Introduction to Anaconda and Python: Installation and setup

Damien Rolon-Mérette^a , Matt Ross^a , Thaddé Rolon-Mérette^a & Kinsey Church^a

^aUniversity of Ottawa

Abstract ■ Python has become one of the most popular programming languages for research in the past decade. Its free, open-source nature and vast online community are some of the reasons behind its success. Countless examples of increased research productivity due to Python can be found across a plethora of domains online, including data science, artificial intelligence and scientific research. This tutorial's goal is to help users get started with Python through the installation and setup of the Anaconda software. The goal is to set users on the path toward using the Python language by preparing them to write their first script. This tutorial is divided in the following fashion: a small introduction to Python, how to download the Anaconda software, the different content that comes with the installation, and a simple example related to implementing a Python script.

Keywords ■ Python, Psychology, Installation guide, Anaconda. **Tools** ■ Python, Anaconda.

drolo083@uottawa.ca

[10.20982/tqmp.16.5.S003](https://doi.org/10.20982/tqmp.16.5.S003)

Acting Editor ■
Nareg Berberian
(University of Ottawa)

Reviewers
■ Tarique Siragy (University of Ottawa)

Introduction

Why learn Python? Python is an object-oriented, interpreted, mid-level programming language that is easy to learn and use while being versatile enough to tackle a variety of tasks (Helmus & Collis, 2016). Its open-source nature has skyrocketed its popularity since its first appearance in 1991 (Van Rossum & Drake Jr, 1995), and is now considered among the top programming languages to learn (Saabith, Fareez, & Vinothraj, 2019).

It is free to use, has cross-platform compatibility (Mac, Windows, Linux, Ubuntu) and has low system requirements, giving any individual the opportunity to code in Python. It already has an immense community ranging from everyday individuals to top research scientists who have developed interesting projects related to data science, machine learning, artificial intelligence, app and game development, scientific research and more. These projects are easy to find thanks to the open-source community that continues to develop the language capabilities and can be searched by simply adding the key word “Python” to any inquiry. This community also provides a wealth of resources such as courses, source codes, solutions to com-

mon problems, video tutorials, and much more. These resources are often free to access and cover the most common problems developers run into at all levels of difficulty.

In addition to the beneficial features mentioned above, Python has many different qualities specifically tailored for the field of psychology (Dalmaijer, 2016). It can be employed at every step of the research procedure, from automation to analysis. For example, in the field of clinical psychology and clinical neuroscience, Python is being used to build sophisticated machine learning algorithms to interpret the onset of development, maintenance, and/or remission of psychopathology based on personal characteristics and sociocultural contexts (Coutanche & Hallion, 2019). In the field of neuroscience, NeuroRA (neural representational analysis; an easy-to-use Python toolbox for comprehensive representation analysis) is being used for conducting powerful cross-modal data analyses for various types of neural data (e.g. EEG, MEG, fNIRS, fMRI; Lu & Ku, 2020). In developmental psychology, Python's natural language processing capabilities are being used to automate and analyze interviews (Underwood, Kirchhoff, Warwick, & Gartstein, 2020). Finally, for data capturing, Python is being used to facilitate the creation of specific experimental

Figure 1 ■ Available Anaconda distribution installation packages across platforms and processor type.

Anaconda Installers		
Windows	MacOS	Linux
Python 3.7	Python 3.7	Python 3.7
64-Bit Graphical Installer (466 MB)	64-Bit Graphical Installer (442)	64-Bit (x86) Installer (522 MB)
32-Bit Graphical Installer (423 MB)	64-Bit Command Line Installer (430 MB)	64-Bit (Power8 and Power9) Installer (276 MB)
Python 2.7	Python 2.7	Python 2.7
64-Bit Graphical Installer (413 MB)	64-Bit Graphical Installer (637 MB)	64-Bit (x86) Installer (477 MB)
32-Bit Graphical Installer (356 MB)	64-Bit Command Line Installer (409 MB)	64-Bit (Power8 and Power9) Installer (295 MB)

paradigms and record participant's answers accordingly (PsychoPy; Knorr, Marxen, & Petzold, 2019). These are just a few examples of how Python can be used in psychology to speed up research at every step and how it can allow for more powerful analyses and research models.

So why learn Python? It can save you time, increase control over your experiments, allow your imagination to be the limit of your research designs and because it can optimize your research productivity. With such objectives in mind, this tutorial was designed to get you started as fast as possible by explaining how to install the most optimal **packages** with the Anaconda distribution tool kit. We cover how to use these packages, important terminology, how to employ various Python interfaces, and how to implement your first few lines of code, effectively building your first program and preparing you for the next few steps in learning Python.

Anaconda

What is Anaconda? Anaconda is a free software that provides you with a toolkit that is tailored for research and science. Installing Anaconda gives you access to different environments that allow you to code in either Python or R. These environments, also known as **integrated development environments (IDEs)**, are platforms or apps that greatly ease the development of code. They serve a similar role to text processors like *Microsoft Word*, *Google Doc*

and *Pages* for writing text, but in truth they are so much more. IDEs contain many useful features to write, edit and debug code, visualize and inspect data, store variables, present results, and collaborate on projects. While the display and the quirks of an IDE differ, the programming language does not. Thus, changing IDEs in Anaconda does not result in drastically changing your Python code. The learning curve is mostly related to understanding Python's syntax. Once you learn how to code in one IDE, you will be able to transfer this skill to another with ease. One IDE is not necessarily better than another, each comes with its own pros and cons and which one you choose comes down to personal preference.

Downloading the toolkit also provides you with an enormous selection of prebuilt functions that the Python community has coded in the past. These functions are regrouped in what are called **libraries** and can be downloaded easily through Anaconda. There are many different ways to install and use Python on your computer, but Anaconda is a simple, well-supported, **graphical user interface (GUI)** that includes the most important libraries and IDEs in its installation. Anaconda also simplifies the process of keeping all of these libraries up to date. Thus, rather than installing Python separately with different IDEs, libraries and functionalities, Anaconda can do it for you in one installation.

As mentioned on their website, Anaconda is “the eas-

iest way to perform Python/R data science and machine learning on a single machine. Developed for solo practitioners, it is the toolkit that equips you to work with thousands of open-source packages and libraries.” It is a great platform for beginners, as well as advanced programmers, to tackle different projects and is the best approach to getting started with Python.

Downloading and Installing Anaconda

The Anaconda website provides the necessary links to download the software for free in either Windows, Mac, or Linux. It also has starter videos, documentation, training, and support. To simplify this process, this tutorial has provided a step-by-step installation guide as well as a summary of the different options you may encounter.

To download and install Anaconda, you must first select your operating system, desired version of Python, and whether you want 32-Bit or 64-Bit Graphical Installer which depends on your system’s processor type (see Fig. 1) and click on the appropriate link from their website to begin the download: [Official Anaconda Website for Downloads](#).

It is important to notice that Python 3 is the latest version and is being constantly updated and supported. The previous version, Python 2 is still offered, but is no longer being maintained. Consequently, this tutorial will focus on Python 3.

Once the download has finished, open the Anaconda setup. If you are using a Windows operating system please follow Fig. 2 and if you are using a Mac operating system follow Fig. 3. For the Windows installation in the “Advanced installation options” (Fig. 2, panel e), we recommend not selecting either option for first time users. We encourage anyone interested in using these options to do their research to ensure they understand the pros and cons.

Once the installation is completed, you can search for the Anaconda Navigator on your device and open it. The layout of Anaconda should resemble Figure 4. Congratulations, you now have everything you need to start coding in Python (and R as well)!

Anaconda Navigator and IDEs

The Anaconda Navigator, shown in figure 4, is a GUI, like we previously mentioned. Think of this as the “menu” page of Anaconda, where you can easily access and launch different IDEs without needing to type commands in a terminal window. There are many different IDEs specially designed for Python programming and the following ones are automatically installed with Anaconda. To access an IDE, click on its icon in the Navigator Home tab.

Jupyter Notebook and JupyterLab

Jupyter Notebook is a web-based IDE that uses your default web browser. Each block of code can be run separately, making it highly flexible and easy to experiment with. This allows for different types of texts to be used in the same Notebook. Thus, code outputs, visualizations, equations, and normal text can be used all in one place. This makes it effective to create and share documents and to present your code and results in an organised and aesthetically pleasing way. Since it is web-based, it also makes it easy to share Notebooks with others and is optimal for collaborations. JupyterLab is an extension of the Jupyter Notebook with many more features. In JupyterLab, Jupyter Notebooks are integrated with other applications, like a command-line Terminal, a code **Console** and a **Text Editor**. As interesting as JupyterLab is, we recommend beginning with Jupyter Notebook as it is easier to use and contains the fundamental tools to learn Python programming and conduct data analysis. Jupyter Notebook will be the go-to IDE for this tutorial. For more information about Jupyter Notebook and JupyterLab, see the Jupyter web page at [jupyter.org](#)

Spyder

Spyder is a highly functional IDE specifically designed for data analysis. It can be seen as an all-in-one IDE, containing an editor to write code scripts, a variable explorer to easily inspect data, an IPython console and much more. It also includes powerful tools for code inspection and debugging, allowing the user to analyze code either entirely or line-by-line to detect and fix errors. The interface is also highly customizable, allowing you to choose things like the arrangement of the different panes and even the layout colour (we recommend Spyder Dark). For more information about Spyder, see the Spyder web page or the Spyder documentation.

<https://www.spyder-ide.org/>

Other IDEs not included in Anaconda

- PyCharm is a feature-rich IDE specifically designed for Python and is amongst the most popular. It contains most of the important tools for programming and is great for large-scale projects. Plus, it can easily be integrated with Anaconda. It has its own large, open source community. There is a free version of PyCharm, but the full professional version is paid.
<https://www.jetbrains.com/pycharm/features/>
- VS Code is another feature-rich IDE run by Microsoft that is compatible with almost every programming language. It is also free! The downside is that it is not Python-specific, meaning a plugin is required to run

Figure 2 ■ Windows Installation and Setup.**(a) Welcome Screen** **(b) Read/accept license agreement** **(c) Installation type** **(d) Choosing location for installation** **(e) Advanced installation options** **(f) Finish installation and close wizard**

Python, but it can be linked to Anaconda.

<https://code.visualstudio.com/docs/languages/python>

- Eclipse is probably the most popular IDE in the world as it is used for Java programming. It is free but requires a plugin to be compatible with Python. This would be an ideal IDE for any programmer with previous experience in Java.

<https://marketplace.eclipse.org/content/pydev-python-ide-eclipse>

- ATOM is a highly customizable IDE that shines with its integration of Git and Github, popular version control tools, making it useful for collaborations. It is easy to set up and very fast to load, but it's also not Python-specific, meaning it requires a Python plug-in.

<https://atom.io/packages/ide-python>

- Sublime Text 3 is a very simple code editor and is lightweight (does not require a heavy download). It is simple and easy to use. It is also compatible with many other programming languages. While technically not an IDE, it is useful for beginners and experts alike.

<https://realpython.com/setting-up-sublime-text-3-for-full-stack-python-development/>

Getting Started with Programming

As previously mentioned, there are many environments that can be used to program in Python. That being said, all of them share similar components, despite being displayed differently. These common features are the **Console** and **Editor**. The difference between these two are the following: the editor permits you to write and save any code/variables as a script and acts as a long-term memory. The console is like a short-term/working memory that allows the output to be displayed but nothing is saved. Thus, the console is used to test certain lines of code to ensure that they work, such as a program to calculate reaction time. The editor is used to save your work in memory, such as a script containing the entire code for your experimental paradigm. In figure 5, the console and editors are represented in all three IDEs (Spyder, Jupyter Notebook, and JupyterLab). Note that the layout differs, but in each IDE, the core features remain the same. It is important to also mention that for each IDE you encounter, certain features are added, making them best suited for specific tasks. Depending on your intentions, it is worth evaluating which

Figure 3 ■ Mac Installation and Setup.**(a) Welcome Screen** **(b) Read/accept license agreement** **(c) Installation type** **(d) Choosing location for installation** **(e) Advanced installation options** **(f) Finish installation and close wizard**

IDE would be best suited for your current needs.

For simplicities' sake, this tutorial will use Jupyter Notebook to demonstrate how to write your first program in Python! Now enough of the dry explanation, let's get coding!

Writing Your First Program

First off, open Jupyter Notebook through the Anaconda navigator. This should result in your default internet browser opening and the display should resemble Figure 6 (where all of your device's files should be available to access). Select the desired destination and follow the steps to create your first Python script.

1. Select "New" and choose the "Python 3" option to open a new Script/Notebook (see Fig. 6).
2. Once the notebook is created, in the top left corner, select the "Untitled" name and rename the notebook for convenience (see Fig. 7).
3. In the first cell, enter the following lines of code exactly as it is shown in the editor (see Fig. 8). While writing, you may notice certain aspects of the code will change colours. This will be elaborated further in a subsequent tutorial.
4. Once everything is properly entered in the editor, press

"Run" to execute your code and display the output in the console. This should result in two things. First, your console should display your code's output. Second, a new cell should appear beneath it (see Fig 9). Congratulations on this first step! You are now a Python programmer and are ready to explore the new horizons that await you!

* Note: to access your Jupyter Notebook again or any other .ipynb files, you must open them from the main Jupyter Notebook screen (Figure 6). Simply clicking on them from the desktop or other files will result in an error message.

Next Steps

Now that you have started programming, there might be questions or projects that you encounter requiring some extra help. A great resource for those situations is [Stack Overflow](#). This free forum for developers allows you to ask questions and get reliable help quickly. It also has a large repository of previously asked questions that you can navigate. Another important resource to explore is [GitHub](#). This is a website for storing, sharing and hosting code that can be downloaded by other programmers. In other words, GitHub allows users to share their scripts and

Figure 4 ■ Anaconda Navigator.

quickly collaborate on projects. There are other types of resources that a programmer in Python can access for free. As previously mentioned, the Python community is very large. This results in different types of collaboration happening virtually, such as taking on a [Kaggle coding challenge](#), or local meetups, such as the ones provided by the University of Ottawa's Python Club. Regardless of your next step, the most important action to take is to practice with patience and to dive into Python's programming language syntax. As with learning any new language, practice makes perfect! To become fluent in Python, it is important to immerse yourself in the Python community by writing questions online, reading blog posts, attending local programming talks, and following online courses ([Codecademy](#); [Udemy](#); [Official Python Tutorial](#)). In other words, your next step in learning Python should be fun, interactive, and challenging.

Authors' note

The authors can be contacted by email at (Damien Rolon-Mérette) drolo083@uottawa.ca, (Matt Ross) mross094@uottawa.ca, (Thaddé Rolon-Mérette) trolo068@uottawa.ca, or (Kinsey Church) kchur026@uottawa.ca.

References

- Coutanche, M. N., & Hallion, L. S. (2019). Machine learning for clinical psychology and clinical neuroscience.
- Dalmajer, E. S. (2016). *Python for experimental psychologists*. Routledge.
- Helmus, J. J., & Collis, S. M. (2016). The python arm radar toolkit (py-art), a library for working with weather radar data in the python programming language. *Journal of Open Research Software*, 4.
- Knorr, F. G., Marxen, M., & Petzold, J. (2019). Pyparadigm-a python library to build screens in a declarative way. *Frontiers in neuroinformatics*, 13, 59.
- Lu, Z., & Ku, Y. (2020). Neurora: A python toolbox of representational analysis from multi-modal neural data. *bioRxiv*.
- Saabith, A. S., Fareez, M., & Vinothraj, T. (2019). Python current trend applications-an overview. *International Journal of Advance Engineering and Research Development*, 6(10).
- Underwood, J. J., Kirchhoff, C., Warwick, H., & Gartstein, M. A. (2020). Leveraging python to process cross-cultural temperament interviews: A novel platform for text analysis. *Journal of Cross-Cultural Psychology*, 51(2), 168–181.
- Van Rossum, G., & Drake Jr, F. (1995). Python tutorial (vol. 620). *CWI Report CS-R9526, Amsterdam, Netherlands, msecke.karlin.mff.cuni.cz/~halas/IT/tutorial.pdf*.

Figure 5 ■ Basic Components of Three Different IDEs, the Console (A), Editor (B), Variable Explorer (C) and Help (E) are displayed in each IDE they apply to.

a) Spyder IDE

b) Jupyter Notebook

c) JupyterLab

Terminology

API : Application Programming Interface is a computing interface which defines interactions between multiple software intermediaries.

GUI : Graphical User Interface is a form of user interface that allows users to interact with electronic devices through graphical icons and audio indicators, such as primary notation, instead of text-based user interfaces, typed command labels or text navigation.

IDE : An Integrated Development Environment is a software application that provides comprehensive facilities to computer programmers for software development. An IDE normally consists of at least a source code editor, build automation tools, and a debugger.

Library : Collection of packages, i.e., Numpy, Matplotlib, Seaborn and Tensorflow.

Package : Collection of modules.

Module : A set of functions, types, classes, etc. Anything ending in .py.

Console : A console (also called a “shell”) is a command line interpreter that takes input from the user and interprets it. Cannot save or store information.

Figure 6 ■ Opening a New Notebook. **Figure 7 ■** Naming Your File.

Editor : A code editor is a tool that is used to write and edit code. It will save and store information as scripts. When running its code, the output is usually displayed in the console.

Citation

Rolon-Mérette, D., Ross, M., Rolon-Mérette, T., & Church, K. (2020). Introduction to Anaconda and Python: Installation and setup. *The Quantitative Methods for Psychology*, 16(5), S3–S11. doi:[10.20982/tqmp.16.5.S003](https://doi.org/10.20982/tqmp.16.5.S003)

Copyright © 2020, Rolon-Mérette, Ross, Rolon-Mérette, and Church. This is an open-access article distributed under the terms of the Creative Commons Attribution License (CC BY). The use, distribution or reproduction in other forums is permitted, provided the original author(s) or licensor are credited and that the original publication in this journal is cited, in accordance with accepted academic practice. No use, distribution or reproduction is permitted which does not comply with these terms.

Received: 11/08/2020 ~ Accepted: 23/09/2020

Figure 8 ■ Writing Your Code.

Figure 9 ■ Running Your Code.