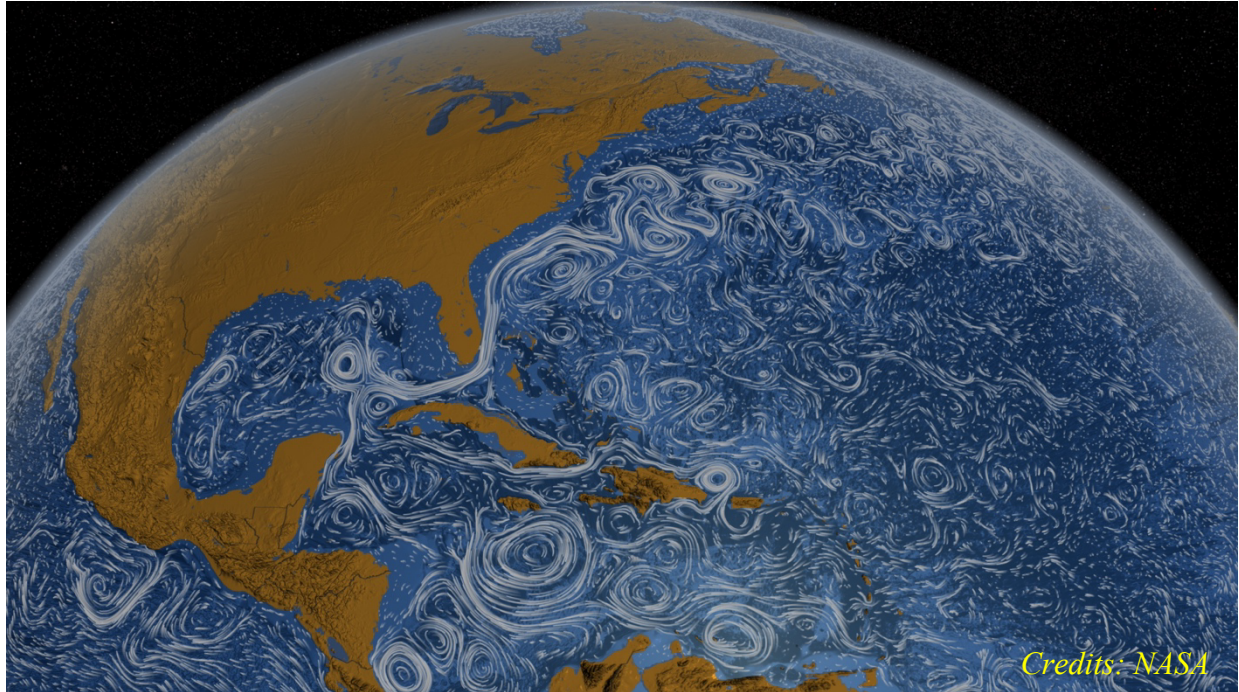


Getting Started



Installing *Miniconda*

During this course, you will frequently need to write code in *Python*. Many of its capabilities are derived from external packages like *numpy*, *matplotlib*, *xarray*, etc. To ensure these packages work well together, you need a package manager like Anaconda's *Miniconda*. We advise *Miniconda* because it does not require a lot of disk space like *Anaconda*, which has the same functionalities but comes with many pre-installed packages. If you already have a working version of *Anaconda*, you do not need to install *Miniconda*.

To install *Miniconda*, follow the instructions on <https://conda.io/docs/user-guide/install/>. The last step is to perform the `conda list` command in your terminal (Mac/Linux) or Anaconda Prompt (Windows) as a test. From here on we assume this step was successful.

Creating a new environment

An environment contains all the packages that a piece of software will use. Since packages get updated regularly, you can create multiple environments with the same packages having different versions. Creating a new environment is also a nice way to start from scratch, such that existing packages do not cause any conflicts. First, we must configure a new solver for conda. Execute the following command in your terminal (Mac) or Anaconda prompt (Windows):

```
conda activate base
```

```
conda install -n base conda-libmamba-solver
```

```
conda config --set solver libmamba
```

Now to create a new environment, enter:

```
conda create -n dyoc -c conda-forge cartopy cmocean geopy leafmap parcels jupyter
```

This creates an environment called ‘dyoc’ with *Python* version 3.11. If this does not work on an existing version of *conda*, you might have to update your *conda* first with `conda update conda`.

To see all your environments, enter

```
conda env list
```

The environment with an asterisk (*) is your current environment. This is generally the *base* environment, which is the default of *conda*. Therefore, at the start of each session you have to activate the *dyoc* environment:

```
conda activate dyoc
```

You will notice that the name before your cursor now changes to *dyoc*. If you would like to return to your *base* environment at any point, you can use the `conda deactivate` command.

For further information on *conda*, you can check out

<https://conda.io/projects/conda/en/latest/user-guide/index.html>.

Downloading *Python* packages

Sometimes you get an `ImportError` when trying to import packages. Often this is because these packages have not yet been installed. You can add packages to your new environment with

```
conda install -n dyoc [packagename]
```

or multiple packages at once using

```
conda install -n dyoc [name1] [name2] [name3]
```

Working in *Jupyter lab*

Instead of writing separate *Python* scripts for different questions, you can make a document of multiple ‘cells’ of executable code that you can run separately. Such a document is called a *notebook* and has the file extension *.ipynb*. To start a new *Jupyter lab* session from the Anaconda prompt or terminal, enter:

```
conda activate dyoc          # remember to activate this before each session
```


```
cd /path/to/mydirectory/    # optional, cd = change current directory
```

```
jupyter lab
```

This will open a tab in your internet browser. Supported browsers are Chrome, Safari and Firefox. Note that you can only access files in *Jupyter lab* that are in *mydirectory*, or any subdirectory thereof. Generally, you start in a convenient location and do not need to change the current directory.

If *Jupyter lab* is opened properly, you can see a file browser on the left. Now in the file browser, go to your desired folder. Then, click the Notebook – Python 3 (ipykernel) to create a new *notebook* or select an existing *notebook* to open it.

You now have opened an empty *notebook*. Modify the file name by right-clicking on the name tab > **rename** Notebook. Below the file name is a tool bar, make sure the toggle menu for the selected cell says ‘Code’. Now hit enter or click on the cell to make sure you can edit the cell (*edit mode*), You should see a blinking cursor now. Now type some *Python* code, e.g. `print("hello world!")`, and hit **shift + enter** to execute. As you can see, the line of code produces some output and a new cell is inserted below. If you want to delete the new empty cell, hit **esc** to enter *command mode* and double-press **d**.

Variables declared in one cell can be used in any other cell (above and below), but only after you have executed the code. This is very useful but may also lead to mistakes, for example when you accidentally set the value of variables to the wrong values because you run cells in the wrong order. If this happens, you can restart your *notebook* by double-pressing **0** (zero) in *command mode* or by clicking the  symbol in the tool bar, clearing all your variables.

If your code is stuck in a loop, press **I** twice in *command mode*. Alternatively you can click the stop button in the tool bar. To create/delete cells, enter *command mode* and move with the arrow keys to your desired cells. Now press **a** (**b**) to create a cell above (below) or press **dd** to delete. You can also select 'Markdown' in the toggle menu to create a cell with text only.

The provided information in this document is rather limited. For a helpful introductory video on *Jupyter lab*, you can watch <https://youtu.be/A5YyoCKxEQU> (~6 min), or for more information, check out the [user guide](#).