

## An introduction to **Conda**

- **Conda**<sup>1</sup> is an open-source package manager for Mac, Windows, and Linux. It can manage packages for Python, R, C, FORTRAN etc. but we will focus on **Python**
  - You can use **miniconda** or **anaconda** but **miniconda** is a smaller download so we will use that
  - Every time you need a new package you don't have you will individually download these with **miniconda** so **you only use the bandwidth and space that you need**
  - You can use the code editor of your choice but we would suggest **Spyder** or **IDLE** (**we strongly suggest Spyder**) because they are smaller downloads. PyCharm is very nice but is a larger download. If you are on a Mac or Linux computer you can also use Vim in the terminal but this requires you to learn a set of keystrokes to edit - although if you do, you can edit code very quickly!
- 

## How to install **miniconda**<sup>2</sup>

- Go to: <https://docs.conda.io/en/latest/miniconda.html> to find the correct miniconda Installers
- **Requirements** for your computer:
  - **Windows** 8 or newer (if older, let me know and we can use <https://repo.anaconda.com/miniconda/>)
  - **MacOS** 10.13 or newer
  - **Linux** (Ubuntu, RedHat, CentOS 7+ etc \*\*please let me know if you are using Linux!)
  - *For everyone: 400mb of free space for the download*
- **Download** the correct installer for your operating system from the “**Latest Miniconda Installer Links**” list<sup>3</sup>
  - For **Windows** - check if you need 32 or 64 bit (in Control Panel open System and check System type)
  - For **Mac** - if you have a very new M1 chip computer use the appropriate installer (downloading the pkg installer is easiest if you can)
  - For **Linux** - for most systems the Miniconda3 Linux 64-bit installer is correct (typing `uname -a` in the terminal will tell you what Linux architecture you are using)
- **Install**
  - **Windows**<sup>4</sup>: double click the **.exe** file you downloaded and follow the instructions

---

<sup>1</sup> <https://docs.conda.io/projects/conda/en/latest/>

<sup>2</sup> <https://docs.conda.io/en/latest/miniconda.html>

<sup>3</sup> <https://docs.conda.io/en/latest/miniconda.html>

<sup>4</sup> <https://conda.io/projects/conda/en/latest/user-guide/install/windows.html>

- Agree to prompts - install for single user “Just Me”
  - Agree to destination folder unless you want to change it (make a note of this)
  - “Register Miniconda3 as my default Python” - **Agree** to this unless you already have a Python setup that you want to keep
  - “Add Miniconda3 to my PATH” - **Do not select this option** unless you have a specific reason to do this (it is not recommended)
  - You don’t need to register for any of the recommended services after the installation is done
  - Once installed open **Anaconda Prompt** from the **Start** menu
  - Test: in the Anaconda Prompt window type:
    - `conda list`
    - You should see a list of packages if it is properly installed
- 
- **Mac<sup>5</sup>**: if you downloaded a *.pkg* file double click this and follow the instructions
  - If you downloaded a *.sh* file find the file in the terminal (probably in your Downloads folder) and type:
    - `bash Miniconda3-latest-MacOSX-x86_64.sh`
    - Use the name of the *.sh* file you downloaded, this name is an example!
    - When installing: [yes] to agreement, [yes] to directory unless you want to change it (make a note if you do), [yes] to init. If you already have other python installed and you want to keep using that then you can use the option to not activate it on startup with the instructions at the end of the install.
    - Close and then re-open your terminal window
  - Test: in the terminal type:
    - `conda list`
    - You should see a list of packages if it is properly installed
- 
- **Linux<sup>6</sup>**: Find the *.sh* file you downloaded (probably in your Downloads folder) and in the terminal window type:
    - `bash Miniconda3-latest-Linux-x86_64.sh`
    - Use the name of the *.sh* file you downloaded, this name is an example!
    - When installing: [yes] to agreement, [yes] to directory unless you want to change it (make a note if you do), [yes] to init. If you already have other python installed and you want to keep using that then you can use the option to not activate it on startup with the instructions at the end of the install.
    - Close and then re-open your terminal window
  - Test: in the terminal type:

---

<sup>5</sup> <https://conda.io/projects/conda/en/latest/user-guide/install/macos.html>

<sup>6</sup> <https://conda.io/projects/conda/en/latest/user-guide/install/linux.html>

- `conda list`
  - You should see a list of packages if it is properly installed
  - **For all systems:** figure out where your miniconda is located - this will be useful later!
    - I used the automatic location chosen by my installer, so for me it is located:
      - `/Users/ellendyer/miniconda3/`
- 

## Making a conda **environment**<sup>7</sup>

- **Conda lets you create multiple environments.** In each environment you can install a unique set of packages that work together and are made for specific tasks. Some people will only need one environment, other people may need different versions and packages that don't all work together so it is useful to have multiple environments. If you want to experiment it is a good idea to create a new conda environment so that you don't break your working environment and have to rebuild it!
- Here is an example of how to **create an environment** called *myenv* (this is the conda website example name!):
  - In **Anaconda Prompt** (Windows), or in the **terminal** (Mac, Linux) type:
    - `conda create --name myenv`
    - (if you want to change the environment name do it here)
    - Conda will ask you if you want to proceed, type: `y`
    - You have created an environment! To test this type: `conda env list` (you should see your environment name listed)
- To work in an environment and install packages there you need to **activate** it:
  - In **Anaconda Prompt** (Windows), or in the **terminal** (Mac, Linux) type:
    - `conda activate myenv`
    - (use whatever environment name you chose!)
- When you create a new environment conda installs the version of python that came with the miniconda download - you don't need to install python again unless you want to create an environment with a specific version of python
- Once you are in your activated conda environment you can install packages and run code:
  - Here is a simple example for how to install code, type:
    - `conda install scipy`
  - To check if you have installed the package scipy type:
    - `conda list`

---

<sup>7</sup> <https://docs.conda.io/projects/conda/en/latest/user-guide/tasks/manage-environments.html>

- (you should see scipy listed with its version number and other packages and dependencies)
- Here is a more complicated example to install **xarray**<sup>8</sup> (we will use this package in other examples) that installs 4 packages at the same time (xarray, dask, netCDF4, and bottleneck):
  - `conda install -c conda-forge xarray dask netCDF4 bottleneck`
- Now if you type:
  - `conda list`
- you should see many more packages including xarray!
- `-c conda-forge` in the xarray install means you are installing xarray from the conda-forge **channel**
- Most package websites will tell you how to install their packages in conda and if you need to use a channel. For example, the pandas<sup>9</sup> package instruction page looks like this:

The final step required is to install pandas. This can be done with the following command:

```
conda install pandas
```

and for the xclim<sup>10</sup> package the instruction page looks like this:

```
$ conda install -c conda-forge xclim
$ conda install -c conda-forge clisops # for subsetting and bias correction functions
```

- Sometimes you might want to install a specific version. If you know the version number you can install the specific version as follows:
  - `conda install scipy=0.15.0`
- If you don't know where to get a package from a good trick is to do an internet/google search: `conda install packagename`
  - Example: I searched for: `conda install metpy` and the first hit was <https://anaconda.org/conda-forge/MetPy> which suggests a few ways to install and tells you where to find the documentation for that package (always have a look at the documentation!) and it looks like the screenshot below:

<sup>8</sup> <https://xarray.pydata.org/en/stable/getting-started-guide/installing.html>

<sup>9</sup> [https://pandas.pydata.org/docs/getting\\_started/install.html](https://pandas.pydata.org/docs/getting_started/install.html)

<sup>10</sup> <https://xclim.readthedocs.io/en/stable/installation.html>

MetPy is a collection of tools in Python for reading, visualizing and performing calculations with weather data.

Conda

Files

Labels







Badges

 License: [BSD 3-Clause](#)  
 Home: <https://github.com/Unidata/MetPy>  
 Development: <https://github.com/Unidata/MetPy>  
 Documentation: <https://unidata.github.io/MetPy>  
 368938 total downloads  
 Last upload: 1 month and 8 days ago

## Installers

Info: This package contains files in non-standard labels.

### conda install ?

 linux-64 v0.11.1  
 win-32 v0.7.0  
  noarch v1.2.0  
 osx-64 v0.11.1  
 win-64 v0.11.1

To install this package with conda run one of the following:

```
conda install -c conda-forge metpy
conda install -c conda-forge/label/cf201901 metpy
conda install -c conda-forge/label/cf202003 metpy
conda install -c conda-forge/label/metpy_rc metpy
```

- *Always select the first option to install with conda forge - copy and paste this command into your Anaconda prompt window or terminal to install.*
- *It is usually ok to install most packages from -c conda-forge*
- Once you have installed all the packages you think you need you are ready to work in your chosen editor. You can keep installing packages in your environment as you need them over time.
- Once you are done installing packages or writing and running code you should **deactivate** your environment by typing:
  - `conda deactivate`

**Exercise:** let's create an environment for our test code and workshop called 'wksenv'

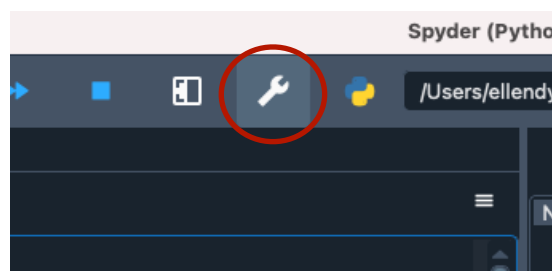
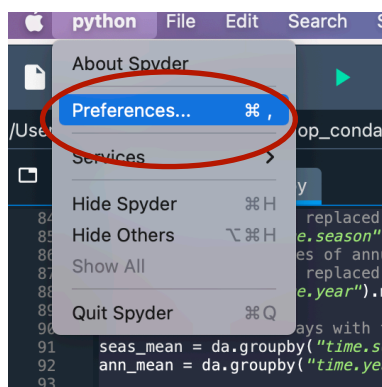
We will do the installs in a few steps because sometimes there are clashes if we try to install them all at once

- `conda create --name wksenv`
- `conda activate wksenv`
- `conda install -c conda-forge xarray dask netCDF4 bottleneck`
- `conda install -c conda-forge scipy cartopy`
- `conda install -c conda-forge spyder-kernels=2.3.3` (\*\*this is only necessary if you are using Spyder as an editor)

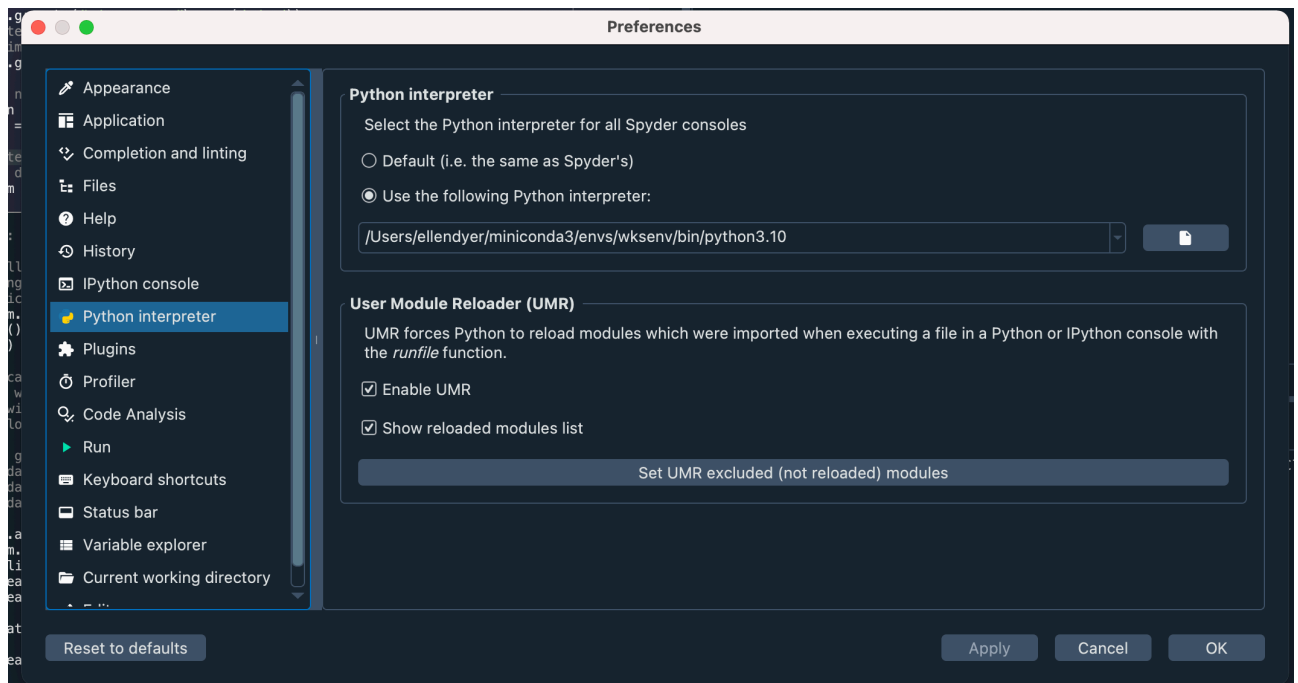
---

## Editors

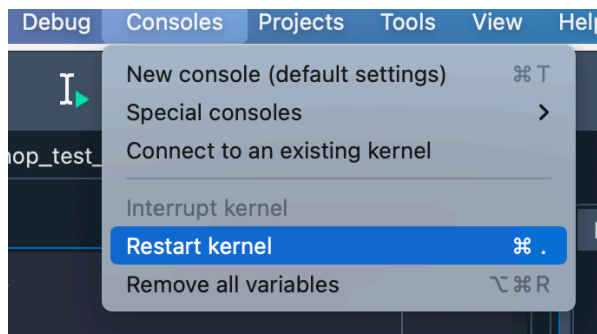
- **IDLE:** comes with miniconda and you can find it on your computer as follows:
  - To open within an environment to use that environment's packages, activate that environment in Anaconda Prompt or the terminal and type:
    - `idle3`
  - The environment needs to have some packages installed like a specific python version (or any other package at all) to be able to access IDLE. This won't be an issue for anyone other than in the testing phase because you always need to install at least one package!
  - \*you always need to open IDLE like this in the environment whose packages you want to use
- **Spyder:** this editor looks a lot like Matlab but it needs to be downloaded
  - Download using an installer: <https://docs.spyder-ide.org/current/installation.html#downloading-and-installing> (choose Windows or Mac installer - click, download and follow instructions)
  - If you want to just install Spyder using miniconda, create and activate an environment (give it a name related to spyder because you will always open spyder from this environment!)
  - Here is an example of how to do this:
    - `conda create --name spyder_env`
    - `conda activate spyder_env`
    - `conda install -c conda-forge spyder-kernels=2.3.3 spyder=5.3.3`
  - Open by typing:
    - `spyder`
  - If you want to open a python file you have already made you can type:
    - `spyder *filepath/filename*`
  - No matter how you install spyder once you open it you need to go to the "Preferences" menu and you can find this in **two places** depending on your operating system as shown below:



and change the “Python Interpreter” to be your chosen conda environment (for our workshop this will be `/Users/ellendyer/miniconda3/envs/wksen/bin/python` (Mac, Linux example) OR `C:\Users\ellendyer\Miniconda3\envs\wksen\python` (Windows example) **\*\*use the correct path for your computer!**



Then go to the “Consoles” menu and select Restart kernel



- **VIM:** this is a natural choice for Mac or Linux because they have easy to work with terminals and vim is already installed
- **Mac, Linux:** open the terminal and activate your conda environment
  - To edit python code with vim type:
    - `vi code_name.py`
  - To run python code with vim type:
    - `python code_name.py`
- **Windows:** you need to download the vim GUI <https://www.vim.org/download.php>
- If anyone wants to download **PyCharm** as an editor please send me an email!

---

## Editing and running code

Let's try to create a test script called **workshop\_test.py** to read in some CHIRPS rainfall data over Ethiopia (you need a file called **workshop\_chirps.nc** which we will give you). We will also give you a copy of **workshop\_test.py** but for now let's build up the file together in the workshop. Create an empty python file called: **workshop\_test\_live.py**. Open the file in your chosen editor.

**#Step 1:** Start with import statements (this tells python which packages

#you will actually use in your code and names

#them, so we will call xarray : xr)

import xarray as xr

from matplotlib import pyplot as plt

import cartopy.crs as ccrs

import cartopy.feature

###

**#Step 2:** Read in our data from a netcdf file of rainfall over Ethiopia

#(edit the path to the file correctly depending on where you saved it)

#Now we have a data array called da and

#we can print it to see what we have:

da = xr.open\_dataarray("/Users/ellendyer/Desktop/workshop\_chirps.nc")

# print data array

print(da)

# print attributes in file metadata

print(da.attrs)

# print a list of array dimensions

print(da.dims)

# print a list of array coordinates

print(da.coords)

# print time and lon dim labels

print(da.time)

print(da.lon)

###

**#Step 3 :** Let's make a quick plot of one month of data.

#To do this we will select one date using the .sel function

#and using the coordinate time and its labels. In xarray you can

#use date strings to select time steps (use the coordinate format

#that we saw when printing da.time above). If you instead wanted to

#choose the first time element you would use .isel which lets you select

#using normal array indices, so the first month would be .isel(time=0).

#use coordinate names

da.sel(time='2018-07-01').plot()



```
plt.show()
plt.clf()
```

```
###
```

**#Step 4 :** Let's select and modify data. First we will make a timeseries by taking an average over the dimensions lat and lon. Remember to use the dimension names that we found in printing da.dims above. We can then select a smaller time range for this time series by using .sel again but this time by taking a slice. If you are selecting a single point you either need to know the exact coordinates or you need to ask xarray to select the 'nearest' point.

```
#make a timeseries plot by taking a regional average
ts = da.mean(dim=('lat','lon'))
print(ts)
ts.plot()
plt.show()
plt.clf()
```

```
#select a time slice from ts timeseries (you can use date strings
#with xarray)the .sel slice method chooses the nearest point so
#you don't need to know exact bounds
ts = ts.sel(time=slice('2015-01-01','2017-12-31'))
ts.plot()
plt.show()
plt.clf()
```

```
#if you use .sel without slice then you need to say
#that you want the nearest point
print(da.sel(lat=3.333,lon=40.111,method='nearest'))
```

```
###
```

**#Step 5 :** Xarray has use some of the useful grouping functions like #groupby (makes bins), rolling (can do rolling averages), #resample (can change time frequencies) and more!

```
#groupby just makes bins but doesn't alter your data array
print(da.groupby("time.season"))
```

```
#if you want to alter it you need to act on it and here we do
#this by taking a time mean with .mean('time')
#calculate long term seasonal means
#(now 'time' has been replaced by 'season')
print(da.groupby("time.season").mean('time'))
#calculate a timeseries of annual averages
#(now 'time' has been replaced by 'year')
print(da.groupby("time.year").mean('time'))
```

```
# create new data arrays with these time means
```

```
seas_mean = da.groupby("time.season").mean('time')
ann_mean = da.groupby("time.year").mean('time')

#calculate long term seasonal anomalies from the long term annual mean
#you can do simple matrix math to data arrays just like you do with numpy
seas_anom = seas_mean - ann_mean.mean('year')
```

```
###
```

**#Step 6 :** plot the seasonal anomaly maps

```
#This will plot all four xarray defined seasons (we can talk about
#selecting different seasons another time!) and plot them with the
#automatic xarray plotting function
seas_anom.plot(col="season")
plt.show()
plt.clf()
```

```
#But we can make nicer plots with geography (using cartopy)
#Here we will just plot for JJA by using .sel and the new coordinate
#season with the label 'JJA'
#We can look at more ways to plot and the details below later on
ax = plt.axes(projection=ccrs.PlateCarree())
seas_anom.sel(season='JJA').plot(ax=ax,transform=ccrs.PlateCarree())
ax.coastlines()
ax.add_feature(cartopy.feature.BORDERS)
ax.add_feature(cartopy.feature.RIVERS)
# for extent the order is [West,East,South,North]
ax.set_extent([33, 47, 3.5, 14.5])
plt.show()
plt.clf()
```

```
###
```

**#Step 7 :** write a data array back to a netcdf file

```
# write seas_anom to netCDF to save your work
#for later or share your calculations
#It's always a good idea to use a defined path!
seas_anom.to_netcdf("/Users/ellendyer/Desktop/workshop_out_sa.nc")
```

---

**Libraries that will be used in future code (you can conda install them now):**

geopandas

rioxarray

---

## Useful links:

### Conda:

- Download the conda cheat sheet (it is very useful!): <https://docs.conda.io/projects/conda/en/latest/user-guide/cheatsheet.html>
- Take a look at this list of common conda issues and help to solve them: <https://docs.conda.io/projects/conda/en/latest/user-guide/troubleshooting.html>
- You can install CDO and NCVIEW with conda (<https://anaconda.org/conda-forge/ncview> , <https://anaconda.org/conda-forge/cdo>) (these are very useful for previewing and altering netcdf files, especially large netcdf files)

### Xarray

- Xarray user guide <https://docs.xarray.dev/en/stable/>
- Xarray can work with csv and excel files but Pandas might be a better tool if this is your main file type and you aren't making maps with data <https://pandas.pydata.org>
- A very good intro tutorial to using xarray with climate data - what our workshop has been loosely based on <https://xarray-contrib.github.io/xarray-tutorial/oceanhackweek-2020/xarray-oceanhackweek20.html#>

### Mapping and climate libraries

- Here is a link to the cartopy documentation [https://scitools.org.uk/cartopy/docs/latest/getting\\_started/index.html](https://scitools.org.uk/cartopy/docs/latest/getting_started/index.html) (have a look at the gallery to see what is possible)
- Has datasets and coding tutorials: [https://docs.digitalearthafrika.org/en/latest/sandbox/notebooks/Beginners\\_guide/07\\_Intro\\_to\\_xarray.html](https://docs.digitalearthafrika.org/en/latest/sandbox/notebooks/Beginners_guide/07_Intro_to_xarray.html)
  - Digital Earth Africa online courses: <https://learn.digitalearthafrika.org/dashboard>
- PyCLIM examples: <https://climate.usu.edu/people/yoshi/pyclm101/index.html>
- A free short course: <https://www.ceh.ac.uk/training/climate-data-analysis-python>
- Climate index library: <https://xclim.readthedocs.io/en/stable/index.html>

---

### Other code for workshop:

- workshop\_test.py
- workshop\_shape.py
- workshop\_station\_shape.py
- TSV\_reader.py
- workshop\_correls.py
- workshop\_curves.py