

[14] xarray developers. *coordinates in xarray*. https://docs.xarray.dev/en/latest/generated/xarray.Coordinates.html. Accessed: 2025-08-08. 2025.

Supporting Slides

SST Anomaly Computation: CDO vs. xarray Performance

Hourly Southern Mediterranean SST dataset (5 years, 446×151 grid)

- **Input data [1.7GB] – SST for southern Med**

time	=	43,830 (≈ 5 years of **hourly** data)
longitude	=	446
latitude	=	151

- **Task:** Compute SST anomalies – Steps:

`cdo daymean`	Daily mean SST from hourly data
`cdo ydaymean`	Daily climatology (mean for each calendar day)
`cdo ydaysub`	Daily anomalies = d. mean – d. climatology

- **Results:**

CDO	**16 seconds** ✓
xarray/Python	**5 minutes** ✘

SST Anomaly Computation: CDO vs. xarray Performance

Commands vs.output (with timing)

```
echo "=== Starting SST anomaly computation ==="
start_total=$(date +%s)

echo "Step 1: Computing daily mean..."
start=$(date +%s)
cdo daymean soouthern_med_cdo.nc sst_daymean.nc
end=$(date +%s)
echo "Step 1 completed in $((end-start)) seconds."

echo "Step 2: Computing daily climatology..."
start=$(date +%s)
cdo ydaymean sst_daymean.nc sst_daily_clim.nc
end=$(date +%s)
echo "Step 2 completed in $((end-start)) seconds."

echo "Step 3: Computing daily anomalies..."
start=$(date +%s)
cdo ydaysub sst_daymean.nc sst_daily_clim.nc
↪  sst_daily_anom.nc
end=$(date +%s)
echo "Step 3 completed in $((end-start)) seconds."

end_total=$(date +%s)
echo "=== All steps completed in
↪  $((end_total-start_total)) seconds ==="
```

```
#=== Starting SST anomaly computation ===
# Step 1: Computing daily mean...
# Processed 2,951,775,180 values over 43,830 timesteps [14.19s]
# Step 1 completed in 14 seconds.

# Step 2: Computing daily climatology...
# Processed 123,041,142 values over 1,827 timesteps [0.95s]
# Step 2 completed in 1 second.

# Step 3: Computing daily anomalies...
# Processed 147,689,778 values over 2,193 timesteps [1.13s]
# Step 3 completed in 1 second.

#=== All steps completed in 16 seconds ===
```

Time-series: use xarray to store station data and convert to netCDF

(Add metadata and compress resulting netCDF file)

1. Generate data

```python
import numpy as np
import pandas as pd

# Time series
tstart = "2025-01-01"
t = pd.date_range(tstart, periods=10)
tmp = 15 + 5* np.random.rand(len(t))
prec = np.random.rand(len(t))

# Station metadata
station_id = "ST001"
station_name = "Central Park"
latitude = 40.785091
longitude = -73.968285
elevation = 15.0  # meters
```

2. xarray → NetCDF

```python
import xarray as xr

ds = xr.Dataset(
    {"tmp": ("time", tmp),
     "prec": ("time", prec)},
    coords={
        "time": t,
        "sid": ("station", ["ST001"]),
        "sname": ("station", ["CP"]),
        "lat": ("station", [40.785]),
        "lon": ("station", [-73.968]),
        "ele": ("station", [15])
    },
    attrs={"desc": "Daily weather data"}
)

ds.to_netcdf(
    f"Station_{station_id}.nc",
    unlimited_dims='time',
    encoding={var: {"zlib": True, "complevel": 4} for var in ds.data_vars}
    )
```

3. Inspect NetCDF

```
$ ncdump -h data.nc

netcdf data {
dimensions:
    time = UNLIMITED ; // (10 currently)
variables:
    double tmp(time) ;
    double prec(time) ;
    double time(time) ;
}
```

High-Resolution Ocean Insights with ParaView

Animating Copernicus Marine SST Data Using Open-Source Scientific Visualization Software

- **Satellite Data (L4):**
  Ultra High Resolution SST
- **ParaView:**
  open-source 3D visualization
- **Workflow:**
  from NetCDF to animation
- **Animated Output:**
  Publication-ready