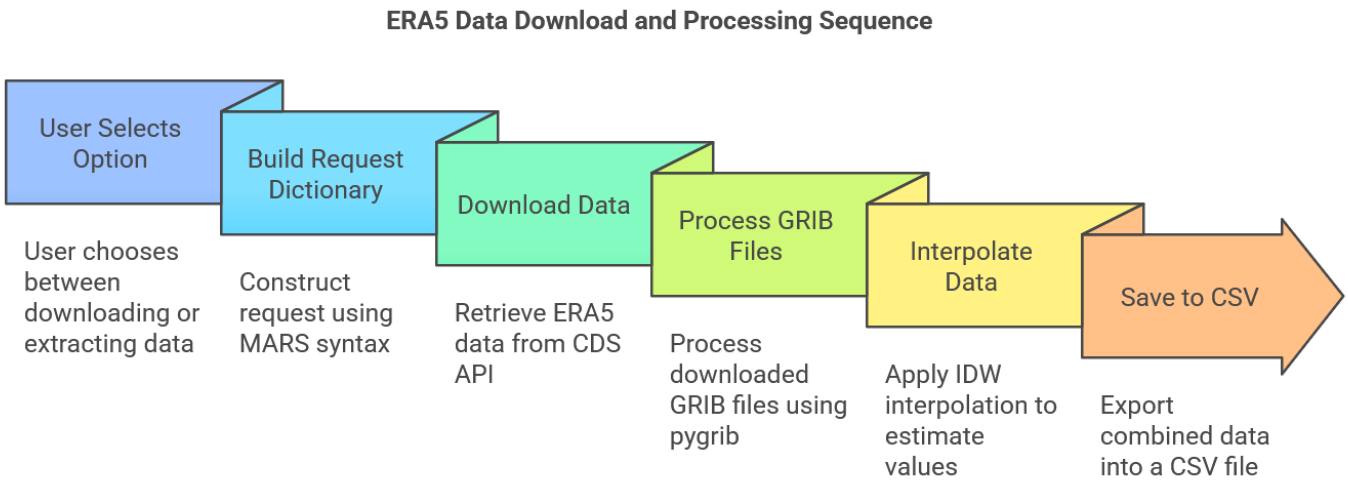
ERA5 Hourly Data Downloader and Extractor

Overview:

This script is designed to work with ERA5 reanalysis data from ECMWF using both the CDS API and the MARS (Meteorological Archive and Retrieval System). MARS is ECMWF's archive retrieval system that enables users to request data using a strictly defined syntax. Detailed information on MARS request syntax and best practices can be found in the official MARS User Documentation.



The script supports two operational modes:

1. **Download & Process:**
    - Downloads ERA5 data in monthly chunks from the CDS API (using MARS syntax rules).
    - Processes the resulting GRIB files to extract a pre-defined set of meteorological and oceanographic variables using Inverse Distance Weighting (IDW) for interpolation.
    - Saves the combined data into a CSV file, sorted by datetime.
2. **Extract Only:**
    - Skips the download phase and directly processes all available GRIB files locally.
    - Uses parallel processing with progress monitoring to efficiently extract data.
    - In addition to matching GRIB messages by their short names, it also extracts data using param IDs when available.

Detailed Functionality:

1. **CDS API and MARS Requests:**
    - The request dictionary is built following the strict syntax required by the MARS system. For example, keys such as `product_type`, `format`, `param`, `year`, `month`, `day`, `time`, `area`, and `grid` must be provided in the correct format.

- This script builds the request using only official ERA5 param IDs (as strings) to avoid ambiguity.
- The `area` key is specified as `[North, West, South, East]` and `time` values are provided in "HH:00:00" format.
- For further details, refer to the MARS User Documentation.

2. **Unified Variable Mapping:**
   - A unified dictionary called `VARIABLES` contains both the official ERA5 param ID and the expected GRIB short name for each variable.
   - From this mapping, a list of param IDs (`PARAM_IDS`) is derived for the CDS API request and a GRIB message key mapping (`GRIB_KEY_MAP`) is created to map the GRIB message short names to internal keys.
3. **GRIB File Processing:**
   - GRIB files are processed using the `pygrib` library.
   - For each GRIB message, the script first attempts to match the short name to our internal keys.
   - If the short name is not found, it falls back to comparing the GRIB message's parameter number (if available) to the expected param IDs.
   - The script uses Inverse Distance Weighting (IDW) interpolation to estimate the value at the target coordinate.
   - Extracted data from all GRIB files are combined into a pandas DataFrame, sorted by datetime, and exported as a CSV file.
4. **Robust Error Handling and Logging:**
   - Detailed logging records each major step and any encountered issues.
   - The download process is retried multiple times with increasing delay intervals if failures occur.
   - After processing, the script checks for missing monthly GRIB files and issues warnings accordingly.

Usage:

When executed, the user is prompted to choose between:

- **Option 1:** Download ERA5 data via the CDS API (using MARS syntax) and process the downloaded GRIB files.
- **Option 2:** Only process existing GRIB files in the data directory.

Installation:

To run `download_era5_data.py`, you need to install Python 3.x and several libraries. Additionally, `ECCODES` is a crucial non-Python dependency for `pygrib`.

1. Install ECCODES:

`ECCODES` is a software package developed by ECMWF for processing WMO FM-92 GRIB, WMO FM-94 BUFR, and WMO CREX messages. It is required for `pygrib` to function correctly.

`ECCODES` can be installed via `conda` or `pip`, or built from source.

Using Conda (Recommended):

```
conda install -c conda-forge eccodes
```

Using Pip:

```
pip install eccodes
```

For more detailed instructions and alternative installation methods, refer to the official ECMWF ECCODES documentation: ECMWF ECCODES Installation and ECCODES with Python Bindings in Conda.

2. Set up CDS API:

To use the CDS API for downloading data, you need to register on the Copernicus Climate Data Store (CDS) and set up your personal API key.

1. **Register:** Go to the Copernicus Climate Data Store website and create an account if you don't already have one.
2. **Get your API Key:** After logging in, navigate to your user profile page. You will find your UID and API key there.
3. **Configure API access:** The `cdsapi` library needs to know your UID and API key. The recommended way to do this is to create a file named `.cdsapirc` in your home directory (e.g., `C:\Users\YourUsername\.cdsapirc` on Windows, or `~/.cdsapirc` on Linux/macOS). The content of this file should be:

   ```
   url: [https://cds.climate.copernicus.eu/api/v2]
   (https://cds.climate.copernicus.eu/api/v2)
   key: <YOUR_UID>:<YOUR_API_KEY>
   ```

   Replace `<YOUR_UID>` with your actual User ID and `<YOUR_API_KEY>` with your API key. For more detailed instructions, refer to the CDS API documentation.

3. Set up a Python Virtual Environment and Install Dependencies:

It's highly recommended to use a virtual environment to manage dependencies for this project. This isolates the project's dependencies from your system's global Python packages, preventing conflicts. You can choose between `venv` (Python's built-in tool) or `conda` (a powerful package and environment manager).

Option A: Using `venv` (Python's built-in virtual environment tool)

1. **Navigate to your project directory** in the terminal.
2. **Create a virtual environment:**

   ```
   python3 -m venv era5env
   ```

   (You can replace `era5env` with your preferred environment name.)
3. **Activate the virtual environment:**
   - **On Windows:**

     ```
     .\era5env\Scripts\activate
     ```

   Your terminal prompt should change to indicate that the virtual environment is active (e.g., `(venv_era5_data) user@host:~`).

4. **Install the required Python packages:**

   ```
   pip install cdsapi numpy pandas pygrib tqdm
   ```

Option B: Using `conda` (Recommended for scientific stack)

1. **Ensure Conda is installed:** If you don't have Conda (Miniconda or Anaconda), download and install it from the official website.
2. **List existing Conda environments** (optional, to see what's already there):

   ```
   conda info --envs
   ```

3. **Create a new Conda environment:**

   ```
   conda create --name era5env python=3.9
   ```

   (You can choose a different Python version, e.g., `python=3.10`. `era5env` is the name of the environment.)

4. **Activate the new Conda environment:**

   ```
   conda activate era5env
   ```

   Your terminal prompt should change to `(era5env) user@host:~`.

5. **Install the required Python packages into the active Conda environment:**

   ```
   conda install -c conda-forge cdsapi numpy pandas pygrib tqdm
   ```

   Using `-c conda-forge` is often recommended for scientific packages with Conda, as it provides pre-compiled binaries.

6. **Verify installed packages** (optional, to confirm installation):

   ```
   conda list
   ```

ECMWF Data Information:

- **Website:** ECMWF
- **ERA5 reanalysis dataset:** ERA5 Dataset
- **Parameter reference:** Parameter Database

For more detailed information on CDS API and MARS request syntax, please refer to the MARS User Documentation.