Starters sheet Python Data Mining Practical Course 2019

(courtesy of module "Research Methods" 17/18)

Work with whatever you (and your team member) are comfortable with. If gedit/vim + terminal works, then that's great. However, IDEs can save you a lot of time, especially if you're working with a large amount of files. So consider taking the opportunity to familiarize yourself with one.

Spyder IDE: Installed on all computers. Simple interface, offers basic functionalities such as refactoring, python interpreter, point+click navigation, etc.

PyCharm IDE: Community edition can be downloaded for free (200MB), can be executed without installation (I suggest creating a link to your desktop/sidebar). Offers more advanced functionality. If you're used to programming in IDEs like Eclipse or Visual Studio, you should try PyCharm.

Basics

If you're proficient in another language but quickly want to get into the basics of python (or need a refresher), you should start with this 10-minute tutorial:

https://www.stavros.io/tutorials/python/

To open an interpreter, simply type "python" in your terminal. Once running, you can type in any command and it will execute – you don't need to create a file (like in Java/C++) to run commands. For actual programming, you should always create files, my personal recommendation is to always have an interpreter open so you can test whether some small command or algorithm works without changing your source code. Almost every python IDE does also come with an interpreter, so you don't have to work with the terminal. If you want to take more time to cover the basics properly, you can work through the beginning chapters of the free ebook http://www.diveintopython.net/

But don't worry – it's not necessary to invest a lot of time into reading books. *The best way to learn Python is to start coding!*

Advanced

Python is arguably the most popular language in the scientific computing community due to 1) its accessibility and 2) its large collection of diverse and community-maintained packages.
Some essential packages you will have to work with in almost all projects are *numpy*, *scipy*, and *matplotlib*. These provide Matlab-like functionality for efficient array calculations. In fact, numpy arrays have somewhat replaced Python's own built-in lists in a lot of use cases.

While you will learn most features once you start searching for them, it's useful to understand the absolute basics and the indexing syntax of numpy arrays so you can read other people's code.
The numpy website offers its own tutorial which you can go through in a short time.

A general crash course of all 3 packages and python can be found here:
http://cs231n.github.io/python-numpy-tutorial