Download Copernicus data - read, manipulate and visualize netcdf files

Climate datasets are available on the CDS store at https://cds.climate.copernicus.eu/datasets To access and download climate data from the Copernicus data store a user must

1. Create an ECMWF account - click on the Log-in / register button (top right icon on the CDS website)
2. Install the cds api and create a .cdsapirc file in their $HOME directory (usually /home/username/.cdsapirc on Linux/MacOS or %USERPROFILE%.cdsapirc e.g. C:\Users\Username folder for Windows users). The .cdsapirc file contains two lines, the URL of the CDS and a crypted key to access the data. Detailed instructions and examples are available at https://cds.climate.copernicus.eu/how-to-api
3. Send a request using the CDS api in Python to retrieve netcdf (or grib) files

This example focuses on Jupyter notebook/Python there are packages in R to use the same API - details are available at https://bluegreen-labs.github.io/ecmwfr/

In [ ]:
```
################################################################################
# Install required packages automatically if not already installed on the system
################################################################################
import sys
!{sys.executable} -m pip install numpy
!{sys.executable} -m pip install matplotlib
!{sys.executable} -m pip install pandas
!{sys.executable} -m pip install cdsapi
!{sys.executable} -m pip install netCDF4
!{sys.executable} -m pip install cartopy
!{sys.executable} -m pip install xarray
!{sys.executable} -m pip install datetime
```

The following script is required to operate the CDS API

Please paste your URL and KEY below to acquire access to the Climate Data Store.

To open an account or obtain a key, please follow instructions at Climate Data Store

*Please note that the following will **replace** any existing key if you run it outside this Container!*

In [ ]:
```
import os

url = "https://cds.climate.copernicus.eu/api"
key = "<PERSONAL-ACCESS-TOKEN>"

os.system("echo 'url: %s' > ~/.cdsapirc" %url)
os.system("echo 'key: %s' >> ~/.cdsapirc" %key)
```

In [ ]:
```
##############################
# Import required packages
##############################
import cdsapi
import netCDF4 as nc
import numpy as np
import pandas as pd
import math
import matplotlib.pyplot as plt
```

```
import cartopy.crs as ccrs
from cartopy.util import add_cyclic_point
import cartopy.mpl.ticker as cticker
import xarray as xr
import datetime
from calendar import monthrange
from datetime import datetime
from datetime import timedelta
```

1) Use website to copy API request directly into Jupyter notebook

Climate datasets are available on the CDS store at https://cds.climate.copernicus.eu/datasets User guide and information about the CDS are available at https://cds.climate.copernicus.eu/user-guide

For the first example, we will use the CDS website to copy and paste the request directly

1. In the search bar type "ERA5 daily"
2. Select "ERA5 post-processed daily statistics on single levels from 1940 to present" [https://cds.climate.copernicus.eu/datasets/derived-era5-single-levels-daily-statistics?tab=overview]
3. 3 Tabs are available: (i) "Overview" provides details about the dataset (ii) "Download" allows to select variables and download the data and (iii) "Documentation" provides links to related scientifc publications and technical reports
4. Click on the Download Tab
5. Tick the boxes "2m temperature", "2025", ", July and August", "Day -> Select all", "Frequency 1-hourly"
6. In "Terms of use" the user needs to accept the lience agreement (to do once for a particular dataset)
7. The data can then be downloaded directly using a browser by pressing the "Submit Form" button, but we will use the API in the following example
8. Under API request click on "Show API request code". This piece of code needs to be copied in the box below. Note that we already imported the cdsapi package earlier ("import cdsapi") and we mofied the output file name and directory (last line of the code below)

In [3]:
```
# Define output directory and file name
outdir = "Data"
outfile = "example1_ERA5.nc" # Name of the output netcdf file

# Create output directory if it does not exist
if not os.path.exists(outdir):
    os.mkdir(outdir)
outfile = outdir + "/" + outfile

################################################################################
# Copy and paste request from website below (1st example is provided below)
################################################################################

dataset = "derived-era5-single-levels-daily-statistics"
request = {
    "product_type": "reanalysis",
    "variable": ["2m_temperature"],
    "year": "2025",
    "month": ["07", "08"],
    "day": [
        "01", "02", "03",
        "04", "05", "06",
```

```
        "07", "08", "09",
        "10", "11", "12",
        "13", "14", "15",
        "16", "17", "18",
        "19", "20", "21",
        "22", "23", "24",
        "25", "26", "27",
        "28", "29", "30",
        "31"
    ],
    "daily_statistic": "daily_mean",
    "time_zone": "utc+00:00",
    "frequency": "1_hourly"
}

client = cdsapi.Client()
client.retrieve(dataset, request).download(outfile) # we added an output file name th
```

```
2025-09-13 15:19:18,959 INFO [2024-09-26T00:00:00] Watch our [Forum](https://forum.ecm
wf.int/) for Announcements, news and other discussed topics.
2025-09-13 15:19:19,507 INFO Request ID is 22d8e741-055f-4ffb-98b7-b29663d292c8
2025-09-13 15:19:19,708 INFO status has been updated to accepted
2025-09-13 15:19:33,521 INFO status has been updated to running
2025-09-13 15:19:53,224 INFO status has been updated to successful
38bfcbcec513a353d3116f020d809548.nc:   0%|          | 0.00/141M [00:00<?, ?B/s]
```

Out[3]: 'Data/example1_ERA5.nc'

Read and manipulate the example file

We have now downloaded global daily temperature data based on the ERA5 dataset for July-August 2025. The output file (Data/example1_ERA5.nc) is in netcdf format (*.nc) and contains gridded temperature data and the associated metadata. We will first read the netcdf file directly into Python and print some information about the variables, dimensions and attributes

In [4]:
```
##############################################################################
# Read climate data file (previously defined as outfile) and print basic information
##############################################################################

ds = nc.Dataset(outfile)

# Print some information about the netcdf file
print(ds)

# The former command probides information about the file format - variables - their d
# Temperature is a 3D variable t2m(valid_time, latitude, longitude)
# Time has 62 points valid_time(62) for July-August 2025, and the related latitude(72

# Metadata can also be accessed using a catalogue/dictionnary
print(ds.__dict__)

# More information about the temperature variable
print(ds['t2m']) # Temperature is in Kelvin hand the 3D temperature array (ntime=62, l
print(ds['latitude']) # lat in degrees
print(ds['longitude']) # Lon in degrees
print(ds['valid_time']) # Time

# We can also print the latitude and longitude numerical values below
# standard resolution from our former API request is 0.25deg for both lat and Lon
print(ds['latitude'][:])
print(ds['longitude'][:])
```

```
<class 'netCDF4.Dataset'>
root group (NETCDF4 data model, file format HDF5):
    GRIB_centre: ecmf
    GRIB_centreDescription: European Centre for Medium-Range Weather Forecasts
    GRIB_subCentre: 0
    Conventions: CF-1.7
    institution: European Centre for Medium-Range Weather Forecasts
    history: 2025-09-10T12:53 GRIB to CDM+CF via cfgrib-0.9.15.0/ecCodes-2.42.0 with
{"source": "2m_temperature.grib", "filter_by_keys": {"stream": ["oper"]}, "encode_cf":
["parameter", "time", "geography", "vertical"]}
earthkit.transforms.aggregate.temporal.daily_reduce(2m_temperature_stream-oper, how=me
an, **{'time_shift': {'hours': 0}})
    dimensions(sizes): valid_time(62), latitude(721), longitude(1440)
    variables(dimensions): float32 t2m(valid_time, latitude, longitude), int64 number
(), float64 latitude(latitude), float64 longitude(longitude), int64 valid_time(valid_t
ime)
    groups:
{'GRIB_centre': 'ecmf', 'GRIB_centreDescription': 'European Centre for Medium-Range We
ather Forecasts', 'GRIB_subCentre': np.int64(0), 'Conventions': 'CF-1.7', 'institutio
n': 'European Centre for Medium-Range Weather Forecasts', 'history': '2025-09-10T12:53
GRIB to CDM+CF via cfgrib-0.9.15.0/ecCodes-2.42.0 with {"source": "2m_temperature.gri
b", "filter_by_keys": {"stream": ["oper"]}, "encode_cf": ["parameter", "time", "geogra
phy", "vertical"]}\nearthkit.transforms.aggregate.temporal.daily_reduce(2m_temperature
_stream-oper, how=mean, **{\'time_shift\': {\'hours\': 0}})'}
<class 'netCDF4.Variable'>
float32 t2m(valid_time, latitude, longitude)
    _FillValue: nan
    GRIB_paramId: 167
    GRIB_dataType: an
    GRIB_numberOfPoints: 1038240
    GRIB_typeOfLevel: surface
    GRIB_stepUnits: 1
    GRIB_stepType: instant
    GRIB_gridType: regular_ll
    GRIB_uvRelativeToGrid: 0
    GRIB_NV: 0
    GRIB_Nx: 1440
    GRIB_Ny: 721
    GRIB_cfName: unknown
    GRIB_cfVarName: t2m
    GRIB_gridDefinitionDescription: Latitude/Longitude Grid
    GRIB_iDirectionIncrementInDegrees: 0.25
    GRIB_iScansNegatively: 0
    GRIB_jDirectionIncrementInDegrees: 0.25
    GRIB_jPointsAreConsecutive: 0
    GRIB_jScansPositively: 0
    GRIB_latitudeOfFirstGridPointInDegrees: 90.0
    GRIB_latitudeOfLastGridPointInDegrees: -90.0
    GRIB_longitudeOfFirstGridPointInDegrees: 0.0
    GRIB_longitudeOfLastGridPointInDegrees: 359.75
    GRIB_missingValue: 3.4028234663852886e+38
    GRIB_name: 2 metre temperature
    GRIB_shortName: 2t
    GRIB_totalNumber: 0
    GRIB_units: K
    long_name: 2 metre temperature
    units: K
    standard_name: unknown
    GRIB_surface: 0.0
    coordinates: number
unlimited dimensions:
current shape = (62, 721, 1440)
filling on
<class 'netCDF4.Variable'>
float64 latitude(latitude)
    _FillValue: nan
```

```
    units: degrees_north
    standard_name: latitude
    long_name: latitude
    stored_direction: decreasing
unlimited dimensions:
current shape = (721,)
filling on
<class 'netCDF4.Variable'>
float64 longitude(longitude)
    _FillValue: nan
    units: degrees_east
    standard_name: longitude
    long_name: longitude
unlimited dimensions:
current shape = (1440,)
filling on
<class 'netCDF4.Variable'>
int64 valid_time(valid_time)
    time_shift: 0 days 00:00:00
    units: days since 2025-07-01 00:00:00
    calendar: proleptic_gregorian
unlimited dimensions:
current shape = (62,)
filling on, default _FillValue of -9223372036854775806 used
[ 90.    89.75  89.5   89.25  89.    88.75  88.5   88.25  88.    87.75
  87.5   87.25  87.    86.75  86.5   86.25  86.    85.75  85.5   85.25
  85.    84.75  84.5   84.25  84.    83.75  83.5   83.25  83.    82.75
  82.5   82.25  82.    81.75  81.5   81.25  81.    80.75  80.5   80.25
  80.    79.75  79.5   79.25  79.    78.75  78.5   78.25  78.    77.75
  77.5   77.25  77.    76.75  76.5   76.25  76.    75.75  75.5   75.25
  75.    74.75  74.5   74.25  74.    73.75  73.5   73.25  73.    72.75
  72.5   72.25  72.    71.75  71.5   71.25  71.    70.75  70.5   70.25
  70.    69.75  69.5   69.25  69.    68.75  68.5   68.25  68.    67.75
  67.5   67.25  67.    66.75  66.5   66.25  66.    65.75  65.5   65.25
  65.    64.75  64.5   64.25  64.    63.75  63.5   63.25  63.    62.75
  62.5   62.25  62.    61.75  61.5   61.25  61.    60.75  60.5   60.25
  60.    59.75  59.5   59.25  59.    58.75  58.5   58.25  58.    57.75
  57.5   57.25  57.    56.75  56.5   56.25  56.    55.75  55.5   55.25
  55.    54.75  54.5   54.25  54.    53.75  53.5   53.25  53.    52.75
  52.5   52.25  52.    51.75  51.5   51.25  51.    50.75  50.5   50.25
  50.    49.75  49.5   49.25  49.    48.75  48.5   48.25  48.    47.75
  47.5   47.25  47.    46.75  46.5   46.25  46.    45.75  45.5   45.25
  45.    44.75  44.5   44.25  44.    43.75  43.5   43.25  43.    42.75
  42.5   42.25  42.    41.75  41.5   41.25  41.    40.75  40.5   40.25
  40.    39.75  39.5   39.25  39.    38.75  38.5   38.25  38.    37.75
  37.5   37.25  37.    36.75  36.5   36.25  36.    35.75  35.5   35.25
  35.    34.75  34.5   34.25  34.    33.75  33.5   33.25  33.    32.75
  32.5   32.25  32.    31.75  31.5   31.25  31.    30.75  30.5   30.25
  30.    29.75  29.5   29.25  29.    28.75  28.5   28.25  28.    27.75
  27.5   27.25  27.    26.75  26.5   26.25  26.    25.75  25.5   25.25
  25.    24.75  24.5   24.25  24.    23.75  23.5   23.25  23.    22.75
  22.5   22.25  22.    21.75  21.5   21.25  21.    20.75  20.5   20.25
  20.    19.75  19.5   19.25  19.    18.75  18.5   18.25  18.    17.75
  17.5   17.25  17.    16.75  16.5   16.25  16.    15.75  15.5   15.25
  15.    14.75  14.5   14.25  14.    13.75  13.5   13.25  13.    12.75
  12.5   12.25  12.    11.75  11.5   11.25  11.    10.75  10.5   10.25
  10.     9.75   9.5    9.25   9.     8.75   8.5    8.25   8.     7.75
   7.5    7.25   7.     6.75   6.5    6.25   6.     5.75   5.5    5.25
   5.     4.75   4.5    4.25   4.     3.75   3.5    3.25   3.     2.75
   2.5    2.25   2.     1.75   1.5    1.25   1.     0.75   0.5    0.25
   0.    -0.25  -0.5   -0.75  -1.    -1.25  -1.5   -1.75  -2.    -2.25
  -2.5   -2.75  -3.    -3.25  -3.5   -3.75  -4.    -4.25  -4.5   -4.75
  -5.    -5.25  -5.5   -5.75  -6.    -6.25  -6.5   -6.75  -7.    -7.25
  -7.5   -7.75  -8.    -8.25  -8.5   -8.75  -9.    -9.25  -9.5   -9.75
 -10.   -10.25 -10.5  -10.75 -11.   -11.25 -11.5  -11.75 -12.   -12.25
 -12.5  -12.75 -13.   -13.25 -13.5  -13.75 -14.   -14.25 -14.5  -14.75
```

```
-15.   -15.25 -15.5  -15.75 -16.   -16.25 -16.5  -16.75 -17.   -17.25
-17.5  -17.75 -18.   -18.25 -18.5  -18.75 -19.   -19.25 -19.5  -19.75
-20.   -20.25 -20.5  -20.75 -21.   -21.25 -21.5  -21.75 -22.   -22.25
-22.5  -22.75 -23.   -23.25 -23.5  -23.75 -24.   -24.25 -24.5  -24.75
-25.   -25.25 -25.5  -25.75 -26.   -26.25 -26.5  -26.75 -27.   -27.25
-27.5  -27.75 -28.   -28.25 -28.5  -28.75 -29.   -29.25 -29.5  -29.75
-30.   -30.25 -30.5  -30.75 -31.   -31.25 -31.5  -31.75 -32.   -32.25
-32.5  -32.75 -33.   -33.25 -33.5  -33.75 -34.   -34.25 -34.5  -34.75
-35.   -35.25 -35.5  -35.75 -36.   -36.25 -36.5  -36.75 -37.   -37.25
-37.5  -37.75 -38.   -38.25 -38.5  -38.75 -39.   -39.25 -39.5  -39.75
-40.   -40.25 -40.5  -40.75 -41.   -41.25 -41.5  -41.75 -42.   -42.25
-42.5  -42.75 -43.   -43.25 -43.5  -43.75 -44.   -44.25 -44.5  -44.75
-45.   -45.25 -45.5  -45.75 -46.   -46.25 -46.5  -46.75 -47.   -47.25
-47.5  -47.75 -48.   -48.25 -48.5  -48.75 -49.   -49.25 -49.5  -49.75
-50.   -50.25 -50.5  -50.75 -51.   -51.25 -51.5  -51.75 -52.   -52.25
-52.5  -52.75 -53.   -53.25 -53.5  -53.75 -54.   -54.25 -54.5  -54.75
-55.   -55.25 -55.5  -55.75 -56.   -56.25 -56.5  -56.75 -57.   -57.25
-57.5  -57.75 -58.   -58.25 -58.5  -58.75 -59.   -59.25 -59.5  -59.75
-60.   -60.25 -60.5  -60.75 -61.   -61.25 -61.5  -61.75 -62.   -62.25
-62.5  -62.75 -63.   -63.25 -63.5  -63.75 -64.   -64.25 -64.5  -64.75
-65.   -65.25 -65.5  -65.75 -66.   -66.25 -66.5  -66.75 -67.   -67.25
-67.5  -67.75 -68.   -68.25 -68.5  -68.75 -69.   -69.25 -69.5  -69.75
-70.   -70.25 -70.5  -70.75 -71.   -71.25 -71.5  -71.75 -72.   -72.25
-72.5  -72.75 -73.   -73.25 -73.5  -73.75 -74.   -74.25 -74.5  -74.75
-75.   -75.25 -75.5  -75.75 -76.   -76.25 -76.5  -76.75 -77.   -77.25
-77.5  -77.75 -78.   -78.25 -78.5  -78.75 -79.   -79.25 -79.5  -79.75
-80.   -80.25 -80.5  -80.75 -81.   -81.25 -81.5  -81.75 -82.   -82.25
-82.5  -82.75 -83.   -83.25 -83.5  -83.75 -84.   -84.25 -84.5  -84.75
-85.   -85.25 -85.5  -85.75 -86.   -86.25 -86.5  -86.75 -87.   -87.25
-87.5  -87.75 -88.   -88.25 -88.5  -88.75 -89.   -89.25 -89.5  -89.75
-90.  ]
[0.0000e+00 2.5000e-01 5.0000e-01 ... 3.5925e+02 3.5950e+02 3.5975e+02]
```

Plot a map to visualize the data

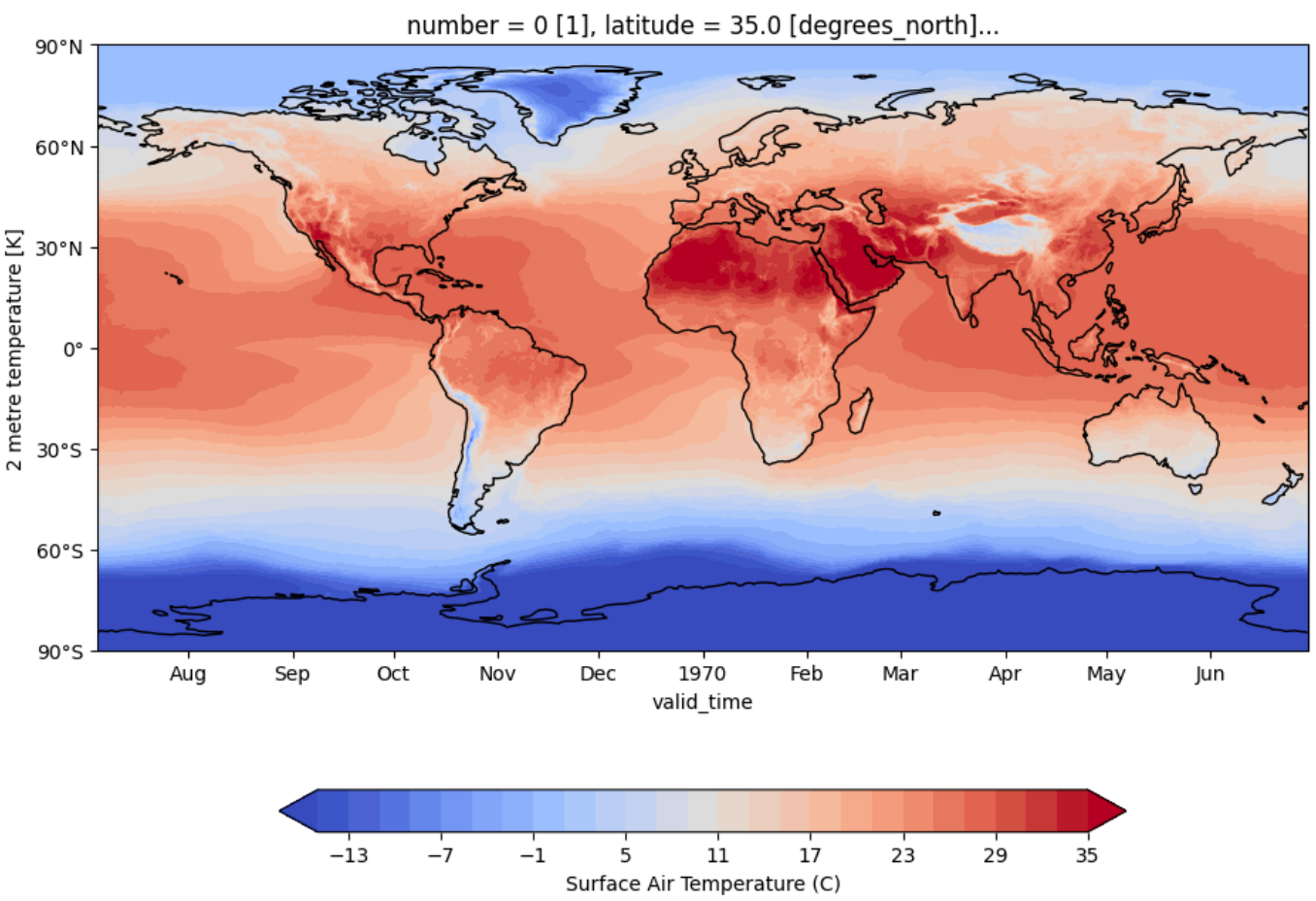
In [5]:

```
###############################################################################
# Plot a map - averaging over the time dimension
###############################################################################

# xarray can also be used to read nc files and print file info
dset = xr.open_dataset(outfile)
print(dset)

# Calculate average on the time dimension
ds_mean=dset.mean(dim='valid_time')
ds_mean = ds_mean -273.15

# Make the figure larger
fig = plt.figure(figsize=(11,8.5))

# Set the axes using the specified map projection
ax=plt.axes(projection=ccrs.PlateCarree())

levels = np.linspace(-15, 35, 26)
# Make a filled contour plot
cs=ax.contourf(dset['longitude'], dset['latitude'], ds_mean['t2m'],
            transform = ccrs.PlateCarree(),cmap='coolwarm',extend='both',levels = lev

# Add coastlines
ax.coastlines()

# Define the xticks for longitude
ax.set_xticks(np.arange(-180,181,60), crs=ccrs.PlateCarree())
lon_formatter = cticker.LongitudeFormatter()
```

```
ax.xaxis.set_major_formatter(lon_formatter)

# Define the yticks for latitude
ax.set_yticks(np.arange(-90,91,30), crs=ccrs.PlateCarree())
lat_formatter = cticker.LatitudeFormatter()
ax.yaxis.set_major_formatter(lat_formatter)

# Add colorbar
cbar = plt.colorbar(cs,shrink=0.7,orientation='horizontal',label='Surface Air Tempera

###############################################################################
# Plot time series for a selected location (Larnaca)
###############################################################################

latsel = 34.9
lonsel = 33.6  # Larnaca

# Extract a dataset closest to specified point
dsloc = dset.sel(longitude=lonsel, latitude=latsel, method='nearest')

# select a variable to plot
dsloc['t2m'].plot()
```

```
<xarray.Dataset> Size: 258MB
Dimensions:     (valid_time: 62, latitude: 721, longitude: 1440)
Coordinates:
    number      int64 8B ...
  * latitude    (latitude) float64 6kB 90.0 89.75 89.5 ... -89.5 -89.75 -90.0
  * longitude   (longitude) float64 12kB 0.0 0.25 0.5 0.75 ... 359.2 359.5 359.8
  * valid_time  (valid_time) datetime64[ns] 496B 2025-07-01 ... 2025-08-31
Data variables:
    t2m         (valid_time, latitude, longitude) float32 257MB ...
Attributes:
    GRIB_centre:             ecmf
    GRIB_centreDescription:  European Centre for Medium-Range Weather Forecasts
    GRIB_subCentre:          0
    Conventions:             CF-1.7
    institution:             European Centre for Medium-Range Weather Forecasts
    history:                 2025-09-10T12:53 GRIB to CDM+CF via cfgrib-0.9.1...
```

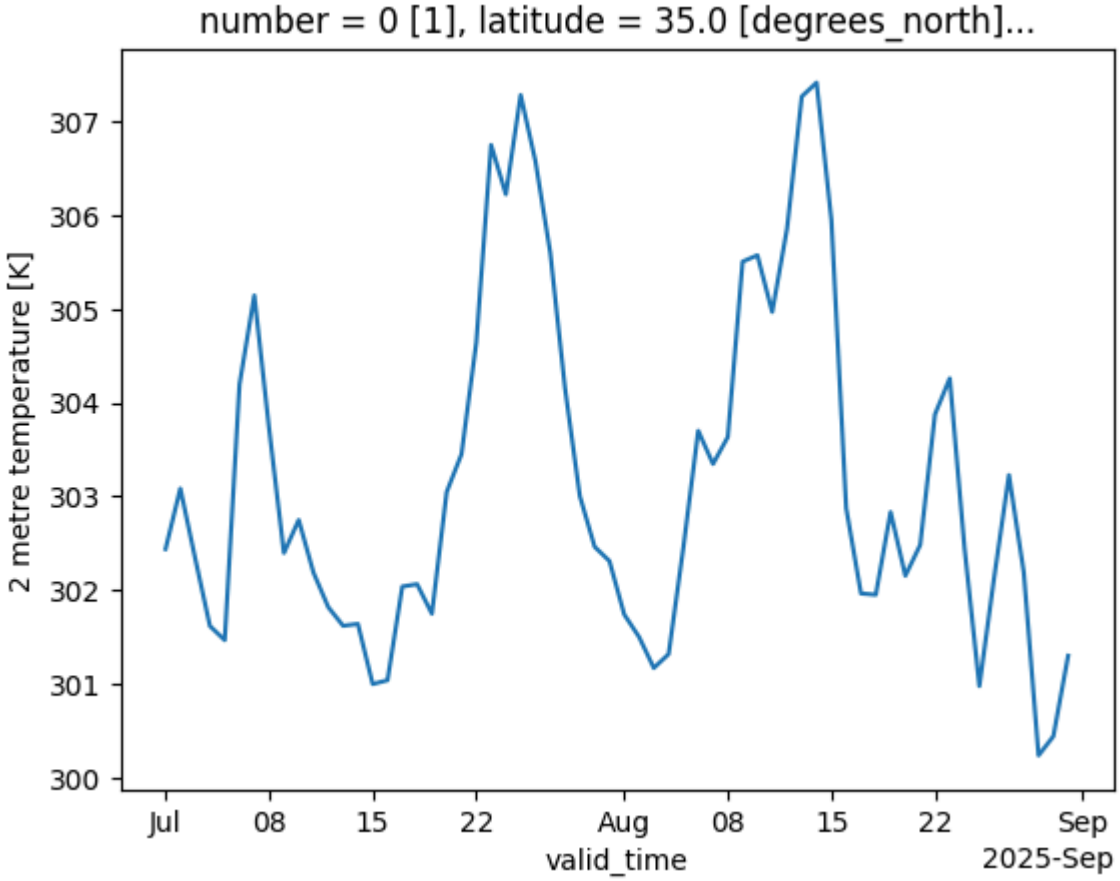
Out[5]: [<matplotlib.lines.Line2D at 0x786150a6e2a0>]



Plot a time series for a given location

In [6]:
```
###############################################################################
# Plot time series for a selected location
###############################################################################

latsel = 34.9
lonsel = 33.6  # Larnaca

# Extract a dataset closest to specified point
dsloc = dset.sel(longitude=lonsel, latitude=latsel, method='nearest')

# select a variable to plot
dsloc['t2m'].plot()
```

Out[6]:
```
[<matplotlib.lines.Line2D at 0x78614bfc7d70>]
```



2) Use a function to automate the download of files for a particular region & time function

The following example will now be based on a generic function to retrieve ERA5 rainfall and temperature data from the CDS and create a single rainfall and temperature file for Cyprus. Note that there are several variables and option to send an API request to the CDS. More details for ERA5 daily data can be found at https://confluence.ecmwf.int/display/CKB/ERA5+family+post-processed+daily+statistics+documentation

In [7]:
```
def download_era5_data(varlist, latmin, latmax, lonmin, lonmax, year_start, year_end,

# Output directory = input climate data
    if not(os.path.isfile(outdir)):
        os.system("mkdir -p "+outdir)

    years_vector = np.arange(year_start, year_end+1, 1)
    years = [str(years_vector) for years_vector in years_vector]
    yearnow   = datetime.now().year
    monthnow  = datetime.now().month

##########################################################

    dataset = "derived-era5-single-levels-daily-statistics"
    for var in varlist:
        for yr in years:
            if yr == str(yearnow):
                nmonth = monthnow - 1
            else:
                nmonth = 12
        months_vector = np.arange(1, nmonth +1 , 1)
        months = [f"{months_vector:02}" for months_vector in months_vector]

        for mn in months:
            outfile = var+"_1d_"+yr+"_"+mn+"_ERA5.nc"
```

```
            outfile = outdir + "/" +outfile
            ndays = monthrange(int(yr), int(mn))[1]
            days_vector = np.arange(1, ndays+1, 1)
            days = [f"{days_vector:02}" for days_vector in days_vector]
            request = {
                'product_type': ['reanalysis'],
                'variable': var,
                'year': yr,
                'month': mn,
                'day': days,
                'data_format': 'netcdf',
                'grid': gridres,
                'area': [latmax, lonmin, latmin, lonmax],
                "daily_statistic": "daily_mean",
                "time_zone": "utc+00:00",
                "frequency": "1_hourly"
            }
            if not(os.path.isfile(outfile)):
                client = cdsapi.Client()
                client.retrieve(dataset, request, outfile)
```

In [ ]:

```
############################################################
# Next block calls the function we defined earlier
############################################################

#varlist   = ["2m_temperature", "total_precipitation"] # variables (temperature and r
varlist   = ["total_precipitation"] # variables (rainfall only)

# Define domain (Cyprus)
latmin = 34.5
latmax = 35.75
lonmin = 32.25
lonmax = 34.75

# Define start year for ERA5 data - test for 2024
year_start = 2024
year_end   = 2024

# Spatial resolution of ERA5 data 0.25, 0.5 or 1 deg res
gridres = "0.25/0.25"

# Call the former function to automate the data download process
download_era5_data(varlist, latmin, latmax, lonmin, lonmax, year_start, year_end, gri
```

3) Example 2 - Download the CERRA high res dataset from the CDS

We will now download temperature data for Europe based on the CERRA data
https://doi.org/10.24381/cds.622a565a

In the Dwonload Tab Select the following:

Variable - 2m temperature

Level Type - Surface

Data Type - Reanalysis

Product type - Analysis

Year 2024

Month August

Day Select all

Time Select all

Data Format netcdf

Accept terms of conditions

Then you just need to copy and paste the API request (example below)

In [9]:
```
# Define output directory and file name
outfile = "example2_CERRA.nc" # Name of the output netcdf file

# Create output directory if it does not exist
if not os.path.exists(outdir):
    os.mkdir(outdir)
outfile = outdir + "/" + outfile

################################################################################
# Copy and paste request from website below (1st example is provided below)
################################################################################

dataset = "reanalysis-cerra-single-levels"
request = {
    "variable": ["2m_temperature"],
    "level_type": "surface_or_atmosphere",
    "data_type": ["reanalysis"],
    "product_type": "analysis",
    "year": ["2024"],
    "month": ["08"],
    "day": [
        "01", "02", "03",
        "04", "05", "06",
        "07", "08", "09",
        "10", "11", "12",
        "13", "14", "15",
        "16", "17", "18",
        "19", "20", "21",
        "22", "23", "24",
        "25", "26", "27",
        "28", "29", "30",
        "31"
    ],
    "time": [
        "00:00", "03:00", "06:00",
        "09:00", "12:00", "15:00",
        "18:00", "21:00"
    ],
    "data_format": "netcdf"
}

client = cdsapi.Client()
client.retrieve(dataset, request).download(outfile)
```

```
2025-09-13 15:27:24,801 INFO [2024-09-26T00:00:00] Watch our [Forum](https://forum.ecm
wf.int/) for Announcements, news and other discussed topics.
2025-09-13 15:27:24,973 INFO Request ID is 803e02cd-6e22-43bd-ad98-709a1a399f51
2025-09-13 15:27:25,097 INFO status has been updated to accepted
2025-09-13 15:27:40,051 INFO status has been updated to successful
7a297ae5313e1daa7a21dcc4e907051.nc:   0%|          | 0.00/435M [00:00<?, ?B/s]
```

Out[9]: 'Data/example2_CERRA.nc'

Use ncdump to visualize file information and CDO to manipulate files

ncdump and ncgen are useful tools to visualize the content of a netcdf file Note that these tools function on MacOsX (brew) or Linux based system - instructions about installing ubuntu on a Windows machine and the other software are provided at the end We will use a linux console to showcase ncdump and CDO functionalities

On the Jupyter notebook in Cyprus you can open a linux console (File -> New -> console) Otherwise you will have to install ubuntu on Windows using the information provided at the end

ncdump - ncview

We will now use the example files we downloaded earlier

First you need to be in the Data directory that should be in the current directory:

cd Data (change directory) in a console

ls -la (list files in the current directory)

Then you can print information about the header of one netcdf file by typing:

ncdump -h example1_ERA5.nc

You will see all variable names, their associated dimensions and attributes

We can also print te values of a specific variable, for example try:

ncdump -v latitude example1_ERA5.nc

This command will pring the latitude values in the file (from 90N to 90S by 0.25deg increment)

To check the version of the Netcdf file type:

ncdump -k example1_ERA5.nc

This command should return netCDF-4, the version of the netcdf file

More details about the use of ncdump is provided on Adrian Tompkins (ICTP) Youtube channel at

https://www.youtube.com/watch?v=ggp6pEHllgU

A tutorial is also available at https://ncar.github.io/CESM-Tutorial/notebooks/resources/netcdf.html

CDO

The Climate Data operator is a powerful tool to manipulate and process climate data and netcdf files (interpolation, computatio, file concatenation, time averaging and statistics).

CDO is based on operators and is usually called by typing

cdo operatorname infile.nc outfile.nc

First we will concatenate daily files, the generic syntax is:

cdo mergetime list_of_files*.nc outfile.nc

in the Data directory type:

cdo mergetime total_precipitation_1d_* precip_2024.nc

This command will concatenate all data into a single file

You can now check that the files have been merged by typing

ncdump -h precip_2024.nc

The time dimension = 366 - hence we have concatenated rainfall data files into a single record for 2024. Usually you can then remove the other files and only work with your 2024 data

We can calculate monthly means using the monmean operator, the generic syntax is:

cdo monmean infile.nc outfile.nc

For our example, in the Data directory type:

cdo monmean precip_2024.nc precip_2024_monmean.nc

Same, if you now type 'ncdump -h precip_2024_monmean.nc', the time dimension is now = 12 (monthly data for 2024)

We will now use the sellonlatbox to spatially subset global data (example1_ERA5.nc). The generic syntax is:

cdo sellonlatbox, lonmin,lonmax,latmin,latmax infile.nc outfile.nc

For Europe, type in the data directory:

cdo sellonlatbox,-15.,30.,25.,60. example1_ERA5.nc example1_ERA5_Europe.nc

We have now subset the European region from the global data in example1_ERA5_Europe.nc.

If you have installed ncview (see instructions at the end for installation) you can rapidly visualyze the data using

ncview example1_ERA5_Europe.nc &

You can also use CDO for spatial interpolation. We will now interpolate the CERRA data(2D lat-lon grid) onto the ERA5 data grid (regular 0.25deg grid) using bilinear interpolation. The generic syntax is:

cdo remapbil,targetgridfile.nc infile.nc infile_interp.nc

For our example, in the data directory type:

cdo remapbil,example1_ERA5_Europe.nc example2_CERRA.nc example2_CERRA_interp.nc

The CERRA data has been interpolated onto the ERA5 grid for Europe

You can check the new file dimensions by typing

ncdump -h example2_CERRA_interp.nc

or use ncview to visualyze the interpolated data:

ncview example2_CERRA_interp.nc &

More details about the use of CDO is provided on Adrian Tompkins (ICTP) Youtube channel at

https://www.youtube.com/watch?v=79o6DXr_3zM

https://www.youtube.com/watch?v=E-ehL3N0Cjo

In [13]:
```
# CDO is a powerful tool to massively process climate data files and can be combined
# processing or by using Python or R and using system commands (os.system()  in Pytho

# For example, we can use ncdump directly into this notebook using a system command:

command =  "ncdump -h ./Data/example2_CERRA.nc" # command in string
os.system(command) # will launch the command directly into Python but you need to use
```

```
netcdf example2_CERRA {
dimensions:
        valid_time = 248 ;
        y = 1069 ;
        x = 1069 ;
variables:
        int64 valid_time(valid_time) ;
                valid_time:long_name = "time" ;
                valid_time:standard_name = "time" ;
                valid_time:units = "seconds since 1970-01-01" ;
                valid_time:calendar = "proleptic_gregorian" ;
        double latitude(y, x) ;
                latitude:_FillValue = NaN ;
                latitude:units = "degrees_north" ;
                latitude:standard_name = "latitude" ;
                latitude:long_name = "latitude" ;
        double longitude(y, x) ;
                longitude:_FillValue = NaN ;
                longitude:units = "degrees_east" ;
                longitude:standard_name = "longitude" ;
                longitude:long_name = "longitude" ;
        string expver(valid_time) ;
        float t2m(valid_time, y, x) ;
                t2m:_FillValue = NaNf ;
                t2m:GRIB_paramId = 167LL ;
                t2m:GRIB_dataType = "an" ;
                t2m:GRIB_numberOfPoints = 1142761LL ;
                t2m:GRIB_typeOfLevel = "heightAboveGround" ;
                t2m:GRIB_stepUnits = 1LL ;
                t2m:GRIB_stepType = "instant" ;
                t2m:GRIB_gridType = "lambert" ;
                t2m:GRIB_uvRelativeToGrid = 1LL ;
                t2m:GRIB_DxInMetres = 5500. ;
                t2m:GRIB_DyInMetres = 5500. ;
                t2m:GRIB_LaDInDegrees = 50. ;
                t2m:GRIB_Latin1InDegrees = 50. ;
                t2m:GRIB_Latin2InDegrees = 50. ;
                t2m:GRIB_LoVInDegrees = 8. ;
                t2m:GRIB_NV = 214LL ;
                t2m:GRIB_Nx = 1069LL ;
                t2m:GRIB_Ny = 1069LL ;
                t2m:GRIB_cfName = "air_temperature" ;
                t2m:GRIB_cfVarName = "t2m" ;
                t2m:GRIB_gridDefinitionDescription = "Lambert conformal" ;
                t2m:GRIB_iScansNegatively = 0LL ;
                t2m:GRIB_jPointsAreConsecutive = 0LL ;
                t2m:GRIB_jScansPositively = 1LL ;
                t2m:GRIB_latitudeOfFirstGridPointInDegrees = 20.292281 ;
                t2m:GRIB_latitudeOfSouthernPoleInDegrees = -90. ;
                t2m:GRIB_longitudeOfFirstGridPointInDegrees = 342.514057 ;
                t2m:GRIB_longitudeOfSouthernPoleInDegrees = 0. ;
                t2m:GRIB_missingValue = 3.40282346638529e+38 ;
                t2m:GRIB_name = "2 metre temperature" ;
                t2m:GRIB_shortName = "2t" ;
                t2m:GRIB_units = "K" ;
                t2m:long_name = "2 metre temperature" ;
                t2m:units = "K" ;
                t2m:standard_name = "air_temperature" ;
                t2m:GRIB_heightAboveGround = 2. ;
                t2m:coordinates = "valid_time latitude longitude expver" ;

// global attributes:
                :GRIB_centre = "eswi" ;
                :GRIB_centreDescription = "Norrkoping" ;
                :GRIB_subCentre = 255LL ;
                :Conventions = "CF-1.7" ;
```

```
                :institution = "Norrkoping" ;
                :history = "2025-09-12T13:39 GRIB to CDM+CF via cfgrib-0.9.15.0/ecCode
s-2.42.0 with {\"source\": \"tmppojo9ag7/data.grib\", \"filter_by_keys\": {}, \"encode
_cf\": [\"parameter\", \"time\", \"geography\", \"vertical\"]}" ;
}
```

Out[13]: 0

Set Up ubuntu on Windows 10-11 and install important softwares (CDO, NCO, netcdf libraries)

The following shows how to concatenate the files that were downloaded from the CDS for Cyprus and do a few plots

Unfortunately, CDO and ncdump are mostly available on Linux/MacOSX Operating systems and binaries are not provided for Windows

We will now use a terminal to access the loal Linux system and use cdo with command lines (details to install useful tools on Windows are provided below).

1) Installing Linux on Windows with WSL

Open PowerShell or Terminal as an Administrator Click the Start button and search for "Terminal" or "PowerShell". Right-click the top result and select Run as administrator. Confirm any User Account Control (UAC) prompts. Run the Installation Command In the administrator terminal, type the following command and press Enter: bash wsl --install This command will perform several actions: Enable the necessary Windows optional features. Download and install the latest WSL kernel. Set WSL2 as the default environment. Download and install the default Linux distribution (Ubuntu) from the Microsoft Store. Restart Your Computer After the command finishes, you'll be prompted to restart your computer to complete the installation. Set Up Your Linux Distribution Once your computer restarts, a Linux terminal will automatically open to begin the setup process. You will be asked to create a Unix username and password. Enter your desired username and then create and confirm a password for it. After Installation Your Linux distribution is now ready to use. You can access it by opening the Start menu and searching for the name of your distribution, such as "Ubuntu". For a full list of available Linux distributions, open an administrator terminal and run wsl --list --online. To install a different distribution, use the command wsl --install -d , replacing with the desired distribution's name from the list.

2) Installing ncdump (NetCDF Tools)

Step 1: Install NetCDF Install the NetCDF tools in a linux terminal:

sudo apt install netcdf-bin

Step 2: Verify the Installation Check if the installation was successful by running: ncdump -h

3) Installing CDO (Climate Data Operators)

Step 1: Open the WSL Terminal Launch your WSL terminal (e.g., Ubuntu).

Step 2: Update Packages Run the following command to update your package list:

sudo apt update && sudo apt upgrade

Step 3: Install CDO by running:

sudo apt install cdo

Step 4: Verify the Installation Check the installed version of CDO:

cdo -V

3) Installing ncview (viewer for netcdf files)

For ubuntu OS:

sudo apt-get install ncview

For MacOSX

brew install ncview

Then you can type in a terminal:

ncview file.nc &

To visualize the data

In [ ]: