Setting up a virtual environment using Anaconda

Contents

[What is a virtual environment? 2](#_Toc97114527)

[Why do use virtual environments? 3](#_Toc97114528)

[Let’s create our first virtual environment (Windows) 4](#_Toc97114529)

[Instructions & Walkthrough 4](#_Toc97114530)

[Instructions Only 9](#_Toc97114531)

[Let’s create our first virtual environment (Mac) 10](#_Toc97114532)

[Instructions & Walkthrough 10](#_Toc97114533)

[Instructions Only 16](#_Toc97114534)

[Using our Virtual Environment in VS Code 17](#_Toc97114535)

**Pre-requisites**: Make sure that you have downloaded Anaconda (& VS Code). If you have not downloaded Anaconda, please see refer to document called “Download and install anaconda.docx”. If you have not installed VS code, please refer to the document “How to download and install VS Code” via the following link:

<https://github.com/spider-z3r0/ULpsych_programing_club/tree/main/preparation>

# What is a virtual environment?

A virtual environment is a discrete, isolated, digital space created by our computer. We can set up a virtual environment for the different projects that we are working on, whether they are data science projects, web development projects, software engineering projects, etc.

There are two key parts to a virtual environment:

1. A specific version of python we are using to run the code.
2. A set of 3rd party python libraries that we want to use in our code.

i) Python is a continuously developing language, meaning there are consistent updates to Python that add functionality. These changes range from minor to significant. When we set up a virtual environment in Anaconda, we specify what specific version of Python that we are using. The specific version that we use will be dependent on the type of project we are working on (so for the purposes of these workshops, there is no need to worry about it).

ii) In our Jupyter Notebook file, *week3.ipynb*, we discussed the idea of modules. Modules are python files containing code written by someone else. We can use modules to carry out specific operations in Python without needing to rewrite that code ourselves. We do this by importing the modules (e.g., *import* **pathlibm,** *import* **webbrowser,** *import* **datetime**).

But where do those modules come from? Last week, those modules (*pathlib*, *webbrowser*, *datetime*) were imported from the Python standard library. The Python standard library contains built in modules, meaning that once you have downloaded Python itself, then you will direct access to those modules. No extra steps needed.

However, not all modules are part of the standard library. There are also libraries, containing sets of modules with certain functionalities, that exist outside of the standard library. For example, *pandas* is a Python library used to conduct data analysis. Libraries such as pandas, meaning they exist outside of the standard library, are called 3rd party libraries. And the great thing about Python is that we can import modules from libraries like pandas as well. They just need to be installed beforehand. So whenever we create a virtual environment, we tell Anaconda what libraries we need, and it will install it for us.

**TL;DR** – Virtual environments are digital spaces that allow us to separate our projects. When we set up a virtual environment, we tell Anaconda what version of Python we want, and what python libraries we want to run in that space.

# Why do use virtual environments?

There are two major reasons to use virtual environments.

**The first reason** is because virtual environments make it easy to separate our projects. This is important, because for any given project, we might need to run a specific version of python and/or a specific version of whatever modules we are importing from third-party libraries.

To demonstrate the importance of this, let me give you an example. Let’s say that you have a relatively old Macbook that uses a regular charger rather than the lighting charger. If the government stormed in tomorrow and seized all old Macbook chargers in the state and replaced them with the new lighting chargers, then you wouldn’t be able to use your Macbook for long. That’s because your Macbook has certain requirements (like having the right charger, dufus[[1]](#footnote-1)) that need to be met for it to work. These requirements, otherwise known as *dependencies*, are not met, then it is back to writing on stone tablets for you my friend.

Similarly, each of our python projects will have certain dependencies – a specific version of either python and/or specific versions of 3rd party library – that need to be met. If we did not use a virtual environment, anytime we updated our python or 3rd party libraries, there is a non-trivial chance our past projects won’t work anymore (leaving us to experience the panic of having code that does not work with no idea how to quickly fix it).

Virtual environments save us from that panic, by enabling us to keep specific versions of python/libraries that our projects depend on.

**The second reason** is that virtual environments are efficient in the long run. You can (and we will) set up a virtual environment that will contain the necessary libraries for conducting data cleaning, processing, and visualisation. So anytime you need to run have some data you want to run analysis on, you can use your that the trusty virtual environment we set up. There is no need to download the same libraries again and again when you want to run the same type of tasks.

Okay that’s enough background information. Let’s make own virtual environment.

**TL; DR**: Virtual environments enable us to set up distinct project spaces designed to use certain versions of python/libraries to carry out specific (and often repetitive) tasks in a simple and logical manner, removing the urge to pummel our computers into dust.

# Let’s create our first virtual environment (Windows)

## Instructions & Walkthrough

This part will give both tell you what you need to do and a walkthrough of what is going on when we are creating our virtual environment. In case you are already familiar with working with command prompts, or are returning to this document at a later point and just want the instructions, then feel free to skip to “instructions only”.

The first thing we are going to do is open the Anaconda Prompt. The prompt is a black screen that enables us to type in the commands needed to create a virtual environment. To open it, type “Anaconda Prompt” into the search bar and click the relevant option.

Graphical user interface, application

Description automatically generated

Doing so will open up the Anaconda prompt, as you can see, it is not the prettiest user interface you will ever encounter.

Shape

Description automatically generated

We can use the Anaconda prompt to now command our computer to do our bidding. The first thing we are going to do is check the version of anaconda that we have installed. To do this type the following into the prompt (after (base) C:\>):

(base) C:\> *conda -V*

This basically asks our computer to check what version (-V) our Anaconda (conda) that we are working with. And as dutiful as ever, our computer will return conda version that you are currently working with. I recently updated mine, so my version will probably differ from yours.

Shape

Description automatically generated with low confidence

To update our Anaconda, we can use the following command:

(base) C:\>conda update conda

This may take a few seconds to run. If you anaconda version is not up to date, it will ask you whether you want to update it, by pressing either [y] or [n]. Press y on your keyboard to update Anaconda. If your anaconda is up to date, then you will see the following.

Text

Description automatically generated

Do not worry about what is going on below our command. The main thing is now we are up to date with Anaconda.

Now let’s create a virtual environment. To do this, type the following into the Anaconda prompt:

(base) C:\>***conda create -n ourenv python=3.10***

Let me explain what is going on here. The first part, ***conda***, is an instruction to working with anaconda. The second part, ***create -***n, tells our computer that we are creating a new (-n) virtual environment. The third part, ***ourenv***, is the name of our virtual environment. And ***python=3.10[[2]](#footnote-2)***, tells anaconda what version of python we will be working with. To basic syntax for creating virtual environments will follow this pattern:

conda create-n ENVIRONMENT\_NAME python=PYTHON VERSION WE WANT

Once you press enter on that command, a whole load of text will appear.

Text

Description automatically generated

This basically tells us where the virtual environment will be set up (**environment location**), the version of python that will be added (**added / updated specs**), and any packages that will be downloaded and installed in our virtual environment. All we really care about, thought, is the last line at the bottom: “**Proceed ([y]/n)?**”

Press y and then press enter.

Proceed ([y]/n)? y

The packages will start downloading to our virtual environment. Once our virtual environment is ready, we will be asked whether we want to activate or deactivate our environment.

Graphical user interface, text

Description automatically generated

To activate our environment, following the instructions provided by the prompt and then press enter;

(base) C:\>conda activate ourenv

Once you do that, you will see that (base) will be replaced by the name of our environment. Congratulations you have successfully created your first virtual environment!

Now we are going to install some third-party packages into our environment. The third party packages we are going to install are pandas, a library filled with modules tailored towards data analysis, and iPython, which is short for Interactive Python, which contains modules that enables us to easily work with Jupyter Notebooks.

To install these third-party libraries, there are two options:

Option 1: Install them together

(ourenv) C:\>pip install ipython pandas

Option 2: Install each library individually

(ourenv) C:\>pip install ipython

Text

Description automatically generated

(ourenv) C:\>pip install pandas

Text

Description automatically generated

Once those two libraries are installed, we have everything we need (for now at least) within our virtual environment. Now we need to select this environment as our kernel in VS Code (see relevant section at the end of the document).

You can now close the Anaconda Prompt.

## Instructions Only

Open the anaconda prompt and type in the following:

(base) C:\> *conda -V*

*(base) C:\>conda update conda*

*(base) C:\>conda create -n ourenv python=3.10*

*Proceed ([y]/n)? y*

*(base) C:\>conda activate ourenv*

*(ourenv) C:\>pip install ipython pandas*

# Let’s create our first virtual environment (Mac)

## Instructions & Walkthrough

This part will give both tell you what you need to do and a walkthrough of what is going on when we are creating our virtual environment. In case you are already familiar with working with command prompts, or are returning to this document at a later point and just want the instructions, then feel free to skip to “instructions only”.

The first thing we are going to do is open our terminal. The terminal is a white screen that enables us to type in the commands needed to create a virtual environment. To open it, type “terminal” into the finder and click the relevant option.

Graphical user interface, application, Word

Description automatically generated

Doing so will open up the terminal prompt, as you can see, it is not the prettiest user interface you will ever encounter.

Graphical user interface, text, application

Description automatically generated

We can use the terminal to now command our computer to do our bidding. The first thing we are going to do is check the version of anaconda that we have installed. To do this type the following into the prompt (after the $ symbol[[3]](#footnote-3)):

$ conda -V

This basically asks our computer to check what version (-V) our Anaconda (conda) that we are working with. And as dutiful as ever, our computer will return conda version that you are currently working with. I recently updated mine, so my version will probably differ from yours.

Graphical user interface

Description automatically generated with medium confidence

To update our Anaconda, we can use the following command:

$ conda update conda

This may take a few seconds to run. If you anaconda version is not up to date, it will ask you whether you want to update it, by pressing either [y] or [n]. Press y on your keyboard and then press enter to update Anaconda.

Table

Description automatically generated

A picture containing table

Description automatically generated

We can then recheck our conda version to make sure it is up to date.

If your anaconda is already up to date, then you will see the following.

Text

Description automatically generated

Do not worry about what is going on below our command. The main thing is now we are up to date with Anaconda.

Now let’s create a virtual environment. To do this, type the following into the Terminal:

$***conda create -n ourenv python=3.10***

Let me explain what is going on here. The first part, ***conda***, is an instruction to working with anaconda. The second part, ***create -***n, tells our computer that we are creating a new (-n) virtual environment. The third part, ***ourenv***, is the name of our virtual environment. And ***python=3.10[[4]](#footnote-4)***, tells anaconda what version of python we will be working with. To basic syntax for creating virtual environments will follow this pattern:

conda create-n ENVIRONMENT\_NAME python=PYTHON VERSION WE WANT

Once you press enter on that command, a whole load of text will appear.

Table

Description automatically generated

This basically tells us where the virtual environment will be set up (**environment location**), the version of python that will be added (**added / updated specs**), and any packages that will be downloaded and installed in our virtual environment. All we really care about, thought, is the last line at the bottom: “**Proceed ([y]/n)?**”

Press y and then press enter.

Proceed ([y]/n)? y

The packages will start downloading to our virtual environment. Once our virtual environment is ready, we will be asked whether we want to activate or deactivate our environment.

Table

Description automatically generated with medium confidence

To activate our environment, following the instructions provided by the prompt and then press enter;

$conda activate ourenv

Once you do that, you will see that (base) will be replaced by the name of our environment. Congratulations you have successfully created your first virtual environment!

Now we are going to install some third-party packages into our environment. The third party packages we are going to install are pandas, a library filled with modules tailored towards data analysis, and iPython, which is short for Interactive Python, which contains modules that enables us to easily work with Jupyter Notebooks.

To install these third-party libraries, enter the following command:

$pip install pandas ipython

Table

Description automatically generated with medium confidence

Text

Description automatically generated

Once those two libraries are installed, we have everything we need (for now at least) within our virtual environment. Now we need to select this environment as our kernel in VS Code (see relevant section at the end of the document).

You can now close the terminal.

## Instructions Only

Open the terminal and type in the following:

$*conda -V*

$*conda update conda*

$*conda create -n ourenv python=3.10*

*Proceed ([y]/n)? y*

$*conda activate ourenv*

$*pip install ipython pandas*

Close terminal.

# Using our Virtual Environment in VS Code

To select our virtual environment, ***ourenv***, in VS code. Click where it says “Select Kernel” on the top right-hand corner (alternatively, you might already have a different kernel working there, if so, click that). From the drop-down menu, select our virtual environment.

Graphical user interface, text

Description automatically generated

Text

Description automatically generated

***NOTE****: This part can be finnicky. If you do not see your virtual environment pop up, then save your files and close VS code. Reopen the VS and the relevant file, click select kernel (or whatever python version is running) and it should appear. You might need to click it a few times. If it is still not there, then give one of us a shout.*

Once it is working, then congratulations, you have set up a virtual environment that is ready to start working!

1. Note to self: look up how to spell dufus later, otherwise I will look like an idiot. Also make sure to delete this note to self. [↑](#footnote-ref-1)
2. Make sure there isn’t a gap between “python”, “=”, and “3.10”, otherwise the command will not work. [↑](#footnote-ref-2)
3. What will appear before your commands will be different than mine. The main thing is what comes after the $. [↑](#footnote-ref-3)
4. Make sure there isn’t a gap between “python”, “=”, and “3.10”, otherwise the command will not work. [↑](#footnote-ref-4)