

# Mastering Machine Learning with Python in Six Steps

A Practical Implementation Guide to  
Predictive Data Analytics Using Python

---

Manohar Swamynathan



Apress®

# Mastering Machine Learning with Python in Six Steps

A Practical Implementation Guide  
to Predictive Data Analytics Using  
Python



Manohar Swamynathan

Apress®

## ***Mastering Machine Learning with Python in Six Steps***

Manohar Swamynathan  
Bangalore, Karnataka, India

ISBN-13 (pbk): 978-1-4842-2865-4  
DOI 10.1007/978-1-4842-2866-1

ISBN-13 (electronic): 978-1-4842-2866-1

Library of Congress Control Number: 2017943522

Copyright © 2017 by Manohar Swamynathan

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

Trademarked names, logos, and images may appear in this book. Rather than use a trademark symbol with every occurrence of a trademarked name, logo, or image we use the names, logos, and images only in an editorial fashion and to the benefit of the trademark owner, with no intention of infringement of the trademark.

The use in this publication of trade names, trademarks, service marks, and similar terms, even if they are not identified as such, is not to be taken as an expression of opinion as to whether or not they are subject to proprietary rights.

While the advice and information in this book are believed to be true and accurate at the date of publication, neither the authors nor the editors nor the publisher can accept any legal responsibility for any errors or omissions that may be made. The publisher makes no warranty, express or implied, with respect to the material contained herein.

Cover image designed by Freepik

Managing Director: Welmoed Spahr  
Editorial Director: Todd Green  
Acquisitions Editor: Celestin Suresh John  
Development Editor: Anila Vincent and James Markham  
Technical Reviewer: Jojo Moolayil  
Coordinating Editor: Sanchita Mandal  
Copy Editor: Karen Jameson  
Compositor: SPi Global  
Indexer: SPi Global  
Artist: SPi Global

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](mailto:orders-ny@springer-sbm.com), or visit [www.springeronline.com](http://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](mailto: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/978-1-4842-2865-4](http://www.apress.com/978-1-4842-2865-4). For more detailed information, please visit <http://www.apress.com/source-code>.

Printed on acid-free paper

# Contents at a Glance

**About the Author ..... xiii**

**About the Technical Reviewer ..... xv**

**Acknowledgments ..... xvii**

**Introduction ..... xix**

**■ Chapter 1: Step 1 – Getting Started in Python..... 1**

**■ Chapter 2: Step 2 – Introduction to Machine Learning..... 53**

**■ Chapter 3: Step 3 – Fundamentals of Machine Learning..... 117**

**■ Chapter 4: Step 4 – Model Diagnosis and Tuning ..... 209**

**■ Chapter 5: Step 5 – Text Mining and Recommender Systems.... 251**

**■ Chapter 6: Step 6 – Deep and Reinforcement Learning..... 297**

**■ Chapter 7: Conclusion ..... 345**

**Index..... 351**

# Contents

<b>About the Author .....</b>	<b>xiii</b>
<b>About the Technical Reviewer .....</b>	<b>xv</b>
<b>Acknowledgments .....</b>	<b>xvii</b>
<b>Introduction .....</b>	<b>xix</b>
<b>■ Chapter 1: Step 1 – Getting Started in Python.....</b>	<b>1</b>
The Best Things in Life Are Free.....	1
The Rising Star .....	2
Python 2.7.x or Python 3.4.x?.....	3
Windows Installation .....	4
OSX Installation .....	4
Linux Installation .....	4
Python from Official Website .....	4
Running Python .....	5
Key Concepts.....	5
Python Identifiers.....	5
Keywords.....	6
My First Python Program .....	6
Code Blocks (Indentation & Suites) .....	6
Basic Object Types.....	8
When to Use List vs. Tuples vs. Set vs. Dictionary.....	10
Comments in Python.....	10
Multiline Statement .....	11

Basic Operators .....	12
Control Structure .....	20
Lists .....	22
Tuple .....	26
Sets.....	29
Dictionary .....	37
User-Defined Functions .....	42
Module.....	45
File Input/Output.....	47
Exception Handling.....	48
Endnotes .....	52
<b>■ Chapter 2: Step 2 – Introduction to Machine Learning.....</b>	<b>53</b>
History and Evolution .....	54
Artificial Intelligence Evolution.....	57
Different Forms .....	58
Statistics.....	58
Data Mining .....	61
Data Analytics.....	61
Data Science.....	64
Statistics vs. Data Mining vs. Data Analytics vs. Data Science .....	66
Machine Learning Categories.....	67
Supervised Learning.....	67
Unsupervised Learning.....	68
Reinforcement Learning .....	69
Frameworks for Building Machine Learning Systems.....	69
Knowledge Discovery Databases (KDD) .....	69
Cross-Industry Standard Process for Data Mining .....	71

SEMMA (Sample, Explore, Modify, Model, Assess).....	74
KDD vs. CRISP-DM vs. SEMMA.....	75
<b>Machine Learning Python Packages .....</b>	<b>76</b>
<b>Data Analysis Packages .....</b>	<b>76</b>
NumPy.....	77
Pandas.....	89
Matplotlib.....	100
<b>Machine Learning Core Libraries .....</b>	<b>114</b>
<b>Endnotes .....</b>	<b>116</b>
<b>■ Chapter 3: Step 3 – Fundamentals of Machine Learning.....</b>	<b>117</b>
Machine Learning Perspective of Data.....	117
Scales of Measurement.....	118
Nominal Scale of Measurement .....	118
Ordinal Scale of Measurement .....	119
Interval Scale of Measurement.....	119
Ratio Scale of Measurement .....	119
Feature Engineering .....	120
Dealing with Missing Data.....	121
Handling Categorical Data .....	121
Normalizing Data .....	123
Feature Construction or Generation.....	125
Exploratory Data Analysis (EDA) .....	125
Univariate Analysis .....	126
Multivariate Analysis .....	128
Supervised Learning– Regression .....	131
Correlation and Causation .....	133
Fitting a Slope.....	134
How Good Is Your Model? .....	136

Polynomial Regression .....	139
Multivariate Regression .....	143
Multicollinearity and Variation Inflation Factor (VIF) .....	145
Interpreting the OLS Regression Results .....	149
Regression Diagnosis .....	152
Regularization .....	156
Nonlinear Regression .....	159
<b>Supervised Learning – Classification .....</b>	<b>160</b>
Logistic Regression .....	161
Evaluating a Classification Model Performance .....	164
ROC Curve .....	166
Fitting Line .....	167
Stochastic Gradient Descent .....	168
Regularization .....	169
Multiclass Logistic Regression .....	171
Generalized Linear Models .....	173
Supervised Learning – Process Flow .....	175
Decision Trees .....	176
Support Vector Machine (SVM) .....	180
k Nearest Neighbors (kNN) .....	183
Time-Series Forecasting .....	185
<b>Unsupervised Learning Process Flow .....</b>	<b>194</b>
Clustering .....	195
K-means .....	195
Finding Value of k .....	199
Hierarchical Clustering .....	203
Principal Component Analysis (PCA) .....	205
<b>Endnotes .....</b>	<b>208</b>



<b>■ Chapter 4: Step 4 – Model Diagnosis and Tuning .....</b>	<b>209</b>
Optimal Probability Cutoff Point .....	209
Which Error Is Costly? .....	213
Rare Event or Imbalanced Dataset .....	213
Known Disadvantages .....	216
Which Resampling Technique Is the Best? .....	217
Bias and Variance .....	218
Bias .....	218
Variance .....	218
K-Fold Cross-Validation .....	219
Stratified K-Fold Cross-Validation .....	221
Ensemble Methods .....	221
Bagging .....	222
Feature Importance .....	224
RandomForest .....	225
Extremely Randomized Trees (ExtraTree) .....	225
How Does the Decision Boundary Look? .....	226
Bagging – Essential Tuning Parameters .....	228
Boosting .....	228
Example Illustration for AdaBoost .....	229
Gradient Boosting .....	233
Boosting – Essential Tuning Parameters .....	235
Xgboost (eXtreme Gradient Boosting) .....	236
Ensemble Voting – Machine Learning’s Biggest Heroes United .....	240
Hard Voting vs. Soft Voting .....	242
Stacking .....	244

Hyperparameter Tuning .....	246
GridSearch .....	247
RandomSearch .....	248
Endnotes .....	250
<b>■ Chapter 5: Step 5 – Text Mining and Recommender Systems ....</b>	<b>251</b>
Text Mining Process Overview .....	252
Data Assemble (Text).....	253
Social Media .....	255
Step 1 – Get Access Key (One-Time Activity).....	255
Step 2 – Fetching Tweets .....	255
Data Preprocessing (Text) .....	259
Convert to Lower Case and Tokenize .....	259
Removing Noise.....	260
Part of Speech (PoS) Tagging .....	262
Stemming .....	263
Lemmatization .....	265
N-grams.....	267
Bag of Words (BoW).....	268
Term Frequency-Inverse Document Frequency (TF-IDF) .....	270
Data Exploration (Text) .....	272
Frequency Chart .....	272
Word Cloud .....	273
Lexical Dispersion Plot .....	274
Co-occurrence Matrix.....	275
Model Building .....	276
Text Similarity.....	277
Text Clustering.....	279

Latent Semantic Analysis (LSA) .....	280
<b>Topic Modeling .....</b>	<b>282</b>
Latent Dirichlet Allocation (LDA) .....	282
Non-negative Matrix Factorization .....	284
<b>Text Classification .....</b>	<b>284</b>
<b>Sentiment Analysis.....</b>	<b>286</b>
<b>Deep Natural Language Processing (DNLP) .....</b>	<b>287</b>
<b>Recommender Systems .....</b>	<b>291</b>
Content-Based Filtering.....	292
Collaborative Filtering (CF) .....	292
<b>Endnotes .....</b>	<b>295</b>
<b>■ Chapter 6: Step 6 – Deep and Reinforcement Learning.....</b>	<b>297</b>
Artificial Neural Network (ANN).....	298
What Goes Behind, When Computers Look at an Image?.....	299
Why Not a Simple Classification Model for Images?.....	300
Perceptron – Single Artificial Neuron .....	300
Multilayer Perceptrons (Feedforward Neural Network).....	303
Load MNIST Data .....	304
Key Parameters for scikit-learn MLP .....	305
Restricted Boltzman Machines (RBM).....	307
MLP Using Keras .....	312
Autoencoders .....	315
Dimension Reduction Using Autoencoder.....	316
De-noise Image Using Autoencoder .....	319
Convolution Neural Network (CNN) .....	320
CNN on CIFAR10 Dataset .....	321
CNN on MNIST Dataset .....	327

Recurrent Neural Network (RNN) .....	332
Long Short-Term Memory (LSTM).....	333
Transfer Learning .....	336
Reinforcement Learning .....	340
Endnotes .....	344
<b>■ Chapter 7: Conclusion .....</b>	<b>345</b>
Summary .....	345
Tips.....	346
Start with Questions/Hypothesis Then Move to Data! .....	347
Don't Reinvent the Wheels from Scratch.....	347
Start with Simple Models .....	348
Focus on Feature Engineering .....	349
Beware of Common ML Imposters .....	349
Happy Machine Learning.....	349
<b>Index.....</b>	<b>351</b>

# About the Author



**Manohar Swamynathan** is a data science practitioner and an avid programmer, with over 13 years of experience in various data science-related areas that include data warehousing, Business Intelligence (BI), analytical tool development, ad hoc analysis, predictive modeling, data science product development, consulting, formulating strategy, and executing analytics program.

He's had a career covering life cycles of data across different domains such as U.S. mortgage banking, retail, insurance, and industrial IoT. He has a bachelor's degree with specialization in physics, mathematics, and computers; and a master's degree in project management. He's currently living in Bengaluru, the Silicon Valley of India, working as Staff Data Scientist with General Electric Digital, contributing to the next big digital industrial revolution.

You can visit him at <http://www.mswamynathan.com> to learn more about his various other activities.

# About the Technical Reviewer



**Jojo Moolayil** is a Data Scientist and the author of the book: *Smarter Decisions – The Intersection of Internet of Things and Decision Science*. With over 4 years of industrial experience in Data Science, Decision Science and IoT, he has worked with industry leaders on high impact and critical projects across multiple verticals. He is currently associated with **General Electric**, the pioneer and leader in data science for Industrial IoT and lives in Bengaluru—the silicon valley of India.

He was born and raised in Pune, India and graduated from University of Pune with a major in Information Technology Engineering. He started his career with Mu Sigma Inc., the world's largest pure play analytics provider and worked with the leaders of many Fortune 50 clients. One of the early enthusiasts to venture into IoT analytics, he converged his learnings from decision science to bring the problem solving frameworks and his learnings from data and decision science to IoT Analytics.

To cement his foundations in data science for industrial IoT and scale the impact of the problem solving experiments, he joined a fast growing IoT Analytics startup called Flutura based in Bangalore and headquartered in the valley. After a short stint with Flutura, Jojo moved on to work with the leaders of Industrial IoT - General Electric, in Bangalore, where he focused on solving decision science problems for Industrial IoT use cases. As a part of his role in GE, Jojo also focuses on developing data science and decision science products and platforms for Industrial IoT.

Apart from authoring books on Decision Science and IoT, Jojo has also been Technical Reviewer for various books on Machine Learning, Deep Learning and Business Analytics with Apress. He is an active Data Science tutor and maintains a blog at <http://www.jojomoolayil.com/web/blog/>.

## Profile

<http://www.jojomoolayil.com/>

<https://www.linkedin.com/in/jojo62000>

I would like to thank my family, friends and mentors.

—Jojo Moolayil

# Acknowledgments

I'm grateful to my mom, dad, and loving brother; I thank my wife Usha and son Jivin for providing me the space for writing this book.

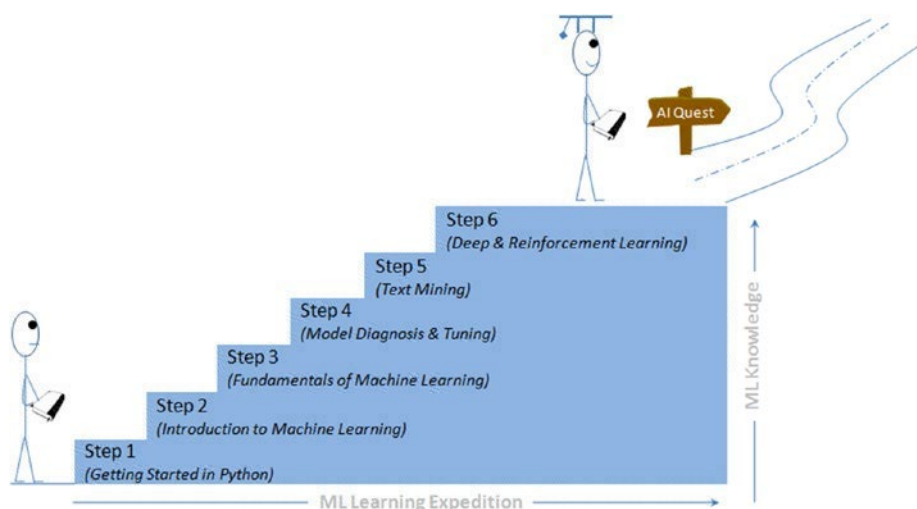
I would like to express my gratitude to my mentors, colleagues, and friends from current/previous organizations for their inputs, inspiration, and support. Thanks to Jojo for the encouragement to write this book and his technical review inputs. Big thanks to the Apress team for their constant support and help.

Finally, I would like to thank you the reader for showing an interest in this book and sincerely hope to help your pursuit to machine learning quest.

Note that the views expressed in this book are author's personal.

# Introduction

This book is your practical guide towards novice to master in machine learning with Python in six steps. The six steps path has been designed based on the “Six degrees of separation” theory that states that everyone and everything is a maximum of six steps away. Note that the theory deals with the quality of connections, rather than their existence. So a great effort has been taken to design eminent, yet simple six steps covering fundamentals to advanced topics gradually that will help a beginner walk his way from no or least knowledge of machine learning in Python to all the way to becoming a master practitioner. This book is also helpful for current Machine Learning practitioners to learn the advanced topics such as Hyperparameter tuning, various ensemble techniques, Natural Language Processing (NLP), deep learning, and the basics of reinforcement learning. See Figure 1.



**Figure 1.** Learning Journey - Mastering Python Machine Learning: In Six Steps

Each topic has two parts: the first part will cover the theoretical concepts and the second part will cover practical implementation with different Python packages. The traditional approach of math to machine learning, that is, learning all the mathematics then understanding how to implement it to solve problems needs a great deal of time/effort, which has proven to be not efficient for working professionals looking to switch careers. Hence the focus in this book has been more on simplification, such that the theory/math behind algorithms have been covered only to the extent required to get you started.



I recommend you work with the book instead of reading it. Real learning goes on only through active participation. Hence, all the code presented in the book is available in the form of iPython notebooks to enable you to try out these examples yourselves and extend them to your advantage or interest as required later.

## Who This Book Is for

This book will serve as a great resource for learning machine learning concepts and implementation techniques for the following:

- Python developers or data engineers looking to expand their knowledge or career into the machine learning area.
- A current non-Python (R, SAS, SPSS, Matlab, or any other language) machine learning practitioners looking to expand their implementation skills in Python.
- Novice machine learning practitioners looking to learn advanced topics such as hyperparameter tuning, various ensemble techniques, Natural Language Processing (NLP), deep learning, and basics of reinforcement learning.

## What You Will Learn

*Chapter 1, Step 1 - Getting started in Python.* This chapter will help you to set up the environment, and introduce you to the key concepts of Python programming language in relevance to machine learning. If you are already well versed with Python basics, I recommend you glance through the chapter quickly and move onto the next chapter.

*Chapter 2, Step 2 - Introduction to Machine Learning.* Here you will learn about the history, evolution, and different frameworks in practice for building machine learning systems. I think this understanding is very important as it will give you a broader perspective and set the stage for your further expedition. You'll understand the different types of machine learning (supervised / unsupervised / reinforcement learning). You will also learn the various concepts are involved in core data analysis packages (NumPy, Pandas, Matplotlib) with example codes.

*Chapter 3, Step 3 - Fundamentals of Machine Learning* This chapter will expose you to various fundamental concepts involved in feature engineering, supervised learning (linear regression, nonlinear regression, logistic regression, time series forecasting and classification algorithms), unsupervised learning (clustering techniques, dimension reduction technique) with the help of scikit-learn and statsmodel packages.

*Chapter 4, Step 4 - Model Diagnosis and Tuning.* in this chapter you'll learn advanced topics around different model diagnosis, which covers the common problems that arise, and various tuning techniques to overcome these issues to build efficient models. The topics include choosing the correct probability cutoff, handling an imbalanced dataset, the variance, and the bias issues. You'll also learn various tuning techniques such as ensemble models and hyperparameter tuning using grid / random search.

*Chapter 5, Step 5 - Text Mining and Recommender System.* Statistics says 70% of the data available in the business world is in the form of text, so text mining has vast scope across various domains. You will learn the building blocks and basic concepts to advanced NLP techniques. You'll also learn the recommender systems that are most commonly used to create personalization for customers.

*Chapter 6, Step 6 - Deep and Reinforcement Learning.* There has been a great advancement in the area of Artificial Neural Network (ANN) through deep learning techniques and it has been the buzzword in recent times. You'll learn various aspects of deep learning such as multilayer perceptrons, Convolution Neural Network (CNN) for image classification, RNN (Recurrent Neural Network) for text classification, and transfer learning. And you'll also learn the q-learning example to understand the concept of reinforcement learning.

*Chapter 7, Conclusion.* This chapter summarizes your six step learning and you'll learn quick tips that you should remember while starting with real-world machine learning problems.

## CHAPTER 1



# Step 1 – Getting Started in Python

In this chapter you will get a high-level overview of the Python language and its core philosophy, how to set up the Python development environment, and the key concepts around Python programming to get you started with basics. This chapter is an additional step or the prerequisite step for non-Python users. If you are already comfortable with Python, I would recommend that you quickly run through the contents to ensure you are aware of all of the key concepts.

## The Best Things in Life Are Free

As the saying goes, “*The best things in life are free!*” Python is an open source, high-level, object-oriented, interpreted, and general-purpose dynamic programming language. It has a community-based development model. Its core design theory accentuates code readability, and its coding structure enables programmers to articulate computing concepts in fewer lines of code as compared to other high-level programming languages such as Java, C or C++.

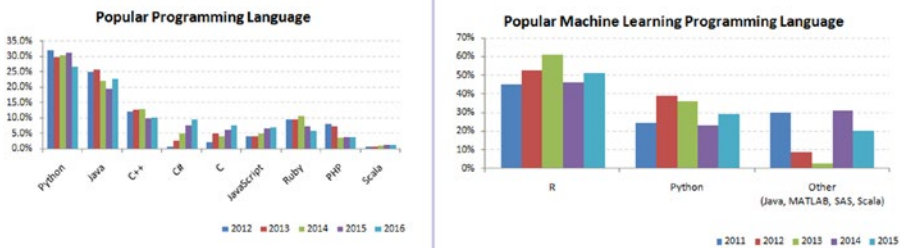
The design philosophy of Python is well summarized by the document “The Zen of Python” (Python Enhancement Proposal, information entry number 20), which includes mottos such as the following:

- Beautiful is better than ugly – be consistent.
- Complex is better than complicated – use existing libraries.
- Simple is better than complex – keep it simple and stupid (KISS).
- Flat is better than nested – avoid nested ifs.
- Explicit is better than implicit – be clear.
- Sparse is better than dense – separate code into modules.
- Readability counts – indenting for easy readability.
- Special cases aren’t special enough to break the rules – everything is an object.
- Errors should never pass silently – good exception handler.

- Although practicality beats purity - if required, break the rules.
- Unless explicitly silenced – error logging and traceability.
- In ambiguity, refuse the temptation to guess – Python syntax is simpler; however, many times we might take a longer time to decipher it.
- Although that way may not be obvious at first unless you're Dutch – there is not only one of way of achieving something.
- There should be preferably only one obvious way to do it – use existing libraries.
- If the implementation is hard to explain, it's a bad idea – if you can't explain in simple terms then you don't understand it well enough.
- Now is better than never – there are quick/dirty ways to get the job done rather than trying too much to optimize.
- Although never is often better than \*right\* now – although there is a quick/dirty way, don't head in the path that will not allow a graceful way back.
- Namespaces are one honking great idea, so let's do more of those! – be specific.
- If the implementation is easy to explain, it may be a good idea – simplicity.

## The Rising Star

Python was officially born on February 20, 1991, with version number 0.9.0 and has taken a tremendous growth path to become the most popular language for the last 5 years in a row (2012 to 2016). Its application cuts across various areas such as website development, mobile apps development, scientific and numeric computing, desktop GUI, and complex software development. Even though Python is a more general-purpose programming and scripting language, it has been gaining popularity over the past 5 years among data scientists and Machine Learning engineers. See Figure 1-1.



**Figure 1-1.** Popular Coding Language(Source: codeeval.com) and Popular Machine Learning Programming Language (Source:KDD poll)

There are well-designed development environments such as IPython Notebook and Spyder that allow for a quick introspection of the data and enable developing of machine learning models interactively.

Powerful modules such as NumPy and Pandas exist for the efficient use of numeric data. Scientific computing is made easy with SciPy package. A number of primary machine learning algorithms have been efficiently implemented in scikit-learn (also known as sklearn). HadoopPy, PySpark provides seamless work experience with big data technology stacks. Cython and Numba modules allow executing Python code in par with the speed of C code. Modules such as nosetest emphasize high-quality, continuous integration tests, and automatic deployment.

Combining all of the above has made many machine learning engineers embrace Python as the choice of language to explore data, identify patterns, and build and deploy models to the production environment. Most importantly the business-friendly licenses for various key Python packages are encouraging the collaboration of businesses and the open source community for the benefit of both worlds. Overall the Python programming ecosystem allows for quick results and happy programmers. We have been seeing the trend of developers being part of the open source community to contribute to the bug fixes and new algorithms for the use by the global community, at the same time protecting the core IP of the respective company they work for.

## Python 2.7.x or Python 3.4.x?

Python 3.4.x is the latest version and comes with nicer, consistent functionalities! However, there is very limited third-party module support for it, and this will be the trend for at least a couple of more years. However, all major frameworks still run on version 2.7.x and are likely to continue to do so for a significant amount of time. Therefore, it is advised to start with Python 2, for the fact that it is the most widely used version for building machine learning systems as of today.

---

For an in-depth analysis of the differences between Python 2 vs. 3, you can refer to Wiki.python.org (<https://wiki.python.org/moin/Python2orPython3v>), which says that there are benefits to each.

---

I recommend Anaconda (Python distribution), which is BSD licensed and gives you permission to use it commercially and for redistribution. It has around 270 packages including the most important ones for most scientific applications, data analysis, and machine learning such as NumPy, SciPy, Pandas, IPython, matplotlib, and scikit-learn. It also provides a superior environment tool conda that allows you to easily switch between environments, even between Python 2 and 3 (if required). It is also updated very quickly as soon as a new version of a package is released and you can just use *conda update <packagename>* to update it.

You can download the latest version of Anaconda from their official website at <https://www.continuum.io/downloads> and follow the installation instructions.

## Windows Installation

- Download the installer depending on your system configuration (32 or 64 bit).
- Double-click the .exe file to install Anaconda and follow the installation wizard on your screen.

## OSX Installation

For Mac OS, you can install either through a graphical installer or from a command line.

### Graphical Installer

- Download the graphical installer.
- Double-click the downloaded .pkg file and follow the installation wizard instructions on your screen.

Or

### Command-Line Installer

- Download the command-line installer.
- In your terminal window type one of the below and follow the instructions: `bash <Anaconda2-x.x.x-MacOSX-x86_64.sh>`.

## Linux Installation

- Download the installer depending on your system configuration (32 or 64 bit).
- In your terminal window type one of the below and follow the instructions: `bash Anaconda2-x.x.x-Linux-x86_xx.sh`.

## Python from Official Website

For some reason if you don't want to go with the Anaconda build pack, alternatively you can go to Python's official website <https://www.python.org/downloads/> and browse to the appropriate OS section and download the installer. Note that OSX and most of the Linux come with preinstalled Python so there is no need of additional configuring.

Setting up PATH for Windows: When you run the installer make sure to check the "Add Python to PATH option." This will allow us to invoke the Python interpreter from any directory.

If you miss ticking "Add Python to PATH option," follow these instructions:

- Right-click on "My computer."
- Click "Properties."
- Click "Advanced system settings" in the side panel.

- Click “Environment Variables.”
- Click the “New” below system variables.
- For the name, enter pythonexe (or anything you want).
- For the value, enter the path to your Python (example: C:\Python32\).
- Now edit the Path variable (in the system part) and add %pythonexe%; to the end of what’s already there.

## Running Python

From the command line, type “Python” to open the interactive interpreter.

A Python script can be executed at the command line using the syntax here:

```
python <scriptname.py>
```

All the code used in this book are available as IPython Notebook (now known as the Jupyter Notebook), it is an interactive computational environment, in which you can combine code execution, rich text, mathematics, plots and rich media. You can launch the Jupyter Notebook by clicking on the icon installed by Anaconda in the start menu (Windows) or by typing ‘jupyter notebook’ in a terminal (cmd on Windows). Then browse for the relevant IPython Notebook file that you would like to paly with.

Note that the codes can break with change is package version, hence for reproducibility, I have shared my package version numbers, please refer Module\_Versions IPython Notebook.

## Key Concepts

There are a couple of fundamental concepts in Python, and understanding these are essential for a beginner to get started. A brief look at these concepts is to follow.

## Python Identifiers

As the name suggests, identifiers help us to differentiate one entity from another. Python entities such as class, functions, and variables are called identifiers.

- It can be a combination of upper- or lowercase letters (a to z or A to Z).
- It can be any digits (0 to 9) or an underscore (\_).
- The general rules to be followed for writing identifiers in Python.
- It cannot start with a digit. For example, 1variable is not valid, whereas variable1 is valid.
- Python reserved keywords (refer to Table 1-1) cannot be used as identifiers.
- Except for underscore (\_), special symbols like !, @, #, \$, % etc cannot be part of the identifiers.

# Keywords

Table 1-1 lists the set of reserved words used in Python to define the syntax and structure of the language. Keywords are case sensitive, and all the keywords are in lowercase except *True*, *False*, and *None*.

**Table 1-1.** *Python keywords*

FALSE	Class	Finally	Is	return
None	Continue	For	Lambda	try
TRUE	Def	From	nonlocal	while
And	Del	Global	Not	with
As	Elif	If	Or	yield
Assert	Else	Import	Pass	
Break	Except	In	Raise	

# My First Python Program

Launch the Python interactive on the command prompt, and then type the following text and press Enter.

```
>>> print "Hello, Python World!"
```

If you are running Python 2.7.x from the Anaconda build pack, then you can also use the print statement with parentheses as in `print ("Hello, Python World!")`, which would produce the following result: Hello, Python World! See Figure 1-2.



**Figure 1-2.** *Python vs. Others*

# Code Blocks (Indentation & Suites)

It is very important to understand how to write code blocks in Python. Let's look at two key concepts around code blocks.



## Indentation

One of the most unique features of Python is its use of indentation to mark blocks of code. Each line of code must be indented by the same amount to denote a block of code in Python. Unlike most other programming languages, indentation is not used to help make the code look pretty. Indentation is required to indicate which block of code a code or statement belongs to.

## Suites

A collection of individual statements that makes a single code block are called suites in Python. A header line followed by a suite are required for compound or complex statements such as *if*, *while*, *def*, and *class* (we will understand each of these in details in the later sections). Header lines begin with a keyword, and terminate with a colon (:) and are followed by one or more lines that make up the suite. See Listings 1-1 and 1-2.

### **Listing 1-1.** Example of correct indentation

```
# Correct indentation
print ("Programming is an important skill for Data Science")
print ("Statistics is a important skill for Data Science")
print ("Business domain knowledge is a important skill for Data Science")

# Correct indentation, note that if statement here is an example of suites
x = 1
if x == 1:
    print ('x has a value of 1')
else:
    print ('x does NOT have a value of 1')
```

### **Listing 1-2.** Example of incorrect indentation

```
# incorrect indentation, program will generate a syntax error
# due to the space character inserted at the beginning of second line
print ("Programming is an important skill for Data Science")
    print ("Statistics is a important skill for Data Science")
print ("Business domain knowledge is a important skill for Data Science")
3
# incorrect indentation, program will generate a syntax error
# due to the wrong indentation in the else statement
x = 1
if x == 1:
    print ('x has a value of 1')
else:
    print ('x does NOT have a value of 1')
```

# Basic Object Types

According to the Python data model reference, objects are Python’s notion for data. All data in a Python program is represented by objects or by relations between objects. In a sense, and in conformance to Von Neumann’s model of a “stored program computer,” code is also represented by objects. Every object has an identity, a type, and a value. See Table 1-2 and Listing 1-3.

**Table 1-2.** *Python object types*

Type	Examples	Comments
None	None	# singleton null object
Boolean	True, False	
Integer	-1, 0, 1, sys.maxint	
Long	1L, 9787L	
Float	3.141592654	
	inf, float('inf')	# infinity
	-inf	# neg infinity
	nan, float('nan')	# not a number
Complex	2+8j	# note use of j
String	'this is a string', "also me"	# use single or double quote
	r'raw string', b'ASCII string'	
	u'unicode string'	
Tuple	empty = ()	# empty tuple
	(1, True, 'ML')	# immutable list or unalterable list
List	empty = []	empty list
	[1, True, 'ML']	# mutable list or alterable list
Set	empty = set()	# empty set
	set(1, True, 'ML')	# mutable or alterable
dictionary	empty = {}	# mutable object or alterable object
	{'1': 'A', '2': 'AA', True = 1, False = 0}	
File	f = open('filename', 'rb')	

**Listing 1-3.** Code For Basic Object Types

```

none = None # singleton null object
boolean = bool(True)
integer = 1
Long = 3.14

# float
Float = 3.14
Float_inf = float('inf')
Float_nan = float('nan')

# complex object type, note the usage of letter j
Complex = 2+8j

# string can be enclosed in single or double quote
string = 'this is a string'
me_also_string = "also me"

List = [1, True, 'ML'] # Values can be changed

Tuple = (1, True, 'ML') # Values can not be changed

Set = set([1,2,2,2,3,4,5,5]) # Duplicates will not be stored

# Use a dictionary when you have a set of unique keys that map to values
Dictionary = {'a':'A', 2:'AA', True:1, False:0}

# lets print the object type and the value
print type(none), none
print type(boolean), boolean
print type(integer), integer
print type(Long), Long
print type(Float), Float
print type(Float_inf), Float_inf
print type(Float_nan), Float_nan
print type(Complex), Complex
print type(string), string
print type(me_also_string), me_also_string
print type(Tuple), Tuple
print type(List), List
print type(Set), Set
print type(Dictionary), Dictionary

----- output -----

<type 'NoneType'> None
<type 'bool'> True

```

```

<type 'int'> 1
<type 'float'> 3.14
<type 'float'> 3.14
<type 'float'> inf
<type 'float'> nan
<type 'complex'> (2+8j)
<type 'str'> this is a string
<type 'str'> also me
<type 'tuple'> (1, True, 'ML')
<type 'list'> [1, True, 'ML']
<type 'set'> set([1, 2, 3, 4, 5])
<type 'dict'> {'a': 'A', True: 1, 2: 'AA', False: 0}

```

## When to Use List vs. Tuples vs. Set vs. Dictionary

- *List*: Use when you need an ordered sequence of homogenous collections, whose values can be changed later in the program.
- *Tuple*: Use when you need an ordered sequence of heterogeneous collections whose values need not be changed later in the program.
- *Set*: It is ideal for use when you don't have to store duplicates and you are not concerned about the order or the items. You just want to know whether a particular value already exists or not.
- *Dictionary*: It is ideal for use when you need to relate values with keys, in order to look them up efficiently using a key.

## Comments in Python

Single line comment: Any characters followed by the # (hash) and up to the end of the line are considered a part of the comment and the Python interpreter ignores them.

Multiline comments: Any characters between the strings `"""` (referred as multiline string), that is, one at the beginning and end of your comments will be ignored by the Python interpreter. See Listing 1-4.

### **Listing 1-4.** Example code for comments

```

# This is a single line comment in Python
print "Hello Python World" # This is also a single line comment in Python

""" This is an example of a multi line
comment that runs into multiple lines.
Everything that is in between is considered as comments
"""

```

## Multiline Statement

Python's oblique line continuation inside parentheses, brackets, and braces is the favorite way of casing longer lines. Using backslash to indicate line continuation makes readability better; however if needed you can add an extra pair of parentheses around the expression. It is important to correctly indent the continued line of your code. Note that the preferred place to break around the binary operator is after the operator, and not before it. See Listing 1-5.

**Listing 1-5.** Example code for multiline statements

```
# Example of implicit line continuation
x = ('1' + '2' +
     '3' + '4')

# Example of explicit line continuation
y = '1' + '2' + \
    '11' + '12'

weekdays = ['Monday', 'Tuesday', 'Wednesday',
             'Thursday', 'Friday']

weekend = {'Saturday',
           'Sunday'}

print ('x has a value of', x)
print ('y has a value of', y)
print days
print weekend

----- output -----
('x has a value of', '1234')
('y has a value of', '1234')
['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday']
set(['Sunday', 'Saturday'])
```

## Multiple Statements on a Single Line

Python also allows multiple statements on a single line through usage of the semicolon (;), given that the statement does not start a new code block. See Listing 1-6.

**Listing 1-6.** Code example for multistatements on a single line

```
import os; x = 'Hello'; print x
```

# Basic Operators

In Python, operators are the special symbols that can manipulate the value of operands. For example, let's consider the expression  $1 + 2 = 3$ . Here, 1 and 2 are called operands, which are the value on which operators operate and the symbol + is called an operator.

Python language supports the following types of operators.

- Arithmetic Operators
- Comparison or Relational Operators
- Assignment Operators
- Bitwise Operators
- Logical Operators
- Membership Operators
- Identity Operators

Let's learn all operators through examples one by one.

## Arithmetic Operators

Arithmetic operators are useful for performing mathematical operations on numbers such as addition, subtraction, multiplication, division, etc. See Table 1-3 and then Listing 1-7.

**Table 1-3.** *Arithmetic operators*

Operator	Description	Example
+	Addition	$x + y = 30$
-	Subtraction	$x - y = -10$
*	Multiplication	$x * y = 200$
/	Division	$y / x = 2$
%	Modulus	$y \% x = 0$
** Exponent	Exponentiation	$x ** b = 10$ to the power 20
//	Floor Division – Integer division rounded toward minus infinity	$9 // 2 = 4$ and $9.0 // 2.0 = 4.0$ , $-11 // 3 = -4$ , $-11.0 /$