


Setting up environments

Global and Virtual environments

Now that we have installed Python, we mentioned that it comes with certain pre-installed basic libraries. But what happens when you want to use Python for a specific task and need to install additional packages? For this, Python has enabled `pip`, a recursive acronym for "Pip Installs Packages" or "Pip Installs Python", as its package manager to automate installation, update, and package removal.

However, installing new packages directly into the download of Python can have difficult consequences. This Python we just installed is the **Global environment**. We mentioned that Python is an open-source interpreted programming language that goes through constant updates. For this reason, newer libraries developed on different Python versions often conflict with libraries without the same updates, and error messages start to pop up.

badPythonEnv [https://github.com/Dmitry Zub, Dec 22, 2021. Python Virtual Environments tutorial using Virtualenv and Poetry. Place of publication: SerpApi. Available \[here\]\(#\).](https://github.com/Dmitry Zub, Dec 22, 2021. Python Virtual Environments tutorial using Virtualenv and Poetry. Place of publication: SerpApi. Available here.)

This image is a beautiful, chaotic example of what happens to your global environment when we do not have order in the different paths and versions of new packages.

To avoid this, a good practice when using Python is to use **virtual environments**. This environment allows for isolating package dependencies so they do not clash. For example, you may have two projects, one for computer vision and another for Natural Language Processing (NLP). Both of them use similar libraries but in different versions of Python. We can not install both versions system-wide, but we can create isolated environments for each project.

NOTE:

These environments are called containers because they do not interact with each other. However, this is only for the system. The folders and files in your storage are all available for all the environments. This means that if you remove, add, or change a folder when you are working inside an environment, that change is permanent and will apply to all environments

There are different ways to build Python environments:

IMPORTANT:

Please note that all of the steps mentioned on this page are recommended from original sources; try to follow them as faithfully as possible. If, in any step, something does not work

as it should, contact a technician first before following any other instructions that need you to move things directly from your terminal.

Python environments (`venv`):

For this type of environment, the only requirement for your computer is to have a version of Python installed on it. `venv` is a Python module that supports lightweight virtual environments.

For this type of environment, you need to be familiar with how the terminal works, how you can move from one folder to another, and the Python versions you have installed. If you are unfamiliar with these requirements, please refer to the How to use Anaconda section.

From Python 3.3 onwards, `venv` should be included in the commands available. To create a virtual environment with this, please open your terminal:

NOTE:

If you have not yet installed Python on your computer, please refer to the Installing Python section of the wiki.

macOS

To enter the terminal, you can search it directly from the Launchpad or application folder. Type `Command` + `Space bar` and type terminal for a shortcut. First, we must ensure you are in the folder where you want to save the environment. When you open the terminal, you should see only your user name:

Python terminal

For this example, I am going to access my Documents folder. You can access whatever folder you wish to save your environment on.

WHY IS THIS IMPORTANT?

If you save a virtual environment with the same name in the same folder, the terminal will interpret it as you want to rewrite it, and you will lose the information from the previous one. Before creating new environments, make sure that the name and folder you choose differ from previous ones.

```
python -m venv [name of the environment]
```

Inside of the brackets, you can change it to whatever name you want. Just make sure that the name of the environment is something easy to remember, or write it down somewhere. The name should also follow the terminal rules: if you are going to name something with more than one word,

you need to hyphenate the words with an underscore (_).

- Example: `python -m venv example_environment`.

The way you activate it is while inside the folder where you created the environment, call `source [name of the environment]/bin/activate`. The name in front of the dollar sign should change to the name of the environment you are currently in.

- Example:

Captura de Pantalla 2024-02-13 a la(s) 9 29 28 a m

WINDOWS OS

For the Windows OS you also need to have previously installed Python.

IMPORTANT:

If you installed Python by downloading the installer directly from the Python page, you might need to add the path to the environment variables of your computer. Please see the section on how to do that in the "Add path to environment" section. If you are not familiar with the path and what it means to add it to the environment variables, please continue with the next steps.

If you have Python already in your Path variables, you can just use the same arguments as the macOS instructions.

- Example: `python -m venv example_environment`

Please note that the same rules apply to Windows, so make sure to name the environment something unique and easy to remember, and also select the correct folder for your environment.

Conda environments

Another way to create environments in Python is to use the Anaconda distribution. For this, we have two options: we can download Anaconda from the official distribution or a more light version of Anaconda called Miniconda.

WHICH VERSION DO I NEED?

These two versions are from the same distribution and are widely used for data science and scientific computing. The main difference between the two is the size of the installation. Anaconda requires at least 3 GB of free disk space, while Miniconda only requires 400 MB (Something you need to take into consideration if you do not have enough space available on your computer). Anaconda comes with a large array of pre-installed packages and a very user-friendly graphical interface that can favor those who are not very familiar with the use

of the terminal or command line prompts. Miniconda only includes the ``conda`` function and Python in its installation.

If you go for the Anaconda version:

macOS

If you go to the official Anaconda page (<https://www.anaconda.com/download#downloads>), you will see the Download button for your OS. If you are using macOS, it is also important to know if your hardware settings are an Intel chip or an M1/M2/M3 chip.

Screenshot 2024-02-12 at 11 00 56

You can check this by going to the Apple at the top left corner of your screen and clicking on About this Mac.

Screenshot 2024-02-12 at 11 04 31

This should prompt a Window that will say the chip your computer has:



Go back to the Downloads page on Anaconda and choose the right setting for your computer.

NOTE:

You can also click on `More info` to see all the different settings on your computer.

Once the package is done loading, you can choose the predetermined installation and click install. If your terminal was opened, please relaunch it, you will now be able to see the pre fix in your user name as `(base)`. Which means that you are in the base or root environment. We explain this later on.

It goes from:

image not found or type unknown

To look something like this:

image not found or type unknown

For Windows users:

For Windows, there is only one universal installer, so it does not matter the version of Windows that you are using. It is only important to know that it is only available for the 64-bit version. If your computer is a 32-bit, please look into the Anaconda archive (<https://docs.anaconda.com/free/anaconda/install/old-os/>).

image not found or type unknown

Follow the installation process. Same as with the Python installation, this installer is going to ask you if you want to add this software to your path variables. If you are not sure or do not understand fully what path does, do not activate this function. We can change it later on if you need it.

NOTE:

If you do not want to add Conda to your path variables, you can use the command prompt directly from the Anaconda Navigator. Otherwise, all the other Windows terminals will not have the `conda` prompt.

image not found or type unknown

Installing Miniconda

NOTE:

Please be mindful that Miniconda has no user interface and only offers the `conda` prompt and Python. It also requires general knowledge of how the terminal works.

Graphical installer:

macOS

You must download the Miniconda installer from the official page (<https://docs.anaconda.com/free/miniconda/>). From here, you can choose the Intel or the M1/2/3 chip version and the bash or pkg versions. We strongly recommend using the bash version since the pkg may skip the "Destination select" process, failing the installation.

bash

- Once you download the document, locate where it is stored (it is most likely in the Downloads folder).
- Go to a new terminal in your computer (`Command` + `Space bar` type terminal + `Enter`)
- Go into the folder where you have the bash file stored (Ej. `cd Downloads`); you can verify it is there by running `ls`.
- This is an optional step, but we recommend verifying the download to ensure everything will run correctly. In your terminal run `shasum -a 256 filename` replacing `filename` with the downloaded file name. If it does not raise any error, follow the next step.
- Run `bash <conda-installer-name>-latest-MacOSX-x86_64.sh`. Replace the `<conda-installer-name>` with the downloaded file's name.
- Once the process is finished, please close and reopen your terminal (`Command` + `q` to close/reboot the program completely. You can see if this was successful by clicking `Command` + `tab`. It should show you all the applications you have opened. Make sure that the terminal does not show there).
- To verify the installation, open a terminal window and run `conda list`. If you installed everything correctly, a list of all the installed packages should appear.

Windows

For Windows, you can just download the `.exe` file and run it. In case you want to install it directly from the command line, you can run the following code:

NOTE:

If your computer is not 64-bit, you can look into miniconda archive installs. Just keep in mind that older versions are not compatible with newer versions of Python, and will limit the amount of new libraries you can use.

```
curl https://repo.anaconda.com/miniconda/Miniconda3-latest-Windows-x86_64.exe -o miniconda.exe
```

The curl command allows the terminal to exchange data directly from another server, in this case, it is making an HTTP request, just like a web browser. If the request fails, make sure that your internet connection is stable. Once the download is complete, type:

```
start /wait "" miniconda.exe /S
```

This will start the installation process. Afterward, you can run `del miniconda.exe` to delete the executable file.

After installing, close the terminal and open the "**Anaconda Prompt (miniconda3)**" to use miniconda.

Revision #3

Created 14 February 2024 14:35:07 by Mayra Cristina Berrones Reyes

Updated 27 August 2024 13:07:24 by Mayra Cristina Berrones Reyes