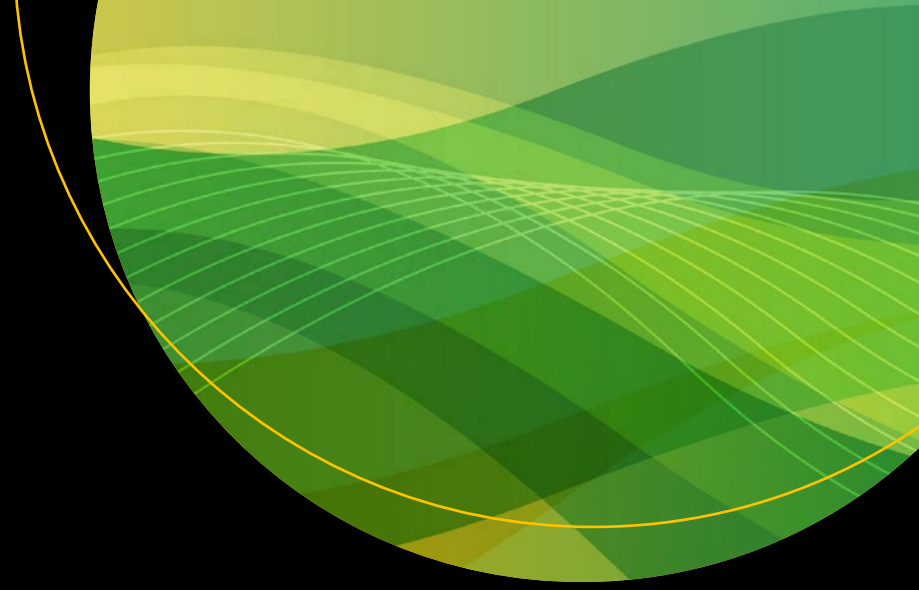

Python for SAS Users

A SAS-Oriented Introduction to Python

Randy Betancourt
Sarah Chen

Apress®

Python for SAS Users

A SAS-Oriented Introduction to Python

Randy Betancourt
Sarah Chen

Apress®

Python for SAS Users: A SAS-Oriented Introduction to Python

Randy Betancourt
Chadds Ford, PA, USA

Sarah Chen
Livingston, NJ, USA

ISBN-13 (pbk): 978-1-4842-5000-6
ISBN-13 (electronic): 978-1-4842-5001-3
https://doi.org/10.1007/978-1-4842-5001-3

Managing Director, Apress Media LLC: Welmoed Spahr
Acquisitions Editor: Susan McDermott
Development Editor: Laura Berendson
Coordinating Editor: Rita Fernando

Cover designed by eStudioCalamar

Cover image designed by Freepik (www.freepik.com)

Distributed to the book trade worldwide by Springer Science+Business Media New York, 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax (201) 348-4505, e-mail orders-ny@springer-sbm.com, or visit www.springeronline.com. Apress Media, LLC is a California LLC and the sole member (owner) is Springer Science + Business Media Finance Inc (SSBM Finance Inc). SSBM Finance Inc is a **Delaware** corporation.

For information on translations, please e-mail rights@apress.com, or visit http://www.apress.com/rights-permissions.

Apress titles may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Print and eBook Bulk Sales web page at http://www.apress.com/bulk-sales.

Any source code or other supplementary material referenced by the author in this book is available to readers on GitHub via the book's product page, located at www.apress.com/9781484250006. For more detailed information, please visit http://www.apress.com/source-code.

Printed on acid-free paper

Table of Contents

About the Authors

Randy Betancourt's professional career has been in and around data analysis. His journey began by managing a technical support group supporting over 2000 technical research analysts and scientists from the US Environmental Protection Agency at one of the largest mainframe complexes run by the federal government. He moved to Duke University, working for the administration, to analyze staff resource utilization and costs. There, he was introduced to the politics of data access as the medical school had most of the data and computer resources.

He spent the majority of his career at SAS Institute Inc. in numerous roles, starting in marketing and later moving into field enablement and product management. He subsequently developed the role for Office of the CTO consultant.

Randy traveled the globe meeting with IT and business leaders discussing the impact of data analysis to drive their business. And they also discussed challenges they faced. At the same time, he talked to end users, wanting to hear their perspective. Together, these experiences shaped his understanding of trade-offs that businesses make allocating scarce resources to data collection, analysis, and deployment of models.

More recently, he has worked as independent consultant for firms including the International Institute for Analytics, Microsoft's SQL Server Group, and Accenture's Applied Intelligence Platform.

Sarah Chen has 12 years of analytics experience in banking and insurance, including personal auto pricing, compliance, surveillance, and fraud analytics, sales analytics, credit risk modeling for business, and regulatory stress testing. She is a Fellow of both the Casualty Actuarial Society and the Society of Actuaries (FCAS, FSA), an actuary, data scientist, and innovator.

Sarah's career began with five and a half years at Verisk Analytics in the Personal Auto Actuarial division, building predictive models for various ISO products. At Verisk she learned and honed core skills in data analysis and data management.

Her skills and domain expertise were broadened when she moved to KPMG, working with leading insurers, banks, and large online platforms on diverse business and risk management problems.

From 2014 to present, Sarah has been working at HSBC bank on wholesale credit risk models. She has experiences in PD, LGD, and EAD models in commercial real estate, commercial and industrial banks, and non-bank financial institution portfolios. She has been active in innovations within the organization.

Over the years, she has used many analytics tools including R and SAS and Python.

Sarah graduated summa cum laude with BA in Mathematics from Columbia University in 2007. She is the founder of Magic Math Mandarin, a school that emphasizes values and tomorrow's skills for children.

About the Technical Reviewers

Ferrell Drewry wrote his first SAS program in 1977 and has continued to use SAS software throughout his 30-year career in the pharmaceutical industry that includes experience managing data management, programming, biostatistics, and information technology departments. Today, Ferrell works as a SAS programmer and manager (in that order) on phase II/III clinical trials.

Ferrell holds a BS in Accounting from the University of Northern Colorado and an MS in Business Administration with a concentration in Management Information Sciences from Colorado State University.

Ferrell lives on the southeastern North Carolina coast where he enjoys being close to his grandchildren, surfing, fishing, and tinkering with his Raspberry Pi (almost in that order).

Travis E. Oliphant is the Founder and CEO/CTO of Quansight, an innovation incubation company that builds and connects companies with open source communities to help gain actionable, quantitative insight from their data. Travis previously co-founded Anaconda, Inc. and is still a Director. Since 1997, he has worked in the Python ecosystem, notably as the primary creator of the NumPy package and as a founding contributor of the SciPy package. Travis also started the Numba project and organized and led the teams that built Conda, Dask, Bokeh, and XND. Travis holds a PhD from the Mayo Clinic and BS and MS in Mathematics and Electrical Engineering from Brigham Young University.

Acknowledgments

As this project developed, there are a number of individuals whose contributions and insights were invaluable. Alan Churchill provided many suggestions used to form the introduction and provided sound advice on how to effectively write the SAS and Python examples. Soon Tan, CEO of Ermas, and Kit Chaksuvej, Senior Technical Consultant at Ermas, provided ideas for improving the source code examples. Tom Weber provided detailed feedback on Chapter 8, "SASPy Module." Tom has been with SAS Institute's R&D for 31 years and recently co-authored "SASPy, a Python interface to Base SAS," available on GitHub.

We were fortunate to have Randy's former boss at SAS Institute, Ferrell Drewry, generously agree to review all of the SAS code and related content. He is an outstanding SAS programmer. Ferrell's meticulous attention to detail helped us tremendously.

And in another piece of good fortune, we received a great deal of benefit by having Travis Oliphant as a technical reviewer for the Python sections of the book. Not only was he generous with his time by providing detailed input, Travis' experiences with Python gave us insights into how the language is constructed. His commitment to the open source development community is a genuine inspiration.

We also want to acknowledge the great support from the Apress Media team we collaborated with. Susan McDermott, Senior Editor, helped us by navigating us through the process of book writing. We sincerely appreciate Susan's enthusiasm and encouragements, and Rita Fernando, coordinating editor, whose steady guidance helped to keep us on track and was always available to offer us assistance when we needed it.

Finally, a personal acknowledgment from Randy to his wife Jacqueline and sons Ethan and Adrian who were enthusiastic supporters from the very beginning of this project.

Introduction

For decades, Base SAS software has been the "gold standard" for data manipulation and analysis. The software can read any data source and is superb at transforming and shaping data for analysis. It has been the beneficiary of enormous resource investments over its lifetime. The company has one of the industry's most innovative R&D staff, and its products are well supported by an outstanding technical support and well documented by very capable technical writers. SAS Institute Inc. has remained focused on gathering customer input and building desired features. All of these characteristics help explain its popularity.

Since the beginning of this millennium, the accelerated growth of open source software has produced outstanding projects offering data scientists enormous capabilities to tackle problems that were previously considered outside the realm of feasibility. Chief among these is Python. Python has its heritage in scientific and technical computing domains and has a very compact syntax. It is a full-featured language that is relatively easy to learn and is able to scale offering good performance with large data volumes. This is one of the reasons why firms like Netflix[1] use it so extensively.

By nature, SAS users are intrepid and are constantly trying to find new ways to expand the use of the software in pursuit of meeting business objectives. And given the extensive role of SAS within organizations, it only makes sense to find ways to combine the capabilities of these two languages to complement one another.

We have four main goals for our readers. The first is to provide a quick start to learning Python for users already familiar with the SAS language.

Both languages have advantages and disadvantages when it comes to a particular task. And since they are programming languages, their designers had to make certain trade-offs which can manifest themselves as features or quirks, depending on one's perspective. This is our second goal: help readers compare and contrast common tasks taking into account differences in their default behaviors. For example, SAS names are

[1]Python at Netflix, at https://medium.com/netflix-techblog/python-at-netflix-bba45dae649e

case-insensitive, while Python names are case-sensitive. Or the default sort sequence for the pandas library is the opposite of SAS' default sort sequence and so on.

Rather than attempting to promote one language over the other, our third goal is to point out the integration points between the two languages. The choice of which tool to utilize for a given task typically comes down to a combination of what you as a user are familiar with and the context of the problem being solved. Knowing both languages enlarges the set of tools you can apply for the task at hand.

And finally, our fourth goal is to develop working examples for all of the topics in both Python and SAS which allows you the opportunity to "try out" the examples by not just executing them but by extending them to suit your own needs.

We assume you already have some basic knowledge of Python, for example, you already know how to import modules and execute Python scripts. If you don't, then you will want to spend more time with Chapter 1, "Introduction," covering topics such as Python installation, executing Python in a Windows environment, and executing Python in a Linux environment.

In Chapter 2, "Python Types and Formatting," we cover topics related to the Python Standard Library such as data types, Booleans with a focus on truth testing, numerical and string manipulations, and basic formatting. If you are new to Python, then it is worthwhile to spend time on this chapter practicing execution of the Python and SAS examples.

If you have a solid grasp of Python Standard Library, you can skip to Chapter 3, "pandas Library." Beyond introducing you to DataFrames, we deal with the missing data problem endemic to any analysis task. The understanding of the pandas library underpins the remainder of the book.

Chapter 4, "Indexing and Grouping," extends your knowledge of the pandas library by focusing on DataFrame indexing and GroupBy operations. A detailed understanding of these operations is essential for shaping data. We end this chapter by introducing techniques you can use for report production.

Data manipulation such as merging, concatenation, subsetting, updating, appending, sorting, finding duplicates, drawing samples, and transposing are covered in Chapter 5. We have developed scores of examples in both Python and SAS to address and illustrate the range of problems you commonly face in preparing data for analysis.

In Chapter 6, "pandas Readers," we cover many of the popular readers and writers used to read and write data from a range of different sources including Excel, .csv files, relational databases, JSON, web APIs, and more. And while we offer detailed

explanations, it is the numerous working examples you can use in your own work that make this chapter so valuable.

Working with date, datetime, time, and time zone is the focus in Chapter 7. In this increasingly instrumented world we live in, we are faced with processing time-based data from literally trillions of sensors. Forming and appropriately handling time Series data is no longer just the domain for time-based forecasting. Once again, we rely on the breadth of the provided examples to help you improve your skills.

In our last chapter, we introduce and discuss SASPy, the open source library from SAS Institute used to expose a Python interface to Base SAS software. The provided examples focus on building useful pipelines where the strengths of both languages come together in a single program to accomplish common data analysis tasks. This integration point between SAS and Python offers an enormous range of possibilities limited only by your imagination.

We hope you enjoy this book as much as we enjoyed putting it together!

Feedback

We would love to receive your feedback. Tell us what you liked, what you didn't like, and provide suggestions for improvements. You can go to our web site, `www.pythonforsasusers.com`, where you can find all of the examples from this book, get updated examples, and provide us feedback.

CHAPTER 1

Why Python?

There are plenty of substantive open source software projects out there for data scientists, so why Python?[1] After all, there is the R language. R is a robust and well-supported language written initially by statistician for statisticians. Our view is not to promote one language over the other. The goal is to illustrate how the addition of Python to the SAS user's toolkit is a means for valuable skills augmentation. Besides, Bob Muenchen has already written *R for SAS and SPSS Users.*[2]

Python is used in a wide range of computing applications from web and internet development to scientific and numerical analysis. Its pedigree from the realm of scientific and technical computing domains gives the language a natural affinity for data analysis. This is one of the reasons why Google uses it so extensively and has developed an outstanding tutorial for programmers.[3]

Perhaps the best answer as to why Python is best expressed in the *Zen of Python*, written by Tim Peters.[4] While these are design principles used to influence the development of a language like Python, they apply (mostly) to our own efforts. These aphorisms are worth bookmarking and re-reading periodically.

[1]Python is an open source language promoted, protected, and advanced by the Python Software Foundation: `https://www.python.org/psf/`. It is currently developed on GitHub

[2]Muenchen, Robert A, (2011). R for SAS and SPSS Users, 2nd Edition.

[3]Google's Python Class at `https://developers.google.com/edu/python/`

[4]PEP 20—The Zen of Python at `www.python.org/dev/peps/pep-0020/`

R. Betancourt and S. Chen, *Python for SAS Users*, https://doi.org/10.1007/978-1-4842-5001-3_1

Setting Up a Python Environment

One of the first questions a new Python user is confronted with is which version to use, Python 2 or Python 3. For this writing we used Python 3.6.4 (Version 3.6, Maintenance 4). The current release of Python is 3.7.2, released on December 24, 2018. Python release 3.8.0 is expected in November 2019. As with any language, minor changes in syntax occur as the developers make feature improvements and Python is no exception. We have chosen Python 3.6 since this was the latest release as time of writing and the release of 3.7 has not impacted any of the chapters. You can read more about the differences between Python 2 and Python 3.[5]

An attractive feature for Python is the availability of community-contributed modules. Python comes with a base library or core set of modules, referred to as the Standard Library. Due to Python's design, individuals and organizations contribute to the creation of thousands of additional modules which are mostly written in Python. Interested in astronomical calculations used to predict any planet's location in space? Then the kplr package is what you need.[6] Closer to home, we will utilize the Python-dateutil 2.7.3 package to extend Python's base capabilities for handling datetime arithmetic.[7]

Just as you can configure your SAS development environment in numerous ways, the same is true for Python. And while there are various implementations of Python, such as Jython, IronPython, and PyPy to make life simpler, organizations package distributions for you so you can avoid having to understand dependencies or using build scripts to assemble a custom environment. At the time of this writing, we are using the Anaconda distribution 5.2.0 for Windows 10 located at Anaconda's distribution page at `www.anaconda.com/download/`.

The Anaconda distribution of Python also supports OSX and Linux. They conveniently take care of all the details for you by providing familiar tools for installing, uninstalling, upgrading, determining package dependencies, and so on. But they do much more than just make a convenient distribution. They provide detailed documentation, support a community of enthusiastic users, and offer a supported enterprise product around the free distribution.

---

[5]Should I use Python 2 or Python 3 for my development activity? at `https://wiki.python.org/moin/Python2orPython3`

[6]kplr: A Python interface to the Kepler data at `http://dfm.io/kplr/`

[7]Python-dateutil 2.7.3 at `https://pypi.org/project/python-dateutil/`

Anaconda3 Install Process for Windows

The following text describes the steps for installing a new version of Python 3.6. If you have an existing version already installed, you can either uninstall the older version or follow the instructions for managing multiple Python installs at https://conda.io/docs/user-guide/tasks/manage-python.html.

1. From www.anaconda.com/download/ download the Anaconda3-5.2.0 for Windows Installer for Python 3.6. Select the 32-bit or 64-bit installer (depending on your Windows machine architecture).
2. From this download location on your machine, you should see the file Anaconda3-5.2.0-Windows-x86-64.exe (assuming your Windows machine is 64-bit) for launching the Windows Installer.
3. Launch the Windows Installer (see Figure 1-1).

Figure 1-1. *Windows Installer*

4. Click Next to review the license agreement and click the "I Agree" button (see Figure 1-2).

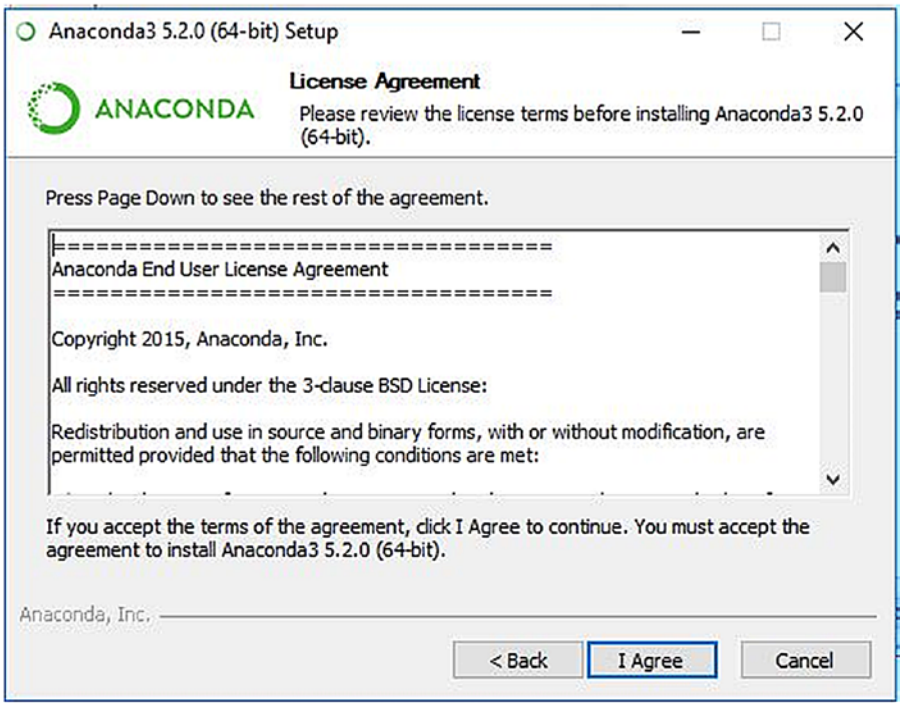

Figure 1-2. *License Agreement*

5. Select the installation type, stand-alone or multi-user (see Figure 1-3).

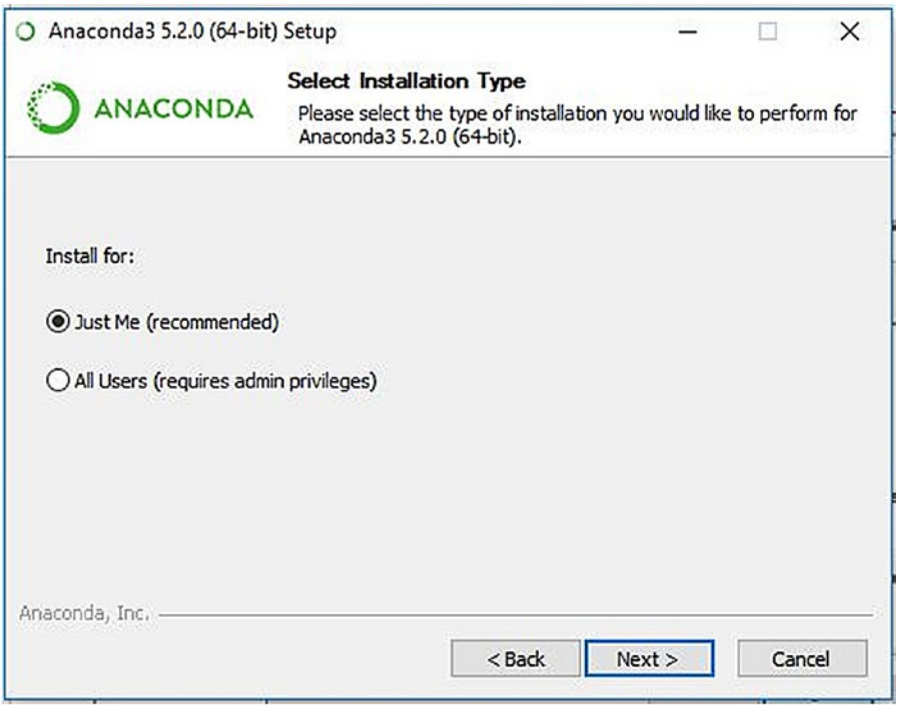

Figure 1-3. *Select Installation Type*

6. Select the installation location (see Figure 1-4).

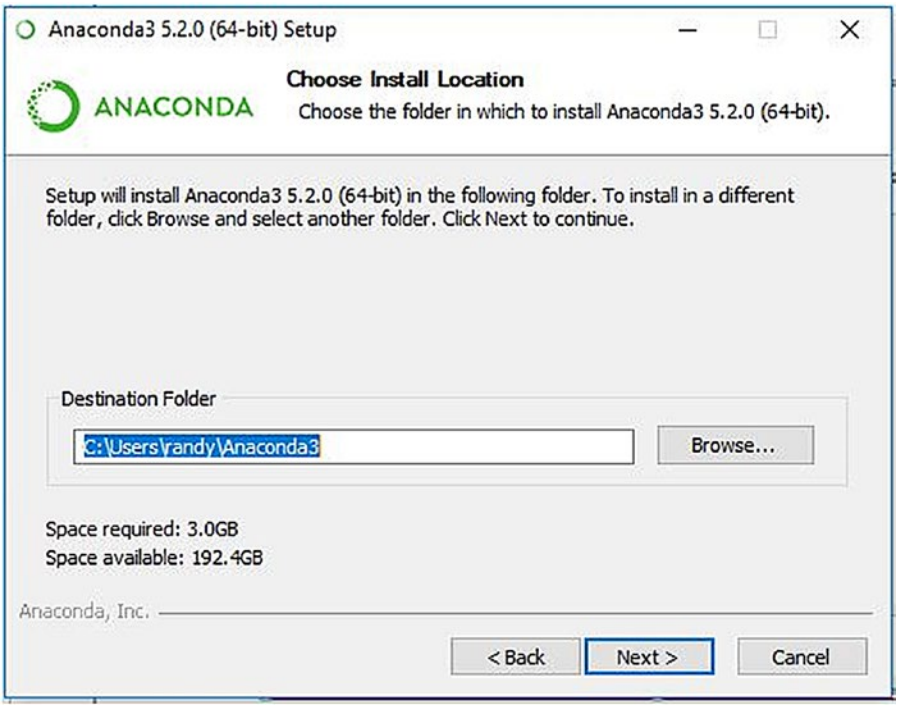

Figure 1-4. *Select Installation Location*

7. Register Anaconda as the default Python 3.6 installation by ensuring the "Register Anaconda as my default Python 3.6" box is checked. Press the "Install" button (see Figure 1-5).

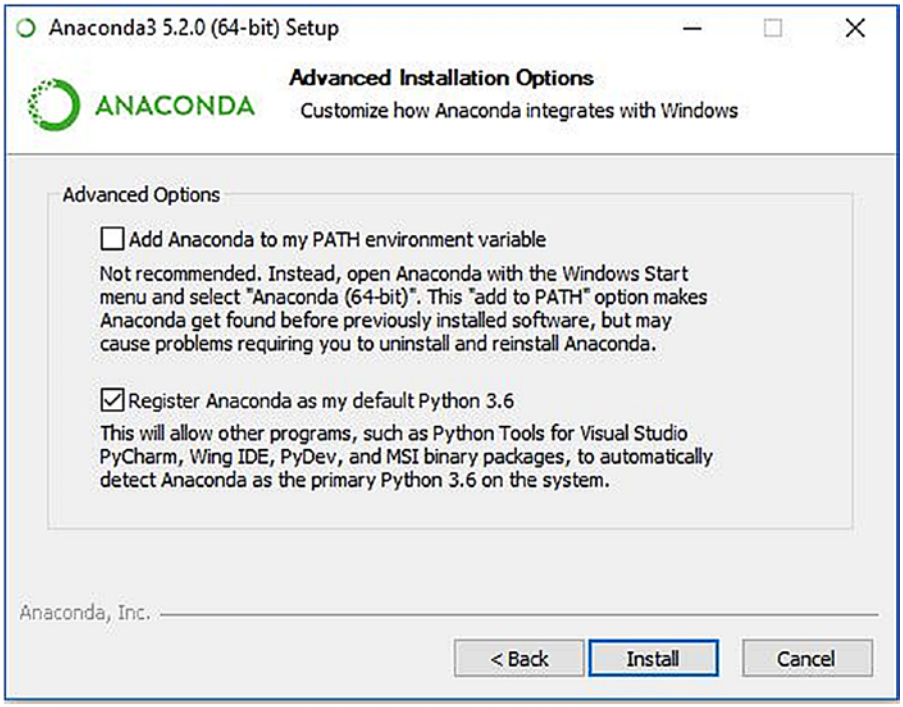

Figure 1-5. *Advanced Installation Options*

8. Start the installation process (see Figure 1-6). You may be asked if you would like to install Microsoft's Visual Studio Code. Visual Studio provides a visual interface for constructing and debugging Python scripts. It is an optional component and is not used in this book.

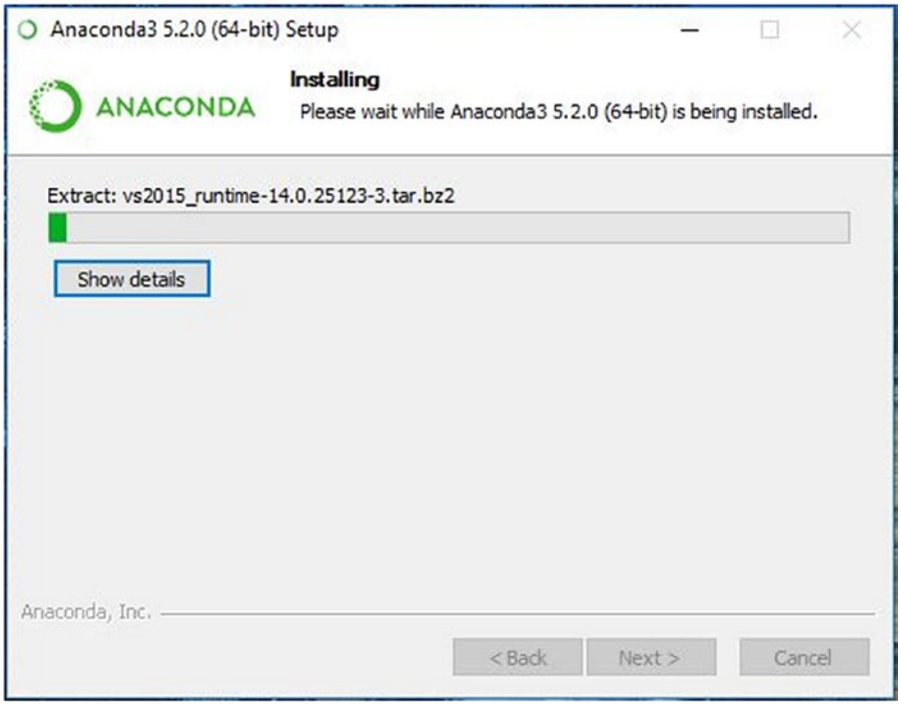

Figure 1-6. *Start Installation*

9. Validate the install by opening a Windows Command Prompt window and enter (after the > symbol prompt):

   ```
   > python
   ```

Assuming the installation worked correctly, the output should look similar to Listing 1-1.

Listing 1-1. Python Command for Windows

```
C:\Users\randy>python
Python 3.6.5 |Anaconda, Inc.| (default, Mar 29 2018, 13:32:41) [MSC v.1900
64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

This is an indication that the installation is complete including modifications made to Windows environment variable PATH.

Troubleshooting Python Installation for Windows

If you receive the error message, 'Python' is not recognized as an internal or external command then ensure the Windows PATH environment variable has been updated to include the location of the Python installation directory.

1. On Windows 10, open File Explorer and select "Properties" for "This PC" (see Figure 1-7).

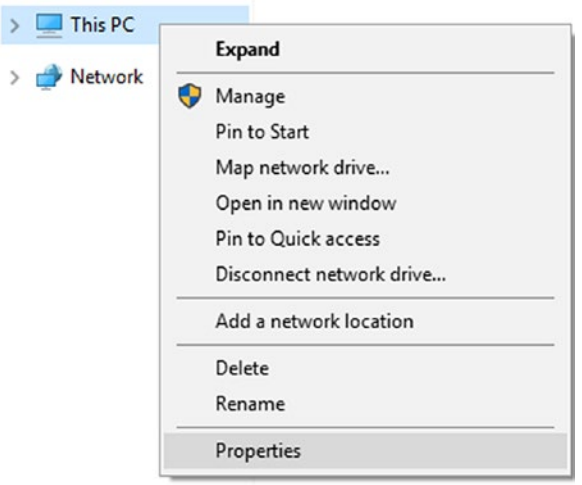

Figure 1-7. *PC Properties*

2. Right-click the "Properties" dialog to open the Control Panel for the System (see Figure 1-8).

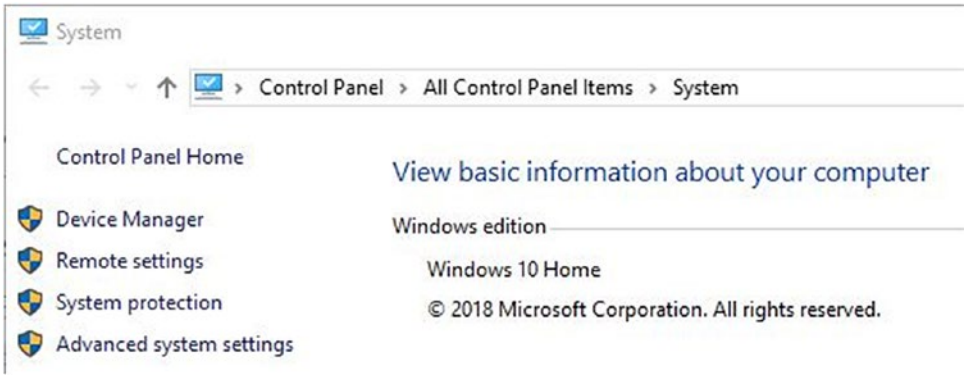

Figure 1-8. *PC Control Panel*

3. Select “Advanced” tab for System Properties and press the Environment Variables… button (see Figure 1-9).

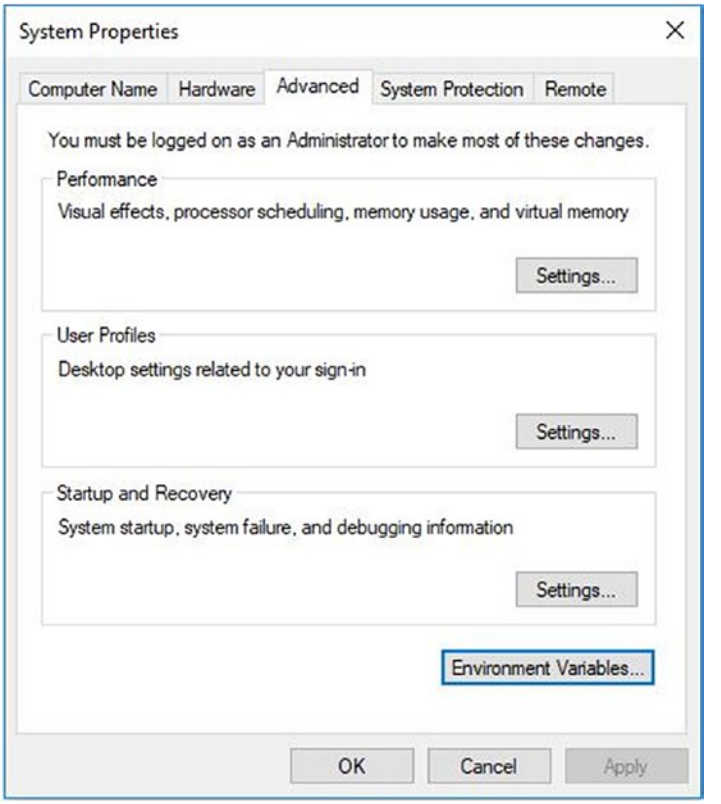

Figure 1-9. *Advanced System Properties*

4. Highlight the “Path” Environment Variables (see Figure 1-10).

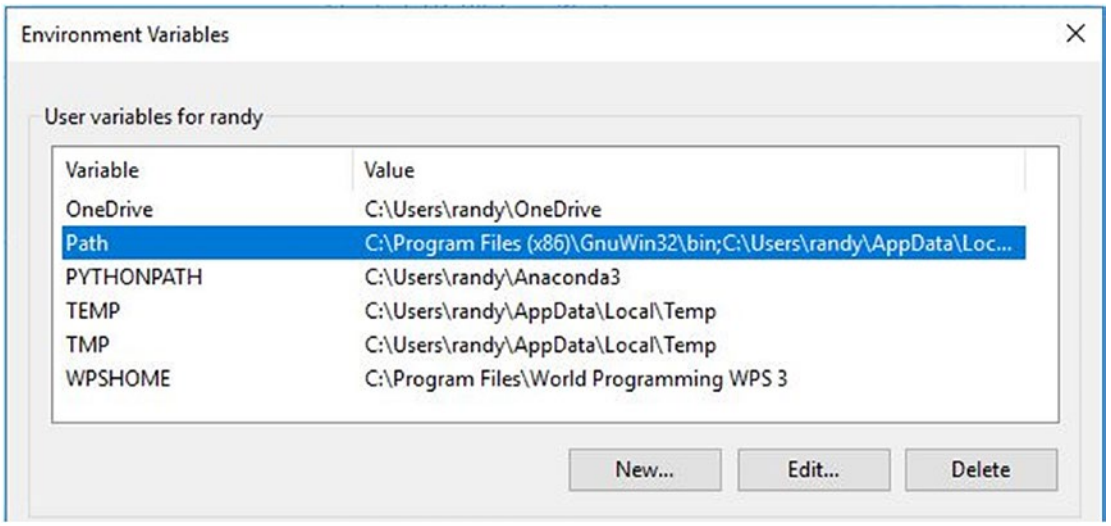

Figure 1-10. *Environment Variables*

5. Edit the Path Environment Variables by clicking the "New" button (see Figure 1-11).

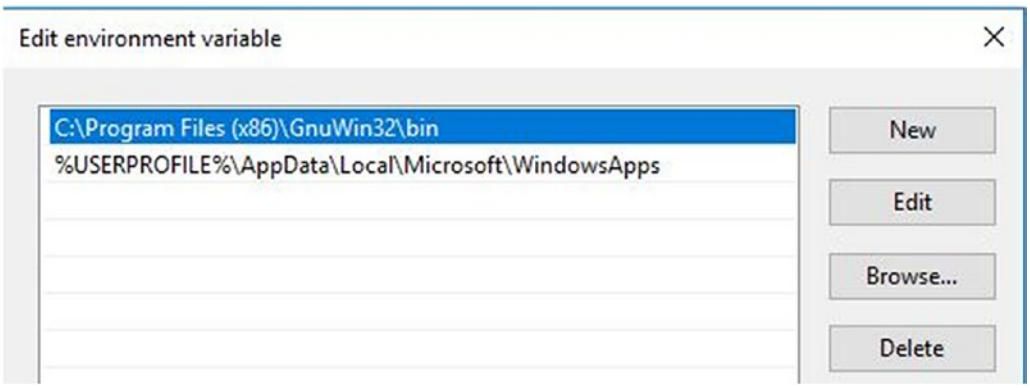

Figure 1-11. *Edit Environment Variables*

6. Add the Anaconda Python installation path specified in step 6 from the Anaconda3 Install Process for Windows as seen earlier (see Figure 1-12).

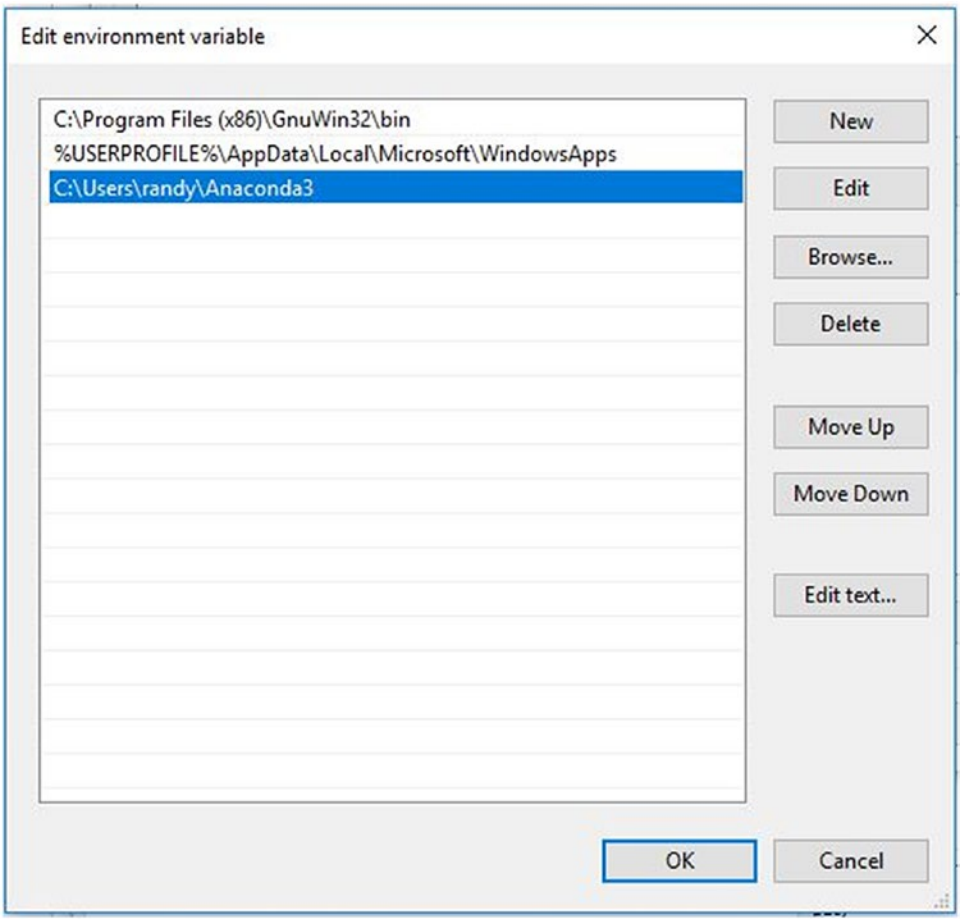

Figure 1-12. *Add Anaconda3 to Path*

7. Ensure the path you entered is correct and click "OK".
8. To validate start a new Windows Command Prompt and enter the command "Python". The output should look similar to the one in Listing 1-2.

Listing 1-2. Validate Python for Windows

```
C:\Users\randy>python
Python 3.6.5 |Anaconda, Inc.| (default, Mar 29 2018, 13:32:41) [MSC v.1900
64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

The three angle brackets (>>>) is the default prompt for Python 3.

Anaconda3 Install Process for Linux

The following are the steps to install Python 3.6 in a Linux environment.

1. From www.anaconda.com/download/ download the Anaconda3-5.2.0 for Linux Installer for Python 3.6. This is actually a script file. Select the 32-bit or 64-bit installer (depending on your machine architecture). Select "Save File" and click "OK" (see Figure 1-13).

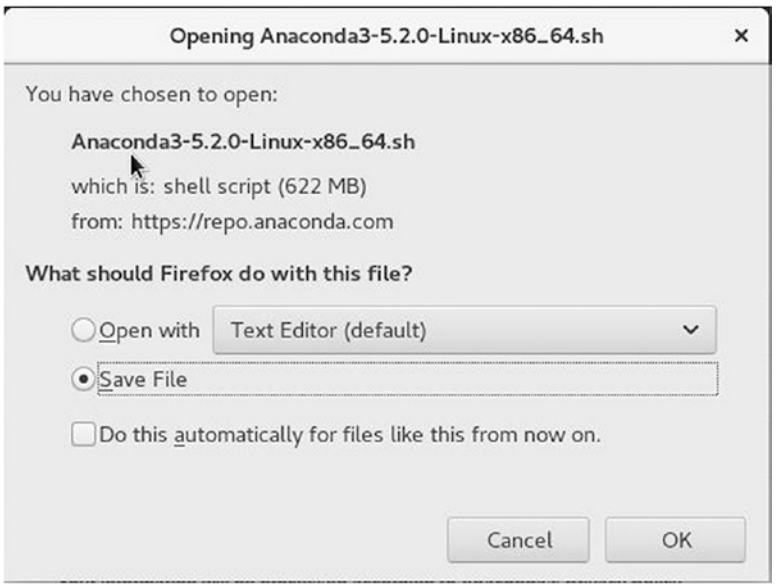

Figure 1-13. *Execute Linux Install*

2. Open a Linux terminal window and navigate to the location for the default directory /<userhome>/Downloads.

```
$ cd /home/randy/Downloads
```

3. Change the permission to allow the script to execute with chmod command.

```
$ chmod +x Anaconda3-5.2.0-Linux-x86_64.sh
```

4. If you are using a Bash shell, you can execute the shell script with ./ preceding the script filename (see Figure 1-14).

```
$ ./Anaconda3-5.2.0-Linux-x86_64.sh
```

Alternatively, you can execute the script with the following (Figure 1-14):

```
$ sh Anaconda3-5.1.0-Linux-x86_64.sh
```

```
drwxr-xr-x.  2 randy randy        45 Jul 17 11:21 .
drwx------. 15 randy randy      4096 Jul 17 11:00 ..
-rw-rw-r--.  1 randy randy 651745206 Jul 17 11:22 Anaconda3-5.2.0-Linux-x86_64.sh
[randy@rhel7-4 Downloads]$ chmod +x Anaconda3-5.2.0-Linux-x86_64.sh
[randy@rhel7-4 Downloads]$ ./Anaconda3-5.2.0-Linux-x86_64.sh
```

Figure 1-14. *Execute Script*

5. Press <enter> to continue and display the License Agreement (Figure 1-15).

```
Welcome to Anaconda3 5.2.0

In order to continue the installation process, please review the license
agreement.
Please, press ENTER to continue
>>>
```

Figure 1-15. *License Agreement*

6. Accept the license term by entering “yes” and pressing <enter> (Figure 1-16).

```
Do you accept the license terms? [yes|no]
[no] >>> yes
```

Figure 1-16. *Accept License Terms*

7. Confirm the Anaconda3 installation directory and press <enter> (Figure 1-17).

```
Anaconda3 will now be installed into this location:
/home/randy/anaconda3

  - Press ENTER to confirm the location
  - Press CTRL-C to abort the installation
  - Or specify a different location below

[/home/randy/anaconda3] >>>
```

Figure 1-17. *Confirm Install Location*

8. Append the Anaconda3 installation directory to the $PATH environment variable by entering "yes" and pressing <enter> (Figure 1-18).

```
installation finished.
Do you wish the installer to prepend the Anaconda3 install location
to PATH in your /home/randy/.bashrc ? [yes|no]
[no] >>> yes
```

Figure 1-18. *Append to $PATH Variable*

You may be asked if you would like to install Microsoft's Visual Studio. Visual Studio provides a visual interface for constructing and debugging Python scripts. It is an optional component and is not used in this book.

9. Confirm the installation by closing the terminal window used to execute the installation script and opening a new terminal window. This action will execute the .bashrc file in your home directory and "pick up" the updated $PATH environment variable that includes the Anaconda3 installation directory (Figure 1-19).

```
[randy@rhel7-4 ~]$ python
Python 3.6.5 |Anaconda, Inc.| (default, Apr 29 2018, 16:14:56)
[GCC 7.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Figure 1-19. *Confirm Installation*