National Ocean Service
Center for Operational Oceanographic Products and Services

USER GUIDE

Accessing and Downloading **Coastal Ocean Reanalysis** Data

CORA

The **Coastal Ocean Reanalysis (CORA)** is a NOAA dataset that blends tide gauge observations with ocean model output to estimate historical water levels, even in areas without long-term observations. CORA provides hourly data from 1979 to 2022, including water levels and wave conditions, at high spatial resolution along the U.S. coast. These data support flood risk assessment, planning, and research. The CORA User Guide helps you **access this data in the cloud** using GitHub-hosted Jupyter Notebooks notebooks, walking you through setup and data filtering. Additionally, you will find instructions to **download the data in bulk** from Amazon Web Services (AWS) through AWS Command Line Interface (CLI) and Globus.

CONTACT US

www.tidesandcurrents.noaa.gov
tide.predictions@noaa.gov

Table of Contents

What's in this guide?

Table of Contents

What's in this guide?

About This Guide

The **Coastal Ocean Reanalysis (CORA)** combines NOAA tide gauge observations with hydrodynamic and wave model output to create a continuous record of coastal water levels from 1979 to 2022, improving data coverage between gauges.

The datasets include:

- *Hourly values for water levels*
- *Significant wave height*
- *Peak wave period*
- *Mean wave direction*
- *Interpolated water level data on a 500-meter coastal grid for uniform spatial resolution*

CORA datasets are hosted and served through ***NOAA's Open Data Dissemination*** platform, a cloud-based repository that can house large amounts of data.

CORA datasets have been optimized so users can query, spatially and temporally without the need for downloading (though that is still an option for users). While this guide focuses on the **Accessing Data** GitHub-hosted notebook, the same setup and workflow can be used with other CORA notebooks.

This guide is broken down into two sections:

1. **Accessing Data in the Cloud:** The **first section** will walk you through **retrieving CORA data** in the cloud using GitHub-hosted Jupyter Notebooks on Windows operating systems. By the end, you will generate a time series graph of historical water levels at your chosen location, along with access to the underlying data used to create it.

2. **Bulk Downloads:** The second section, meant for more technical users, will show you different options for downloading the **full time series** of CORA data.

Whether you are new to working with data or an experienced user, step-by-step instructions are provided throughout. **Appendix I** provides additional guidance.

Accessing Data in the Cloud

Preparing to Run the Notebook

This section walks you through accessing, analyzing, and visualizing CORA data using step-by-step GitHub-hosted Jupyter notebooks. These notebooks are designed to help most users work with CORA data efficiently – no large downloads required.

If you're new to running notebooks or working with code, be sure to check out **Appendix I**. It gathers all the coding tips and tricks from this guide in one place, making it a helpful reference for beginners.

Before you begin, you'll need to download a Python environment and package manager, such as Anaconda Distribution[1], which comes with:

- Anaconda Navigator, a desktop application,
- Built-in support for using Jupyter Notebooks, and
- Anaconda Prompt (conda) command-line interface through the Anaconda Navigator

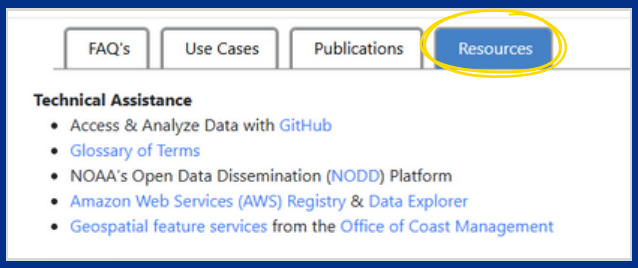
You will need all of these downloads to complete this user guide. No need to open these right now, but make sure you have access. Note that while this guide is written using ***Windows operating systems*** and ***Anaconda Distribution***, there are other options available and we encourage you to use the software that fits your needs.

Once you have successfully completed **Preparing to Run the Notebook** for the first time, you can skip these steps next time. See **Appendix II: Continuing to Use the Notebooks** for streamlined instructions for your next use.

[1]As a government agency, CO-OPS does not endorse or recommend any specific brand, product, or platform, including those used for coding or data analysis. Any third-party tools mentioned in this user guide, such as Anaconda Distribution, are included solely to demonstrate one possible workflow for accessing CORA data. Users are encouraged to explore other available options and choose the software or tools that best meet their individual needs.

1 Getting to the Data

To access the GitHub notebooks, **navigate to the CORA landing page**: https://tidesandcurrents.noaa.gov/cora.html

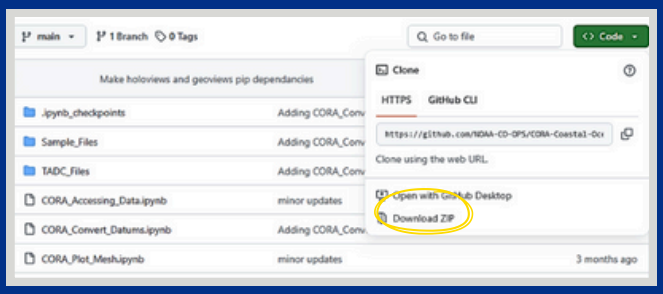


Scroll down to the **Resource** tab, then click on the **GitHub link**. You will be redirected to the NOAA CO-OPS GitHub for CORA, away from a government site.

Alternatively, use this link to navigate to the GitHub repository: https://github.com/NOAA-CO-OPS/CORA-Coastal-Ocean-Reanalysis-CORA

Downloading GitHub Notebooks

Once on the GitHub page, click the green "**Code**" dropdown and choose "**Download Zip**." The file will automatically start downloading and will be saved as "**CORA-Coastal-Ocean-Reanalysis-CORA-main.zip**" in your default downloads folder.



If you'd like to clone the notebooks using Git command prompt, navigate to the README.md file in the GitHub repository, scroll to the "Usage" heading and follow the instructions.

Go to your "Downloads" folder and unzip (or extract) the file. Save the extracted folder to a location on your computer that you can easily find later. For the examples in this guide:

- The files will be saved in a folder called "**CORA Notebooks**" on the Desktop.
- The file path will be **C:\Users\User.Name\Desktop\CORA Notebooks**.

The GitHub Notebooks are now available on your computer and you can move onto **Getting Ready to Run the Notebook**.

2 Getting Ready to Run the Notebook

This page summarizes the key steps for getting ready to run the CORA notebooks. More experienced users may use this as a quick-start checklist, while new users can use it as a reference to help you keep track of where you are in the process. Each step is explained in more detail on the pages that follow, so if you need additional guidance at any point, continue reading for step-by-step instructions.

Copy the pathway to your CORA folder

- Open the folder where you saved your CORA notebooks
- Copy pathway

Connect Anaconda to the CORA Notebooks

- Open Anaconda prompt
- Type **cd** and paste your file pathway
- Press Enter

Create CORA Environment

- Type **condaenv create - f environment.yml**
- Press Enter
- Wait for installation

Activate the CORA Environment

- Type **conda activate cora**
- Press Enter

Launch Jupyter Notebook

- Type **Jupyter notebook**
- Press Enter
- Wait for Jupyter notebook to launch in browser

Open "Accessing Data" Notebook

- Open file: CORA_Accessing_Data.ipynb
- Move onto **Accessing Data in the Cloud**

Getting Ready to Run the Notebook *(continued...)*

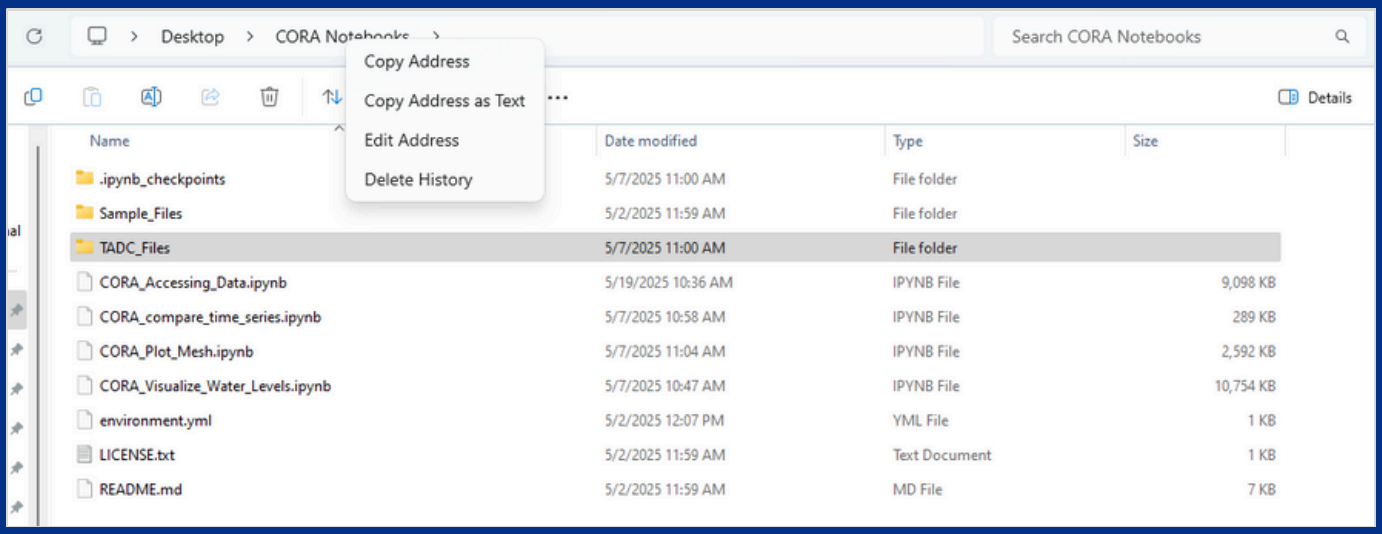
Before you run your notebook, you need to set up your coding virtual environment to install Python libraries and packages necessary for the code to execute seamlessly.

A coding environment is a self-contained workspace with the specific tools your notebook needs. For this project, you'll set up an environment that includes all the Python libraries required to run the **CORA_Accessing_Data.ipynb** notebook. This will help you avoid errors by making sure everything is compatible and ready to go.

Copy the Folder Pathway

First, you need to navigate to the folder where you saved your CORA GutHib notebooks, then save the file pathway. Here's how to do that:

1. Open the folder on your computer where you saved the CORA Notebooks.
2. Once in the folder, you'll see a path at the top of the window (pictured below). Right-click on the folder name.
3. Select "**Copy Address as Text**" or "**Copy as path**" depending on your system. You will paste this into Anaconda Prompt in the next step.



Getting Ready to Run the Notebook *(continued...)*

Connect Anaconda to CORA Notebooks

Open **Anaconda Prompt** (you can find it by searching for "Anaconda Prompt" in your desktop search bar). You will see a black screen with a blinking cursor (pictured below).

Once Anaconda Prompt is open, you'll need to tell Anaconda where your CORA Notebooks are saved by navigating to that folder using a file path. To do this, start by typing:

```
cd
```

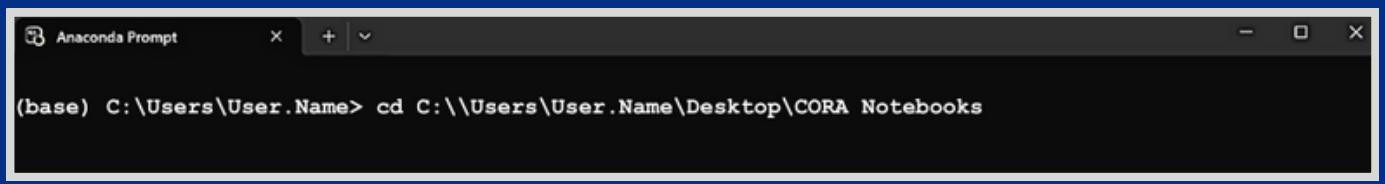
cd stands for change directory. It tells the system to go into the folder that you specified.

Next to **cd**, paste your file pathway. Our example file pathway is:

```
C:\Users\User.Name\Desktop\CORA Notebooks
```

The final product will look something like:

```
(base) C:\Users\User.Name> cd C:\Users\User.Name
\Desktop\CORA Notebooks
```



Press **Enter** on your keyboard. Anaconda Prompt is now working within the folder you copied.

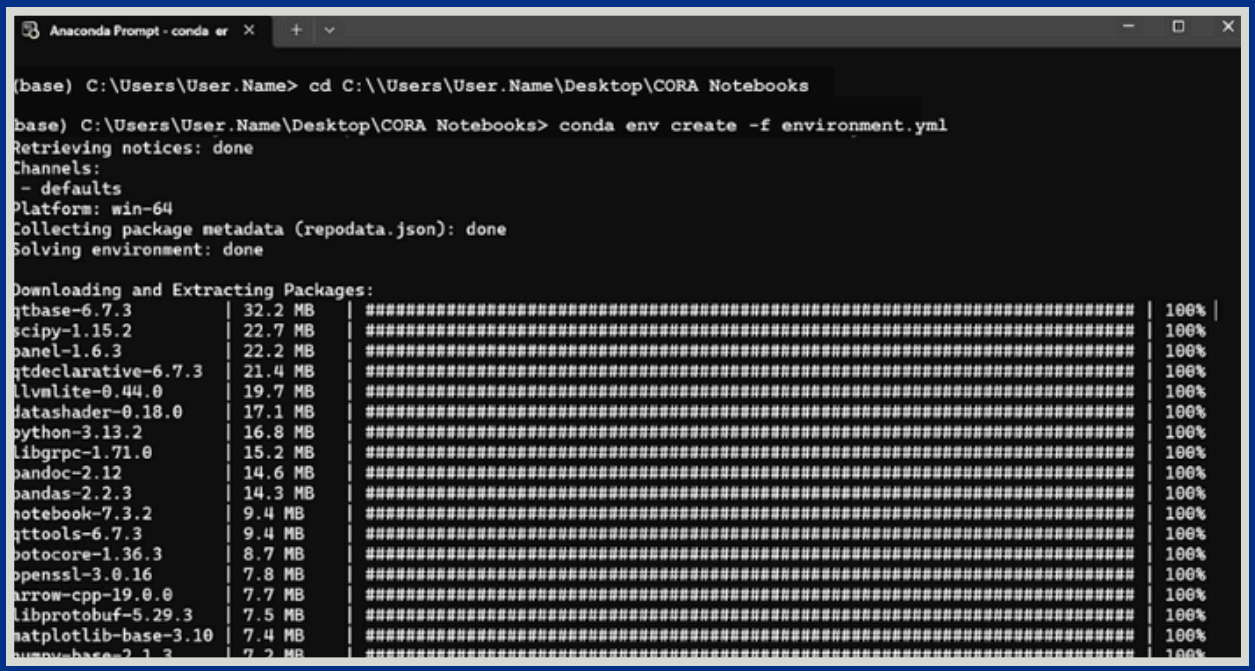
Getting Ready to Run the Notebook *(continued...)*

Create the CORA Environment

Now that you're in the CORA Notebooks folder in Anaconda Prompt, it's time to create your environment. Type the following command into Anaconda Prompt and press Enter:

```
conda env create -f environment.yml
```

Wait while your environment installs. This step may take 5–30 minutes, depending on your internet speed and computer. Do not lock your computer during this process. You'll see a lot of text as it installs various packages. This is normal.

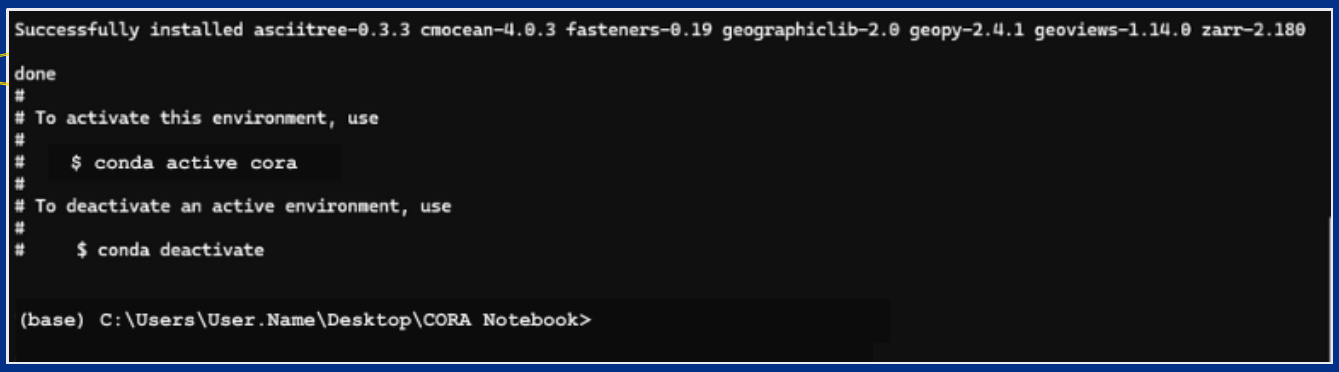


When you run the **conda env create -f file_name command**, Conda is reading the environment.yml file and installing all the required packages listed in it. These packages are needed to run the CORA notebook correctly. This process sets up a separate, isolated environment so that everything is properly configured and self-contained. The text you see in the prompt shows the progress of each package being downloaded and installed.

Getting Ready to Run the Notebook *(continued...)*

Activate the CORA Environment

Once the installation is complete, you will see this screen:



You can now **activate your new environment** by typing:

```
conda activate cora
```

Then press **Enter**. You will see your **"(base)" change to "(cora)"**. This indicates you are successfully in the new environment.

Launch Jupyter Notebook

Finally, launch the notebook by typing:

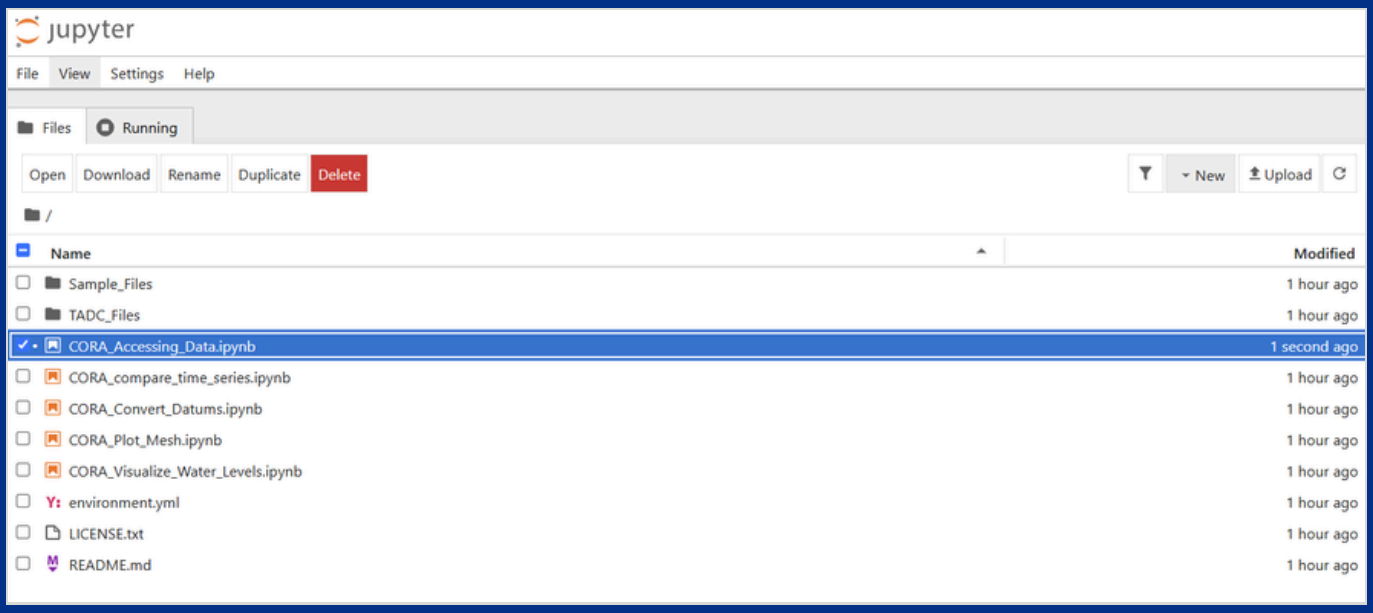
```
jupyter notebook
```

Press **Enter**. This will open the Jupyter Notebook in a new tab in your default web browser (you may be prompted to choose an application to open the file). It may take a minute or two to launch.

Getting Ready to Run the Notebook (continued...)

Open the "Accessing Data" Notebook

Once your **Jupyter Notebook** has launched, you should see multiple files, including five notebooks. Open the file called **CORA_Accessing_Data.ipynb** by clicking on it. It will launch in a new tab.



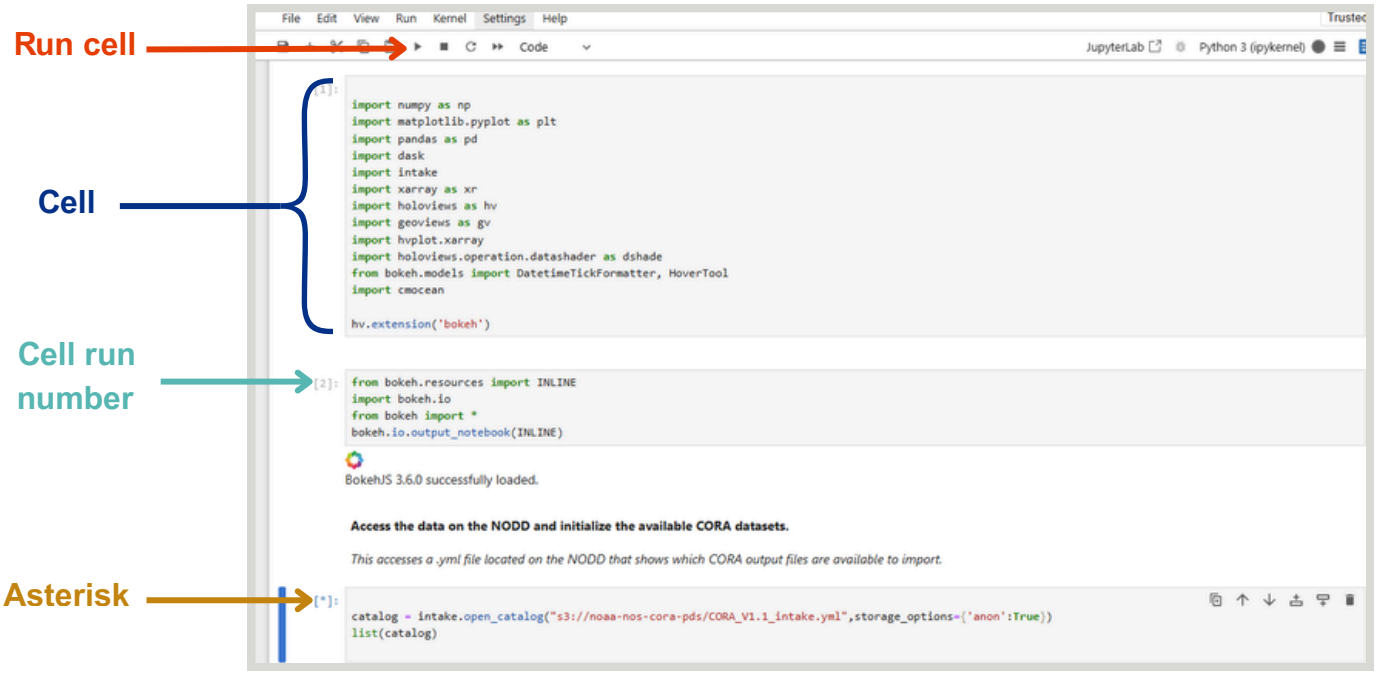
You are ready to start running the notebook.

Accessing Data in the Cloud

Navigating the GitHub-Hosted Jupyter Notebook

1 Import Libraries

To begin running the notebook, start in the first cell to import some libraries that you need to run this notebook. Press **Ctrl + Enter** or click the "▶" button at the top of the screen to run the cell.



When the cell is done running, the asterisk will be replaced by a number. When that happens you can move onto the next cell that also installs more packages. Click in the cell and run (**Ctrl + Enter**) to import packages.

2 Access NOAA Open Dissemination Data (NODD)

This cell will help you **access the data from the NOAA Open Data Dissemination (NODD)** program and initialize the available CORA datasets. **This step connects your notebook to the Amazon Web Services cloud storage and initializes the CORA data for your selected location.** You don't need to go to the NODD to download anything manually, the notebook handles the connection and works with the data for you.

To access the data on the NODD, run this cell:

```
catalog = intake.open_catalog("s3://noaa-nos-cora-pds/CORA_V1.1_intake.yml",storage_options={'anon':True})
list(catalog)
```

After running the cell, you'll see a catalogue showing the available CORA datasets, including:

- **CORA-V1.1-fort.63:** Hourly water levels from model nodes (native grid)
- **CORA-V1.1.maxele.63.nc:** maximum water level elevation, at all mesh nodes
- **CORA-V1.1-swan_DIR.63:** Hourly mean wave direction
- **CORA-V1.1-swan_TPS.63:** Hourly peak wave periods
- **CORA-V1.1-swan_HS.63:** Hourly significant wave heights
- **CORA-V1.1-Grid:** Hourly water levels interpolated from model nodes to 500-meter resolution coastal grid
- **-timeseries** datasets are optimized for pulling long time series (greater than a few days). For up to a few days of data, use the regular dataset (not labeled **'-timeseries'** in the catalog description).

For more information about these datasets, see the **NOAA's Coastal Ocean Reanalysis: Gulf of Mexico, Atlantic, and Caribbean** technical report.

3 Create an xarray Dataset

Now that you have access to all the datasets on the NODD, this step will help you **create a dataset that you can work with**. The code uses **Dask** to split the CORA data into chunks, which helps manage large files efficiently without using too much memory.

Once the data is initialized, we'll convert it into a **Dask-backed xarray dataset**. xarray is a tool that makes it easier to work with multi-dimensional data (like time, latitude, longitude, etc.) by labeling each dimension, so you can access, manipulate, and analyze the data more easily.

To create the Dask-backed xarray dataset, **run this cell**:

```
ds = catalog["CORA-V1.1-fort.63-timeseries"].to_dask()
ds
```

?

What is Dask?

Dask is a Python library that makes it possible to work with large datasets - even ones too big to fit into your computer's memory - by breaking them into smaller pieces and loading only what's needed, when it's needed.

In the CORA notebooks, Dask plays a key role in helping you work with more than 20 TB of data stored in the cloud. When you run cells early in the notebook, you're not immediately downloading or loading all that data into memory. Instead, you're setting up a reference to the data without actually pulling it in yet.

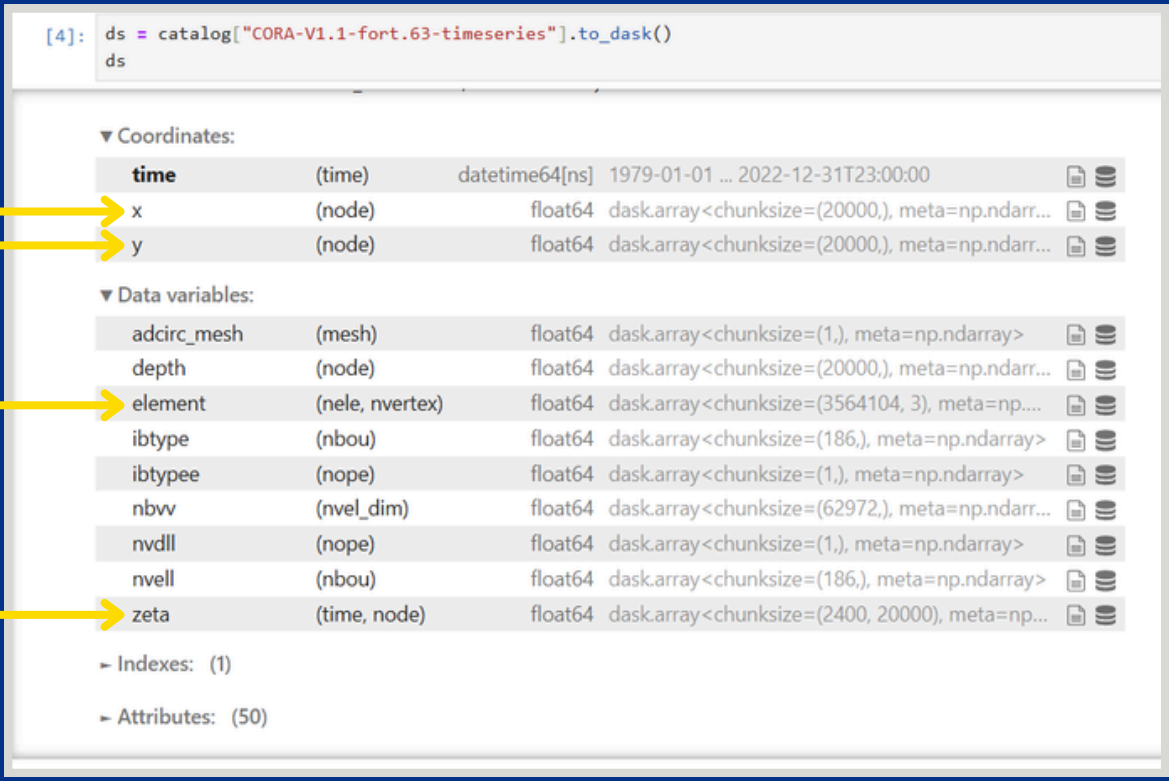
The data only gets loaded into memory when you explicitly tell the notebook to do so, using the .compute() command near the end of the notebook. This delayed execution (known as lazy evaluation) is what allows you to explore and extract small subsets of data (e.g., one location and one year) without overloading your system.

For instance, if you're only interested in hourly water levels near Marblehead, MA from 2014 to 2015, there's no need to download the full dataset, which spans the entire East Coast and Gulf over the past 40 years.

Create an xarray Dataset *(continued...)*

Once the code has finished running, an **xarray dataset will appear** showing the available data. At this stage, none of this data has been loaded into the memory on your computer.

latitude
longitude
native grid
water levels



```
[4]: ds = catalog["CORA-V1.1-fort.63-timeseries"].to_dask()
     ds
```

▼ Coordinates:

time	(time)	datetime64[ns]	1979-01-01 ... 2022-12-31T23:00:00
x	(node)	float64	dask.array<chunksize=(20000,), meta=np.ndarr...
y	(node)	float64	dask.array<chunksize=(20000,), meta=np.ndarr...

▼ Data variables:

adcirc_mesh	(mesh)	float64	dask.array<chunksize=(1,), meta=np.ndarray>
depth	(node)	float64	dask.array<chunksize=(20000,), meta=np.ndarr...
element	(nele, nvertex)	float64	dask.array<chunksize=(3564104, 3), meta=np....
ibtype	(nbou)	float64	dask.array<chunksize=(186,), meta=np.ndarray>
ibtypee	(nope)	float64	dask.array<chunksize=(1,), meta=np.ndarray>
nbvv	(nvel_dim)	float64	dask.array<chunksize=(62972,), meta=np.ndarr...
nvdll	(nope)	float64	dask.array<chunksize=(1,), meta=np.ndarray>
nvell	(nbou)	float64	dask.array<chunksize=(186,), meta=np.ndarray>
zeta	(time, node)	float64	dask.array<chunksize=(2400, 20000), meta=np...

► Indexes: (1)

► Attributes: (50)

Most users will be interested in **latitude/longitude, time, and water levels**, while some may also want depth data.

- **Latitude and longitude** are denoted by “x” and “y” respectively under “Coordinates.” These are based on **node numbers**, which some users may need.
- **Water levels** are denoted by the row labeled “Zeta.”

Some users may be interested in:

- **Element**, which represents the **native grid**, an irregular triangle mesh.

The rest are specific to the Advanced Circulation (ADCIRC model) used to develop the CORA model and will not be used by most users.

4 Find the Closest Data Points

Now that you have a dataset that you can work with, you will **create a function to find the nearest model node to any set of coordinates**. You'll use this function later, once you specify the location you're interested in. This sets you up to answer the question: *What were CORA's water levels at the node closest to my location of interest?*

Run the cell:

```
# find the indices of the points in (x,y) closest to the points in (xi,yi)
def nearxy(x,y,xi,yi):
    ind = np.ones(len(xi),dtype=int)
    for i in range(len(xi)):
        dist = np.sqrt((x-xi[i])**2+(y-yi[i])**2)
        ind[i] = dist.argmin()
    return ind
```

This should run fairly quickly.

Provide Coordinates

Now, you have the option to **define your coordinates** (latitude and longitude), so that you can find the closest node to your location. In the notebook, we provide coordinates for Charleston, SC, but you can replace them with coordinates for any other location.

- **Lat** (latitude) - Enter the latitude in decimal degrees (e.g., 32.7765)
- **Lon** (longitude) - Enter the longitude in decimal degrees (e.g., -79.9311)

Once you have changed the latitude and longitude, **run the cell**:

```
[6]:
lat = 32.775
lon = -79.9239
```

5 Match Coordinates to Grid Points

Now, you will **find the closest CORA grid point to your chosen location**. This step compares your latitude and longitude to the CORA model grid and returns the index of the nearest grid point.

Gridded data are interpolated from the native triangular mesh onto a 500m grid with continuous spatial resolution across the coast and between tide stations. This is particularly useful for layering datasets of the same resolution for geospatial analysis and flood mapping.

Run this cell (no need to change anything):

```
[7]: ind = nearxy(ds['x'].values,ds['y'].values,[lon], [lat])
```

6 Extract and Visualize Water Level Data

Now, you can **extract data for specified time**, **create dataframes for saving the data**, and **create a plot**. This step:

- Extracts (loads) water level data (zeta) for a specific time range,
- Creates dataframes for each node, and
- Generates interactive plots of the data using hvplot

This will allow you to visualize water levels over time for each selected node.

Before running the code, you have the option to change the:

- **Start Date:** The beginning of the time range (e.g., "1989-09-01"). CORA data starts in 1979.
- **End Date:** The end of the time range (e.g., "1989-09-30"). Data is available through 2022.

The longer the requested time period, the longer it will take to run this cell. It is recommended that you enter a year or less, but longer time periods can be input.

Extract and Visualize Water Level Data *(continued...)*

Once you have decided on a time period, **run the cell:**

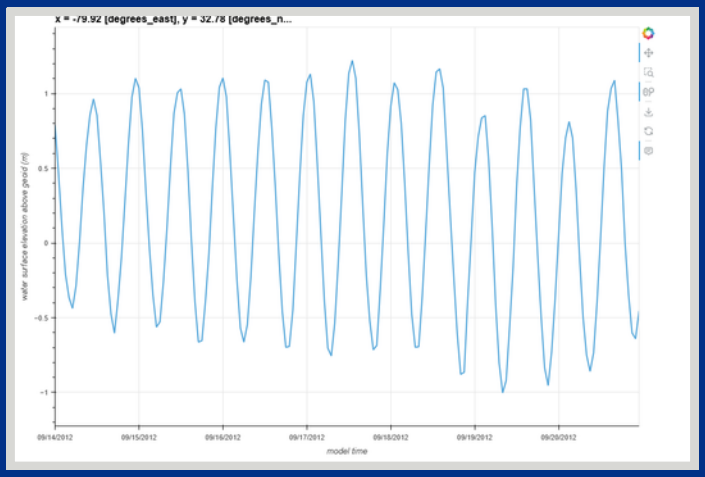
```
[8]: start="2012-09-14"
     end="2012-09-20"

     tickfmt = DatetimeTickFormatter(years="%m/%d/%Y", months="%m/%d/%Y", days = '%m/%d/%Y', hourmin = '%H:%M')
     tooltips = [
         ("time", "@time{%F %T}"),
         ("water level", "@zeta"),
     ]
     hover = HoverTool(tooltips=tooltips,formatters={
             '@time': 'datetime'})

     zeta_tslice = ds['zeta'][:,ind].sel(time=slice(start, end)).compute()

     plot = zeta_tslice.hvplot(x='time', grid=True, xformatter=tickfmt, tools=[hover], width=1000, height=700)
```

You will get a plot that looks something like this (depending on location and time parameters you have entered):



The plot is **interactive**, meaning you can explore the data directly in your browser or Jupyter Notebook. Here are a few features to try:

- **Hover over points** to see the exact timestamp and water level value.
- **Zoom in** by clicking and dragging a box over the time range you want to inspect more closely.
- **Pan across time** by clicking and dragging the plot left or right.
- **Reset the view** using the toolbar on the right side of the plot.

These interactive tools make it easier to explore changes in water levels over time and identify patterns or specific events.

7 Save the Data to a NetCDF File

You have the option to save the water level data you just generated to a NetCDF file by **running this cell**:

```
[ ]: zeta_tslice.to_netcdf('/pathtofile/DataSlice.nc')
```

This creates a file on your computer that stores the data in a standardized, shareable format that can be used in other tools or analyses later.

8 Save the Time-Series Plot

Finally, to save the interactive plot as an HTML or PNG file, **run this cell:**

```
[ ]: hv.save(plot,'/pathtofile/DataPlot.html')
```

This lets you keep a copy of your results that can be opened outside of the notebook—great for sharing or adding to reports.

Want to explore CORA data for a different location or time period?

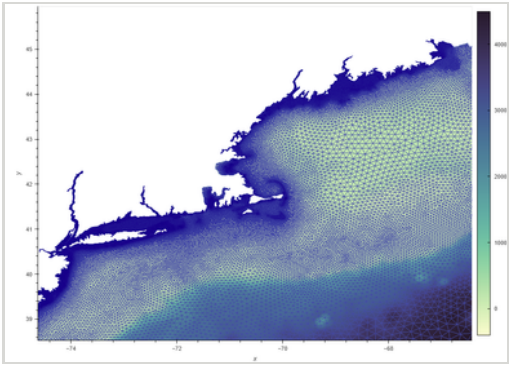
1. Go back to the **Provide Coordinates** step and enter a new latitude and longitude. Run the cell.
2. Re-run the **Match Coordinates to Grid Points** cell.
3. In the **Extract and Visualize Water Level Data** step, update the start and end dates if needed. Then run the cell.

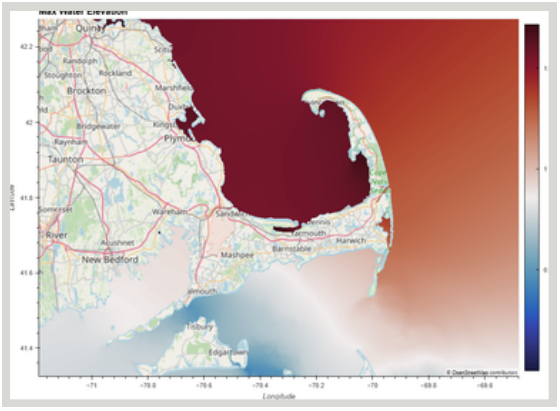
You'll see an updated graph that reflects the new location and/or time period.

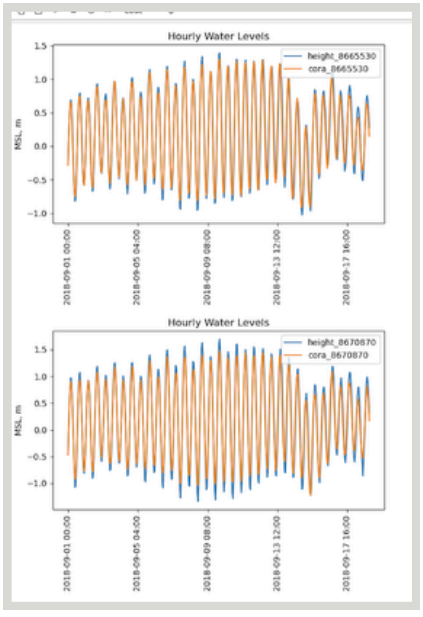
Explore More with CORA Notebooks

That's it! Now that you have learned how to access CORA data, you can explore additional notebooks that provide different ways to work with it. These include:

- **Visualizing Water Levels** – Creates a 2D surface plot to show water levels across the CORA model area
- **Native Grid** – Shows the model's grid (or mesh) and overlays elevation data.
- **Convert Datums** – Lets you upload a .csv file with CORA water level data and convert it from Mean Sea Level (MSL) to other datums using NOAA's **Tidal Analysis Datum Calculator**
- **Compare Time Series** – Pulls observed water level data from NOAA tide stations and compares it to the corresponding CORA data
- **Bounding Box** – Select data by geographic bounding box and map scatter plots through time by coordinate or node ID







All of these notebooks follow the same setup steps described in this guide. Once your environment is ready, you can open and run any of them to explore different CORA features and outputs.

Bulk Downloads

This section will walk you through the process for **bulk downloads** of CORA datasets. Due to the size and resolution of CORA datasets, bulk download requires sizable data storage and time to access. For this reason, this option is ***not recommended for most users*** but may be useful for those needing large data files. Once you have finished this section, you will have access to the **full time series** for modeled data, gridded, and derived products.

Available Data

CORA offers two main types of modeled data that can be downloaded in bulk:

- **Maximum Water Level** – Yearly optimized data, including water surface elevation, wave direction, wave height, and peak wave period.
- **Grid Structure** – Assimilated 500-meter grids covering 1979–2022. These files provide information about the underlying model grid and water surface elevation on specific dates.

Each file is provided in NetCDF (.nc) format and follows a naming convention that indicates the contents:

- **Maximum Water Level** files include:
  - **fort.63_YEAR.nc –** Water surface elevation (ADCIRC)
  - **swan_DIR.63_YEAR.nc** – Wave direction
  - **swan_HS.63_YEAR.nc** – Wave height
  - **swan_TPS.63_YEAR.nc** – Peak wave period
- **Grid Structure files** will look something like: 500m_grid_zeta_19790218.nc. This filename indicates:
  - **500m** – 500-meter grid spacing
  - **grid** – Gridded data structure
  - **zeta** – Water surface elevation
  - **19790218** – Date in YYYYMMDD format (e.g., February 18, 1979)

Bulk Downloads *(continued...)*

You can browse these datasets directly through the Amazon Web Services (AWS) S3 Explorer by clicking **Go to the Data** under your selected dataset on the **CORA landing page**. However, for bulk downloads, or to download individual files, we recommend the following options:

- **Amazon Web Services Command Line Interface,** which is code-based
- **Globus**, which offers a click-to-download option

Once downloaded, you can process and analyze these files using NetCDF-compatible tools.

Downloading With Amazon Web Services Command Line Interface

The Amazon Web Services Command Line Interface (AWS CLI) is a unified tool to manage your AWS services. With just one tool to download and configure, you can control multiple AWS services from the command line and automate them through scripts.

To download all of the fort.63 files from the AWS CLI, follow these steps:

1. Go to the AWS Command Line Interface (CLI) webpage: https://aws.amazon.com/cli/
2. Follow the instructions to download and install the CLI on your computer.
3. Once installed, open the CLI and enter the following command to download the full fort.63 dataset:

```
aws s3 cp s3://noaa-nos-cora-
pds/V1.1/assimilated/native_grid/ . --recursive --
exclude "*" --include "fort.63*.nc"
```

There are about 5 terabytes of fort.63 files. Since you are downloading all of the fort.63 files in the bucket, **this download will take days to complete**, so plan accordingly.

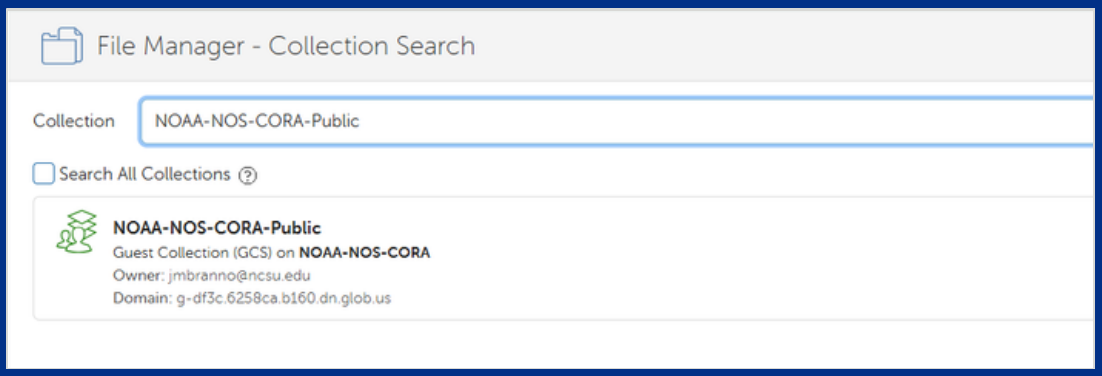
Bulk Downloads *(continued...)*

Downloading With Globus

Globus is a non-profit service where you can **move, share, and discover** data no matter where it lives. Access and manage all your data, even protected data, from anywhere with just a web browser.

To download CORA files using Globus, follow these steps:

1. Sign up for a Globus account: https://www.globus.org/get-started
2. Once logged in, search for **"NOAA-NOS-CORA-Public"** under “Collection” to find and connect with CORA datasets hosted on the NODD. Open the collection from the search results.



3. On the file list, open the **“V 1.1”** folder.
4. Open the **"assimilated"** folder to access the most current datasets.

5. From here, you can download individual folders one-by-one. **These downloads can take days to complete, so plan accordingly.**

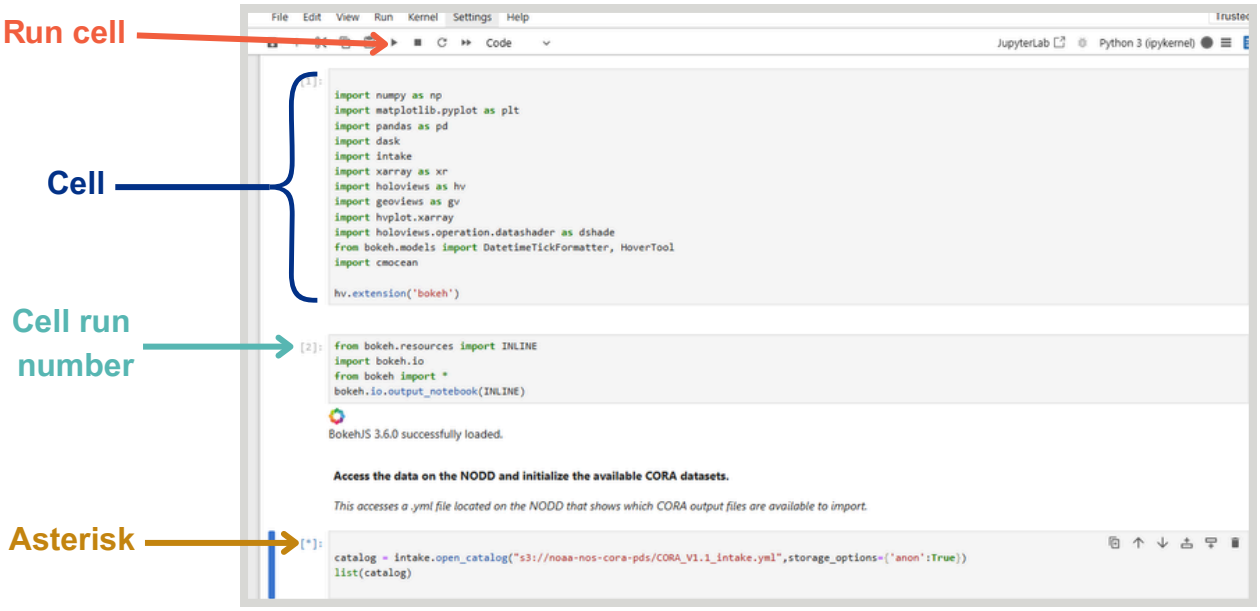
Appendix I: Coding Tips for Beginners

Running a Notebook

What is a "cell" in a notebook?

Notebooks are made up of cells - these are the gray boxes you see throughout the page. Each cell contains a chunk of code or text that can be run on its own.

- To "**Run**" a cell, click inside it and press **Ctrl + Enter** or click the "▶" button at the top of the screen.
- Run each cell in order from top to bottom.
- While a cell is running, you'll see an asterisk (*) next to it. This means the code is still processing. *Sometimes the cell runs so quickly that the asterisk disappears almost instantly.*
- Once the cell has finished running, the asterisk turns into a number (like [3]) showing the order it was run. This can be helpful for keeping track of what cells have been executed.
- The notebook will be completed once all the cells have been run.



Warning vs. Errors

- **Warnings:** Not all **red messages** mean something is broken. If you see **"FutureWarning"** or **"UserWarning,"** don't worry - these are just tips or notes about future updates. If the notebook keeps going, it's probably just a warning and can be ignored.
- **Errors:** True errors **stop the notebook** and usually include the word "**Error**" or "**Traceback**" in red text.

When running this first cell, **you may encounter an error** related to the configuration of the environment. This sometimes happens when a package is updated and loses support for other packages. If an error occurs while running this cell, refer to the **Issues** in the GitHub repository.

- If this is a **known issue** (make sure to also look at the "Closed" Issues), we are likely working to fix the problem or have developed a fix for the problem
- If the **issue is not listed**, we encourage you to open a new issue so the error can be fixed

Coding Language

- **cd -** Stands for "change directory." It tells the system to go into the folder you specified.
- **Cell -** Notebooks are made up of cells - these are the gray boxes you see throughout the page. Each cell contains a chunk of code or text that can be run on its own.
- **Dask -** A tool that helps Python handle large datasets more efficiently. Instead of loading an entire file into memory all at once (which can slow things down or even freeze your computer), Dask splits the data into smaller chunks, loading only what's needed when it's needed.
- **Environment -** A coding environment is a self-contained workspace with the specific tools your notebook needs. Setting up an environment will help you avoid errors by making sure everything is compatible in your notebook and ready to go.

Coding Language

- **Kernel -** The engine that runs your code inside a notebook. It takes the instructions you write and executes them, returning results. Each Jupyter kernel is linked to an environment, which is where all the necessary libraries and dependencies are stored. When we set up our environment, we also created a kernel that allows our notebook to use that environment. Jupyter supports different kernels for languages like Python, R, and Julia. Think of your environment as the set of instructions and the kernel as the worker that follows those instructions to run your code.
- **NOAA Open Data Dissemination (NODD) -** A platform that provides public access to NOAA's vast environmental data on commercial cloud platforms. Through partnerships with industry leaders, NODD expands access to high-quality data while reducing costs and risks associated with federal data services.
- **Xarray -** A tool that makes it easier to work with multi-dimensional data (like time, latitude, longitude, etc.) by labeling each dimension, so you can access, manipulate, and analyze the data more easily.

For definitions of key CORA terms and concepts, see the **CORA Glossary**.

Appendix II: Continuing to Use the Notebooks

Next time you want to use the CORA notebooks, **you don't need to repeat all the setup steps**, your environment is already configured.

Skip all the steps in Preparing to Run the Notebook and, instead, follow these instructions:

1. Open **Anaconda Prompt**
2. In the Prompt, you'll see something like this:

```
(base) C:\Users\User.Name>
```

Copy and paste the pathway to your CORA notebook and press **Enter**.
Our example pathway is:

```
cd C:\Users\User.Name\Desktop\CORA Notebooks
```

3. When the prompt updates to:

```
(base) C:\Users\User.Name\Desktop\CORA Notebooks>
```

Type:

```
conda activate cora
```

Press Enter.

4. You will now see something like:

```
(cora) C:\Users\User.Name\Desktop\CORA Notebooks>
```

Next to this **type:**

```
jupyter notebook
```

Jump to Navigating the GitHub-Hosted Jupyter Notebook (page 12) to continue working with the data.