Python for Spatial Data Analysis

Module GG3209 - Second Part

Dr. Fernando Benitez-Paez

2023-02-23

Table of contents

1 Getting started **3**

2 Content **4**

3 Advisement **5**

4 Assessment **6**

5 University Staff **7**

6 Our Research **8**

7 Introduction **9**
- 7.1 Slides . . . 9
- 7.2 Installing and setting up your Python environment . . . 9
  - 7.2.1 Virtual Python Environments: . . . 9
- 7.3 Python Package Manager – Mini-Conda : . . . 11
- 7.4 Installing a python package manager – Mamba . . . 14

8 Working with tabular Data **15**

9 Working with Spatial Data **16**

10 Unsupervised statistical learning – Clustering **17**

11 Auxiliary Data **18**

References **19**

1 Getting started

Welcome to the second part of the module of **GG3209 Spatial Analysis with GIS**. This part will take advantage of the initial part, which provided you with a solid understanding of spatial data formats (vector&raster) and use them to perform multiple types of analysis like the so-called Multi-Criteria Evaluation (MCE) using the widely popular Open-Source GIS tool, QGIS.

Now in this second part, you will be guided to install, handle and use another powerful tool in the geospatial field, Python. It is a free and open-sourced scripting language that was commonly used to automate tasks in the GIS world. Nowadays is one of the most popular programming languages, especially for GIScience. It is widely used in the private, public sectors and academia for cutting-edge research, where scripts, front-end and back-end components are created using this language. Python is also widley popular as a easy-to-code programming language to deploy new methods, share knowledge, list and fetch data, and run spatial analysis through multiple scientific fields.

In fact, most companies, or institutes where you probably want to apply once you finish your degree, will be happily interested in your development skills using Python and will validate your current ork in platforms like **GitHub**, where you can share and disseminate your project. At this stage whether you are student of Geography or Sustainable Development, you probably are familiar with R and its powerful capacity for spatial statistics. Now this module aims to introduce you Python which is mainly used for scalable and robust spatial analysis, front-end applications and process vast among of data. Every day, more packages and code-repositories are shared and maintained for easy use and installation, allowing developers or analysts from all backgrounds and expertise use and integrate them into their own code.

This part of the module and the lab (technical practice) are meant to be an introduction to Python and some of the spatial libraries. Like any other new language, you need to learn the basic rules (syntax) to write your own scripts, and soon with practice, you will become a python developer.

2 Content

The content included in this module stands to be a brief introduction to Python where you get familiar with multiple concepts up to now new for most of you, we will cover the basis of the programming logic, Python, Version Control (essential for reproducibility and open science) and some of the main libraries for geospatial analysis.

There are many other concepts and interesting exercises we could apply to learn and see the potential of Python in handling spatial data. However we have designed this part in the way you can see the difference between using GIS tools (user-interface based) and creating code to run script routines. As any other new language the best way to master it is through constant practice. **So don't get stress out** if you find out this module difficult or different, you will slowly get better and efficient creating new scripts.

> **Note**
>
> If you think this is the line of work you would like to pursue, you can learn more about in the module GG4257 - Urban Analytics: A Toolkit for Sustainable Urban Development where we have more time to properly described other libraries, more practical exercises, and use Python for more advance analysis.

The module is structured in the following lectures including the correspondent practical Lab.

- Introduction to Python, Jupyter Notebooks, and GIT
- Working with tabular data (NumPy, Pandas)
- Working with spatial data in Python (GeoPandas, Rasterio)
- Clustering analysis

3 Advisement

> **Tip**
>
> Do not be afraid of failure or getting errors, even during the installation process; it has happened to all of us, regardless of the level of expertise or number of projects created. **In programming, failure is part of the process**; The key is to find the basis of any issue and understand how code, logic, and syntax work in harmony to get the results you are expecting.

> **Important**
>
> All the work described here can be executed in the computer labs, ad we highly recommend you use that environment for all the practical exercises included in this book. However we have also integrated instructions for you to install a specific list of components to get an essential but scalable environment that allows you to write, clone, debug and execute code for this course.

4 Assessment

This part of the module is assessed by 100% coursework

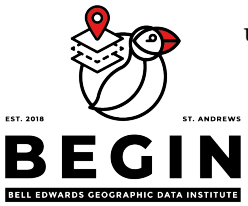
5 University Staff

Module Co-ordinators: Dr Urška Demšar or Dr Fernando Benitez-Paez

Office hours: By appointment online and live during the labs

Lab assistants: Dr Charlotte van der Lijn , Ali Moayedi, Benjamin Ong, Georg Kodl

6 Our Research



If you want to know more about why spatial data holds the key to unlocking a deeper understanding of our planet and its intricate systems. Let's BEGIN a spatial and data-driven conversation and be part of our multidisciplinary group in St Andrews.
https://begin.wp.st-andrews.ac.uk/

7 Introduction

In this first week we will cover the introduction to Python, why is it important, the software components required to have a simple python environment to work with this programming language, relevant and basic concepts as well the description of the workflow of version control. In this week we have lof of work and concepts to work on, but luckily this will stablish all the concepts and components we need for the rest of the course.

7.1 Slides

Click here to open the slides in a separate tab Week 1 - Slides

7.2 Installing and setting up your Python environment

7.2.1 Virtual Python Environments:

Virtual environment in python is a programming environment which works in a way that the Python interpreter, libraries, and scripts installed into it are isolated from the ones installed in other virtual environments (or used by the operating system, important in macOS). This ensures that all the installed packages **work nicely together**. You can create multiple environments on your computer for different projects (having e.g., different versions of Python and specific libraries), and you can swap easily between environments by activating them from the command prompt with a single command. There will be instructions to doing that in the following sections.

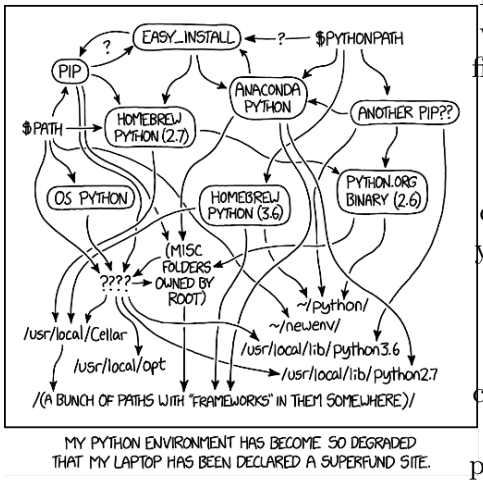


Figure 7.1: Source: https://xkcd.com/1987/

Perhaps the first exercise that helps you get a sense of what is like of working with Python is setting up your first **Python Environment - PyEnv**. A PyEnv is an independent space where you install all the packages, python version and software components you would need to write and run your scripts. Although is not a compulsory activity, it is recommended to do it before you jump to write and deploy python scripts. It is also relevant that you feel comfortable working with the **Terminal** or **Command-Prompt** in your operating system, whether it is a Windows, Mac, or Linux computer. Technically you could manually install every single library we need. However, we will use a predefined Python environment file that includes all the packages and the Python interpreter we will need for this course. The reason for doing this, is because installing Python libraries and its dependencies can get very tricky and confusing, depending on the operating system, the version, the path where we install, the type of users, even the package manager that suppose to help us can actually be a source of confusion. So, the best way and certainly the most secure way is using a unified single environment file and a package manager to create our Python Environment - PyEnv.

Let's start with the package manager, we will use Conda, more specifically Mini-Conda (a lighter version of Conda), which is a package, dependency and environment management for any language including Python. It is an open-source component, and can be installed on Windows, macOS, and Linux. With MiniConda you could quickly installs, runs, and updates packages and their dependencies. Mini-Conda easily creates, saves, loads, and switches between environments on your local computer.

7.3 Python Package Manager – Mini-Conda :

> **!** Important
>
> For **MacOS users**, please be aware that macOS already includes an old version of Python, so it is needed to install an updated version of Python that works with the other packages and dependencies that we need for this course.

Got to: https://docs.conda.io/en/latest/miniconda.html and get the installer based on your operating system.

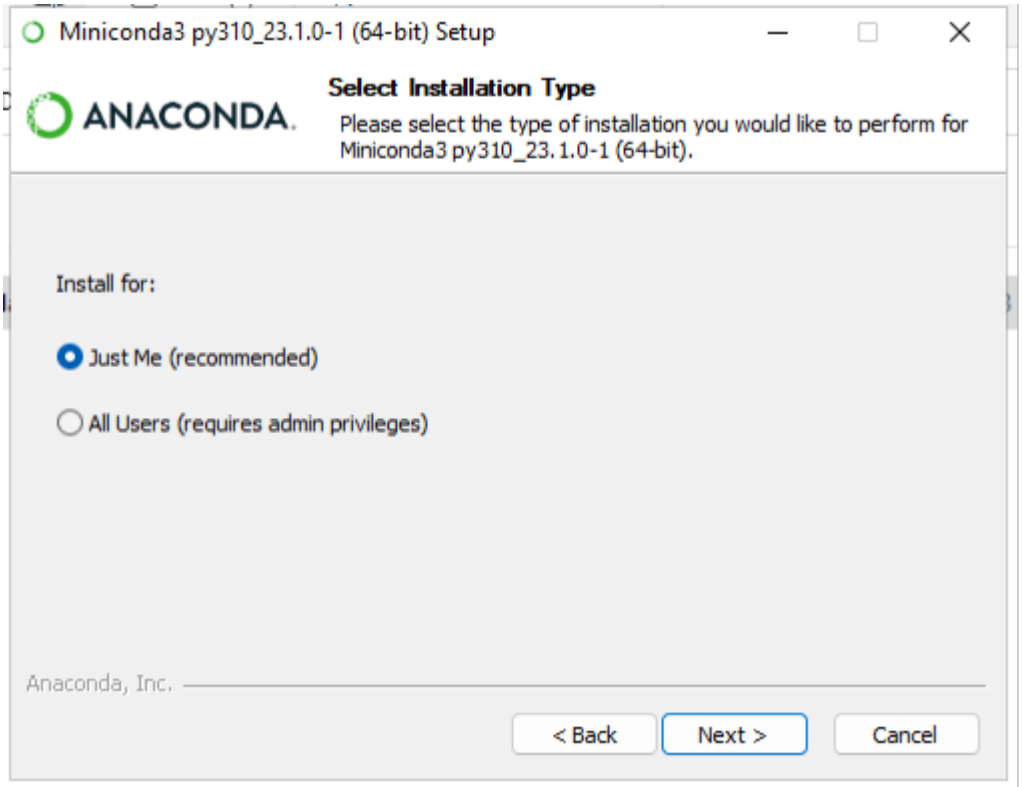
Latest Miniconda Installer Links

Latest - Conda 22.11.1 Python 3.10.8 released December 22, 2022

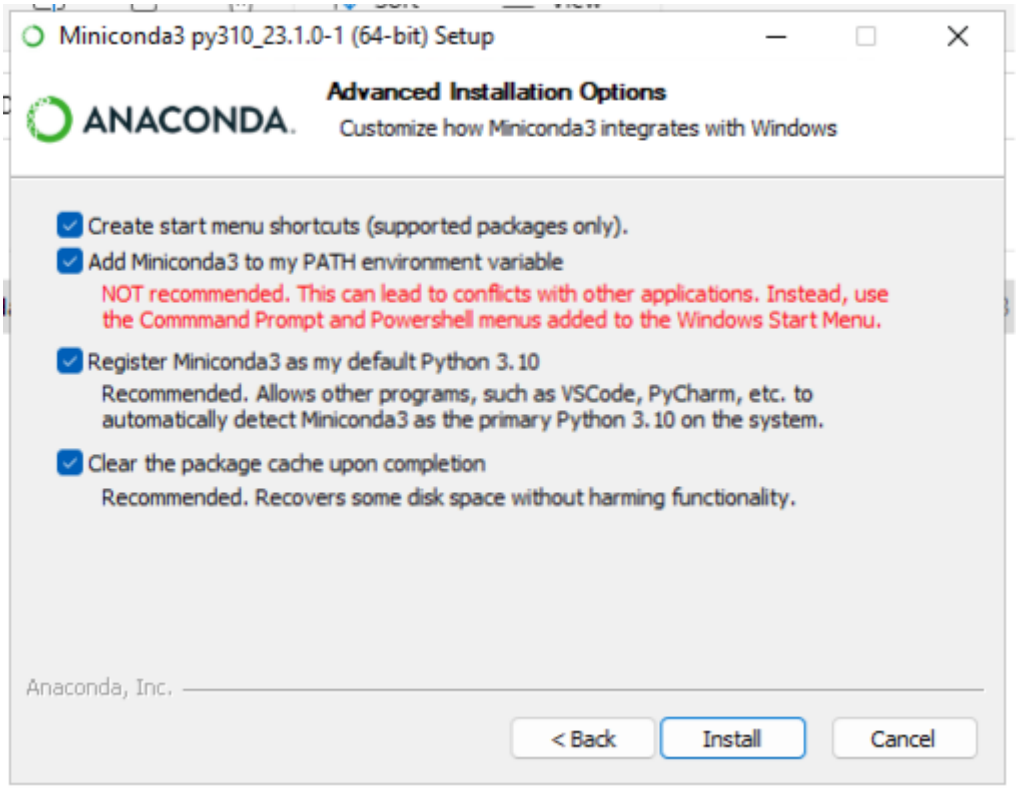
Platform	Name	SHA256 hash
Windows	Miniconda3 Windows 64-bit	2e3086630fa3fae7636432a954be530c88d0705fce497120d56e0f5d865b0d51
	Miniconda3 Windows 32-bit	4fb64e6c9c28b88beab16994bfba4829110ea3145baa60bda5344174ab65d462
macOS	Miniconda3 macOS Intel x86 64-bit bash	7406579393427eaf9bc0e094dcd3c66d1e1b93ee9db4e7686d0a72ea5d7c0ce5
	Miniconda3 macOS Intel x86 64-bit pkg	9195ffba1a6984c81c69649ce976a38455ace5b474c24a4363e5ca65fc72e832
	Miniconda3 macOS Apple M1 64-bit bash	22eec9b7d3add25ac3f9b60621d8f3d8df3e63d4aa0ae5eb846b558d7ba68333
	Miniconda3 macOS Apple M1 64-bit pkg	fbb33c5770b10a0d5a0deef746e7499bfaf8ff840d0d517175036dd8449357f6
Linux	Miniconda3 Linux 64-bit	00938c3534750a0e4069499baf8f4e6dc1c2e471c86a59caa0dd03f4a9269db6
	Miniconda3 Linux-aarch64 64-bit	48a96df9ff56f7421b6dd7f9f71d548023847ba918c3826059918c08326c2017
	Miniconda3 Linux-ppc64le 64-bit	4c86c3383bb27b44f7059336c3a46c34922df42824577b93eadecefbf7423836
	Miniconda3 Linux-s390x 64-bit	a150511e7fd19d07b770f278fb5dd2df4bc24a8f55f06d6274774f209a36c766

Once you have downloaded the installer, double click on the installer file to install it. In general, you could follow the default options, but for this course, make sure you pick the following:

1. Select “Just Me” during the installation, and MiniConda will only be available for the current user. This will not require any administrator rights for the installation.

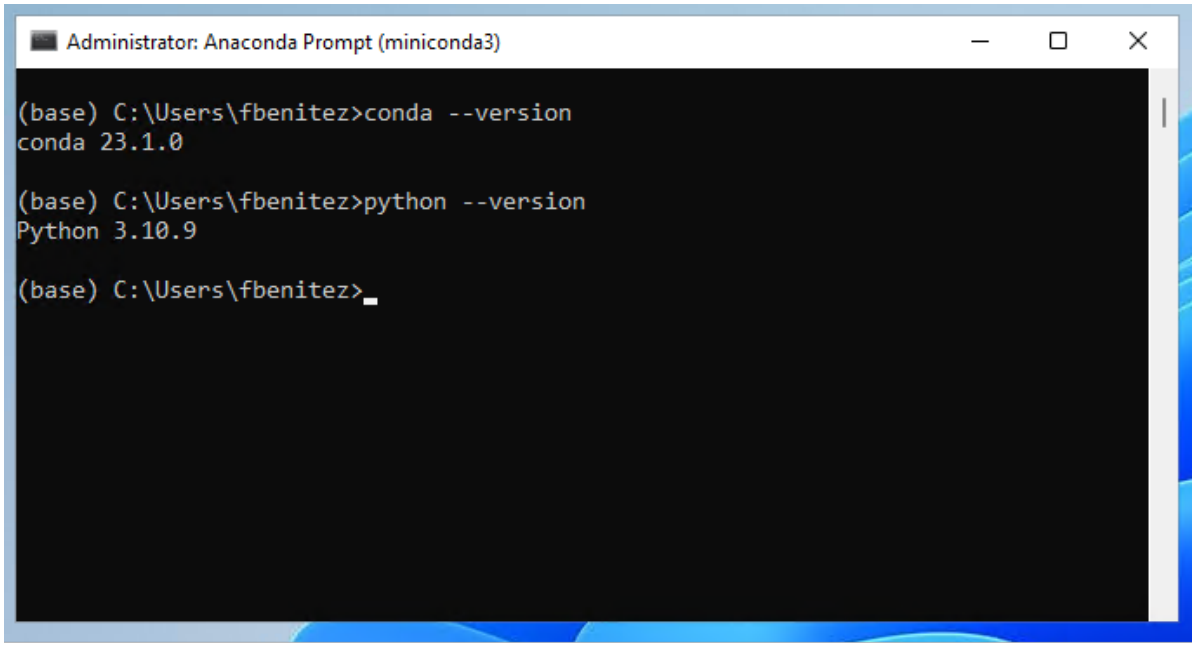


2. Make sure you pick all the advanced installation options.



3. After the installation is complete, you can validate that python and MiniConda were installed appropriately by running the following commands in your terminal or command prompt.

Open the **Anaconda Prompt (miniconda3)** or **Terminal** for macOS, from the Start menu (Apps) and run the command `conda --version`, it should return something like.



You might have noticed that I also ran the command `python --version` this one is to validate the python version you have in your system (e.g., 3.10.9)

If you have any problems with the **Miniconda installation**, you can find some installation tips on the Miniconda website.

7.4 Installing a python package manager – Mamba

Ok, now your computer has **MiniConda** installed; the next step is installing a python package manager. We will use the python package manager called Mamba to handle the installation of python packages in **Miniconda**. This is particularly important as mamba will help us ensure everything we install or remove is consistent. This can be very tedious work, so Mamba is a very convenient tool for the consistency of our virtual environments. Having a python package manager is not always a requirement when you set your virtual environment but will make your life easier when you start to create multiple environments.

1. Open a terminal window or command prompt in Windows (as an admin user) and run the following command:

   ```
   conda install mamba -n base -c conda-forge
   ```

2. If you get a message asking you to confirm the installation of new packages type yes and Enter

8 Working with tabular Data

This lecture

See Knuth (1984) for additional discussion of literate programming.

9 Working with Spatial Data

This is a book created from markdown and executable code.

See Knuth (1984) for additional discussion of literate programming.

10 Unsupervised statistical learning – Clustering

This is a book created from markdown and executable code.

See Knuth (1984) for additional discussion of literate programming.

11 Auxiliary Data

This is a book created from markdown and executable code.

See Knuth (1984) for additional discussion of literate programming.

References

Knuth, Donald E. 1984. “Literate Programming.” *Comput. J.* 27 (2): 97–111. https://doi.org/10.1093/comjnl/27.2.97.