Ultimate Python Programming

Learn Python with 650+ programs, 900+ practice questions, and 5 projects

Deepali Srivastava

www.bpbonline.com

First Edition 2024

ISBN: 978-93-55516-558

LIMITS OF LIABILITY AND DISCLAIMER OF WARRANTY

The information contained in this book is true to correct and the best of author's and publisher's knowledge. The author has made every effort to ensure the accuracy of these publications, but publisher cannot be held responsible for any loss or damage arising from any information in this book.

All trademarks referred to in the book are acknowledged as properties of their respective owners but BPB Publications cannot guarantee the accuracy of this information.

Dedicated to

Sri Anjaneya Swamy

About the Author

Deepali Srivastava has a Master's degree in Mathematics and is an author and educator in the field of computer science and programming. Her books "C in Depth" and "Data Structures Through C in Depth" are widely used as reference materials by students, programmers and professionals looking to enhance their understanding of programming languages and data structures. These books are known for their clarity, depth of coverage, and practical approach to learning. In addition to her writing, Deepali Srivastava has been involved in creating online video courses on Data structures and Algorithms, Linux and Python programming. Her books and courses have helped 350,000+ students learn computer science concepts. Her work has been appreciated by students and has been a valuable resource for those looking to build their programming skills.

Acknowledgement

I would like to thank God for blessing me with the opportunity and inspiration to write this book, and for giving me the strength to do it.

I am grateful to my husband Suresh Kumar Srivastava for always believing in my capabilities and consistently inspiring me to give my best. He introduced me to book writing and helped me unleash my potential. His thoughtful suggestions and feedback helped me improve the content and presentation of this book.

I would like to thank my parents, my brother and my sister-in-law for their unwavering love and support. Blessings of my parents and late parents-in-law are a major source of my inner strength.

I am indebted to my teachers in my journey of education, especially my teachers and friends in MJP Rohilkhand University Bareilly, where I got introduced to the world of programming.

I extend my appreciation to the readers of my books and students of my online courses for their interest in my work, and for their appreciation and suggestions. Any sort of feedback is valuable to me and helps me in improving my work and creating better content.

I am grateful to the BPB Publications team for their guidance and support provided during every step of the publishing journey. Special appreciation goes to the editing team, layout team, and all other contributors involved in bringing this book to life.

Preface

Python is a widely used general-purpose programming language. Its popularity can be attributed to its simplicity and a rich set of powerful features. The clean and intuitive syntax makes it an excellent choice for novices, allowing them to grasp the fundamentals of programming quickly, and the advanced features make it appealing to experienced programmers too. It can run on various platforms, including Windows, macOS, and Linux. Since it is an open-source software, it is freely available to all.

The widespread usage of Python is evident in the technology world, with major companies and organizations such as Google, Amazon, Instagram, Facebook, and NASA using it in different ways. Whether you are involved in machine learning, data science, artificial intelligence, scientific computing, automation or you need to create robust web applications and games, Python provides the necessary tools and resources. The extensive collection of libraries available in Python can be effectively utilized across diverse domains. Therefore, adding Python to your skill set can greatly enhance your capabilities and open up numerous opportunities in various fields.

This book provides a thorough and comprehensive introduction to Python, focusing on the core programming concepts and problem-solving skills required for building a solid foundation in programming. Throughout the book, there are numerous programming examples and end-of-chapter exercises to give you a hands-on experience. The exercises include multiple-choice questions and programming problems; multiple-choice questions will assess both your memory and comprehension of the topic, while the programming exercises will provide you with a chance to apply the acquired concepts. The book includes coding conventions and best practices for writing efficient, readable, and maintainable code. The code in the book is written and tested using Python version 3.11, which is the most recent version at the time of publishing the book.

Python is easy to learn. You can start writing Python programs within a few days. However, if you wish to leverage all the powerful features of Python, a more in-depth exploration is required. The content in this book can assist you in achieving that. This book includes 21 chapters that gradually introduce new topics so learners can proceed at a sustainable pace. If you are a beginner, start from the first chapter and go through all the chapters in order, and work out the examples and exercises along the way. If you have a working knowledge of Python, you can quickly browse through the initial chapters and then randomly jump to topics that are new to you or that you want to master. However, I would still recommend reading the chapters in sequence to get the most out of the book. If you are transitioning from some other language, you might be tempted to skip the initial information, but I would suggest you go through all the basic details to avoid any confusion later. Here is a brief summary of the chapters presented in the book.

Chapter 1 is an introduction to Python and shows the installation process. Chapter 2 covers the fundamental elements of Python, such as data types, variables, input, output, and many other basic concepts you need to get started in Python. Chapter 3 provides a detailed explanation of strings that represent textual data in Python. Chapters 4 and 5 cover the container types: lists, tuples, dictionaries, and sets. Chapter 6 provides an insight into conditional execution. In chapter 7, we will see how to perform repetitive tasks using loops, and chapter 8 discusses some common looping techniques in Python. Chapter 9 introduces the concept of comprehensions which help us write shorter and readable code.

Chapter 10 contains detailed coverage of functions. We will see how to create our own functions and will discuss parameters, arguments, arguments passing, function objects, and many other details about functions. Chapter 11 shows how to create and use modules and packages. Chapter 12 is about namespaces and scoping rules. Chapter 13 shows how to write programs that can create files, write data into files, and read the data stored in files. Chapters 14, 15, and 16 provide you with a strong understanding of the object-oriented concepts. We will discuss classes, objects, methods, inheritance, polymorphism, and magic methods. Chapters 17 and 18 are devoted to advanced topics like iterators, generators, and decorators. Chapter 19 is about functional programming and lambda functions. Chapter 20 shows how to handle run-time errors in Python, and Chapter 21 discusses context managers that are used to automate common resource management patterns.

At the end of each chapter, you will find exercises, and their solutions are provided at the end of the book. I would suggest that you try to solve these exercises by yourself before looking at the solution. Solving exercises and writing code will help you to internalize the concepts presented in the book.

Some typographical conventions are followed throughout the book for a good reading experience. The code snippets and programs in the book appear in `this font` to differentiate them from the regular text. Program elements, such as variable names, types, etc., within the regular text, are in `this font`. Any output produced by the code on the screen as a result of running a program or anything that the user has to input through the screen appears in `this font`.

My aim was to write an absolute hands-on book that is simple enough to follow and yet gives detailed knowledge. Reading this book will be a breeze, yet it will give you a comprehensive knowledge of Python and instill the confidence to excel in any written test, interview, or professional work. Programming is fun only when you get your hands dirty with code. Reading a book is not enough for learning programming. I highly recommend that you try the coding examples and exercises presented in the book. The efforts you put in to strengthen your fundamentals of core programming concepts will take you a long way in your software development journey.

By the end of this book, you will develop a strong foundation in core Python skills and will get the ability to explore the vast range of functionalities offered by the standard library and third-party libraries. As you progress, you will continue to be amazed by the capabilities of Python and the remarkable libraries available. With your newfound skills you can venture into diverse fields like data science or machine learning. Moreover, if this is the first programming language you are learning, equipped with the foundation of programming concepts and problem-solving skills, you can easily learn any other programming language.

After using this book as a tutorial to learn the language, you can always refer to it as a handy resource whenever you need to recall or review any concept and apply it to your work.

Writing this book was a very enjoyable, insightful, and amazingly satisfying journey for me and I am sure my readers will have a similar experience while reading the book. I hope you enjoy reading the book and start loving Python.

Happy programming!

Code Bundle and Coloured Images

Please follow the link to download the
Code Bundle and the ***Coloured Images*** of the book:

https://rebrand.ly/z815rfg

The code bundle for the book is also hosted on GitHub at
https://github.com/bpbpublications/Ultimate-Python-Programming.
In case there's an update to the code, it will be updated on the existing GitHub repository.

We have code bundles from our rich catalogue of books and videos available at
https://github.com/bpbpublications. Check them out!

Errata

We take immense pride in our work at BPB Publications and follow best practices to ensure the accuracy of our content to provide with an indulging reading experience to our subscribers. Our readers are our mirrors, and we use their inputs to reflect and improve upon human errors, if any, that may have occurred during the publishing processes involved. To let us maintain the quality and help us reach out to any readers who might be having difficulties due to any unforeseen errors, please write to us at :

errata@bpbonline.com

Your support, suggestions and feedbacks are highly appreciated by the BPB Publications' Family.

Piracy

If you come across any illegal copies of our works in any form on the internet, we would be grateful if you would provide us with the location address or website name. Please contact us at **business@bpbonline.com** with a link to the material.

If you are interested in becoming an author

If there is a topic that you have expertise in, and you are interested in either writing or contributing to a book, please visit **www.bpbonline.com**. We have worked with thousands of developers and tech professionals, just like you, to help them share their insights with the global tech community. You can make a general application, apply for a specific hot topic that we are recruiting an author for, or submit your own idea.

Reviews

Please leave a review. Once you have read and used this book, why not leave a review on the site that you purchased it from? Potential readers can then see and use your unbiased opinion to make purchase decisions. We at BPB can understand what you think about our products, and our authors can see your feedback on their book. Thank you!

For more information about BPB, please visit **www.bpbonline.com**.

Join our book's Discord space

Join the book's Discord Workspace for Latest updates, Offers, Tech happenings around the world, New Release and Sessions with the Authors:

https://discord.bpbonline.com

Table of Contents

Introduction to Python

1

Python is a widely used high-level and general-purpose programming language originally developed by Guido Van Rossum in the early 1990s in the Netherlands. It is maintained by a community of core developers who are actively engaged in its growth and advancement. Although the official logo of Python shows two intertwined snakes, it is not named after any snake. Van Rossum named this language after a 1970s comedy show 'Monty Python's Flying Circus'.

Python has three major versions; the initial version, Python 1.0, was released in January 1994. The second major version, Python 2.0, was released in 2000, and the third major version, Python 3.0, was released in 2008. Python 3 is not backward compatible with Python 2; this means that the code written in Python 2 may not work as expected in Python 3 without making some modifications. In this book, we will use Python 3. The latest release of Python is available on its official website www.python.org. Python is an open-source software, which means that it is free to use and distribute.

1.1 What makes Python so popular

Python is a general-purpose language used in a wide variety of domains. It is used extensively in different fields such as web development, data mining, artificial intelligence, image processing, robotics, network programming, developing user interfaces, database programming, scientific and mathematical computing, game programming, and even education. Most of the top companies and organizations, such as Google, Facebook, Amazon, and NASA, use Python in different ways. Let us see some of the key factors that contribute to Python's popularity.

Python is very easy to learn. It doesn't take much time to become productive with Python. This is why it is often the introductory programming language taught in many universities. Compared to languages such as C++ or Java, Python code tends to be more concise, requiring fewer lines of code to achieve the same functionality. Due to the simple syntax of Python, programmers can focus more on finding the solution to a problem instead of getting caught up in complex language features. Python uses indentation for grouping together statements, resulting in a visually clean layout that enhances code readability.

Python offers a convenient command line interface known as the 'Python interactive shell' or 'Python REPL' (Read-Eval-Print Loop). With the Python interpreter, you have the option to work interactively, allowing you to test and debug small sections of code in real-time. The interactive mode serves as a useful tool for experimenting and exploring Python's features.

One of the main advantages of Python is that it takes care of memory management automatically. Python's built-in memory management system allocates memory when needed and frees it up when it is no longer in use. Programmers do not have to worry about managing memory manually, as they would have to do in other languages like C or C++.

Python includes a vast standard library of modules; this is why the phrase 'Batteries included' is often used for Python. These modules contain code that you can use in your own programs. In addition to the extensive standard library, many third-party libraries are also available for use. Thus, you have access to lots of prewritten reusable code in the form of standard library modules and third-party modules, which can do most of the work for you and save you from reinventing the wheel. This code can be incorporated into your code to develop complex solutions with minimal effort. Whether you are working on web programming, creating graphics, analyzing data, performing mathematical calculations, engaging

in scientific computing, or developing games, you will find reusable code modules that can help you achieve your goals.

Python supports multiple programming paradigms, including procedural, functional, and object-oriented programming. Thus, programmers have the flexibility to choose the coding structure that best suits their needs. The object-oriented features of Python are much easier to implement and are more intuitive when compared to similar features found in other programming languages.

Python is a cross-platform and portable programming language, which means that programs written in Python can be developed and executed on various hardware platforms and operating systems. The same code can be executed on multiple platforms without making any significant changes. The cross-platform development minimizes the efforts required to adapt the programs to different systems and thus facilitates code reuse and sharing on different platforms.

Python has the capability to interact with software components written in other languages. Python code can call libraries written in C and C++, and it can also integrate with components developed in Java and .NET. This allows Python programmers to tap into the strengths and functionalities of other languages and libraries written in them. Python is also embeddable which means that Python code can be placed within the code of another language like C or C++.

Another reason for Python's popularity is its large base of active and supportive developer community. Community members are actively engaged in improving and enhancing the capabilities of Python as well as in developing various libraries and tools. There are numerous resources and extensive support available due to the vibrant community members.

Python has emerged as the preferred programming language for developers because of its ease of use and powerful features. It is suitable both for beginners and experts alike, and due to its versatility, it can be used in a variety of applications.

In the next section we will learn about Python implementations and will see what happens internally when a Python program is executed. While it is not necessary to have this knowledge in order to write and run programs, having a fundamental understanding of what occurs behind the scenes during program execution is beneficial for a comprehensive understanding of the language.

1.2 Python implementation

The terms C, C++, Basic, Java, or Python refer to programming languages, which are essentially sets of rules and specifications. In order to use these languages, they need to be implemented by creating software that allows us to write programs in that language and run them on a computer. The implementation of a language is the program that actually runs the code that you write in that language. An implementation translates the source code to native machine code instructions (binary 0s and 1s) so that the computer's processor can execute it.

There are primarily two approaches to implementing a programming language: compilation and interpretation. In compilation, a compiler translates the complete program code in one go to another language such as machine code or bytecode. If the translated code is machine code that is understood by the processor, then it is directly executed, and if it is bytecode, then it has to be again input to another interpreter or compiler. In interpretation, an interpreter translates the code to machine code one line at a time; a line of code is read, translated, and executed, then the next line is read, translated, and executed, and so on. The code is translated line by line at run time, so the interpreted implementations tend to be slower than the compiled ones, which translate the whole code at once.

An implementation of a language can be a compiler, interpreter, or a combination of both. A programming language can have multiple implementations, and these implementations can be written in different languages and can use different approaches to compile or interpret code. The notion of interpretation

and compilation is associated with language implementation rather than the language itself; describing a language as compiled or interpreted is not technically correct. The language implementations that are written for a language are described as compiled or interpreted and not the language. Compilation or interpretation is not a part of the language specification; it is an implementation decision. The implementations of C and C++ mostly use the compilation approach, while Java, Python, and C# implementations generally use a combination of compilation and interpretation techniques. C and C++ compilers translate source code to machine code, which is executed directly by the processor.

Python has multiple implementations. The original and standard implementation of Python is CPython written in C language. It is the most widely used and up-to-date implementation of Python. When you download Python software from the official site python.org, this is the implementation that you get. The other implementations are Jython written in Java, and IronPython written for the .NET platform. PyPy is the implementation that is written in RPython, which is a subset of Python.

The software that is used for running Python programs is referred to as Python interpreter. Let us understand how CPython interpreter combines the compilation and interpretation techniques to execute a Python program.

We write our Python code in a source file (.py file), but the computer cannot understand and execute this code; it can execute only machine code, which consists of instructions written in binary form (0s and 1s). The source code has to be converted to machine code so that the processor can execute it. The source code is not directly converted to machine code. It is first compiled into an intermediate form known as the bytecode. This bytecode is a low-level code that is Python-specific and platform-independent, but it is not understandable to the processor.

There is another software called Python Virtual Machine (PVM), that is responsible for executing this bytecode on a specific platform. The bytecode passes through the Python Virtual machine; it interprets this bytecode, which means that it converts the bytecode instructions to machine code instructions one by one and sends these machine code instructions to the processor for execution, and we get the output. So, the job of PVM is to convert the bytecode instructions to machine code instructions that the processor can understand and execute.

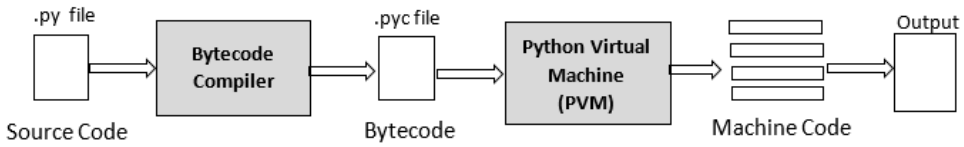

Figure 1.1: *The execution of a Python program*

This is what happens when we execute a Python program. The intermediate compilation step is hidden from the programmer; we can just type and run our program immediately. The programmer does not have to explicitly compile the code, so there is no separate compile time in Python; there is only runtime. The compilation to bytecode is done to improve the efficiency as the bytecode can be interpreted faster than the original source code.

In this whole process, the bytecode complier is a software that converts source code to bytecode, and PVM is a software that converts bytecode to machine code for the target platform. Python Virtual machine contains some platform-specific components that may be implemented differently for each platform. This allows the virtual machine to covert the bytecode into native machine code according to the platform. It abstracts away the underlying hardware and operating system details and thus provides a consistent runtime environment for Python programs across different platforms. Both the bytecode compiler and the virtual machine are part of the Python interpreter software and are included in your Python installation.

The intermediate bytecode is generally cached for faster execution. It is stored in .pyc or .pyo files inside a folder named __pycache__ and the programmer can just ignore these files. When the program is run multiple times without modifying the source code, the compiled bytecode from the cached file is loaded and executed instead of re-compiling from source code to bytecode every time. This bytecode is stored only for imported files, not for the top-level scripts; we will see the difference between the two later in the book.

The Jython implementation translates Python code into Java bytecode, enabling its execution on a Java virtual machine. An advantage of Jython is its ability to directly access Java libraries. Similarly, IronPython is designed for the .NET framework and facilitates integration with .NET components.

Some implementations of virtual machines (bytecode interpreters) use just-in-time (JIT) compilation approach to speed up the interpretation process. The PyPy implementation of Python has better speed as it includes a just-in-time compiler for faster execution of the bytecode. Just-in-time compiler will compile the frequently executed blocks of bytecode to machine code and cache the result. Next time, when the virtual machine has to execute the same block of bytecode, the precompiled(cached) machine code is utilized and executed, resulting in faster execution. So, the JIT compiler uses the compilation approach to improve the efficiency of bytecode execution.

1.3 Installing Python

To download Python, visit the official website of Python. On the homepage, select the Downloads option to go to the download page, or you can directly go to www.python.org/downloads/. The website will automatically detect your operating system and provide a suitable installer that corresponds to your system's requirements, whether it be 32-bit or 64-bit. Click on the Download button to download the installer (.exe) file for the latest version of Python. At the time of writing this book, the latest version is 3.11.3. If you wish to download any previous version of Python, you can scroll down the page and click on the download button located next to the version number you desire.

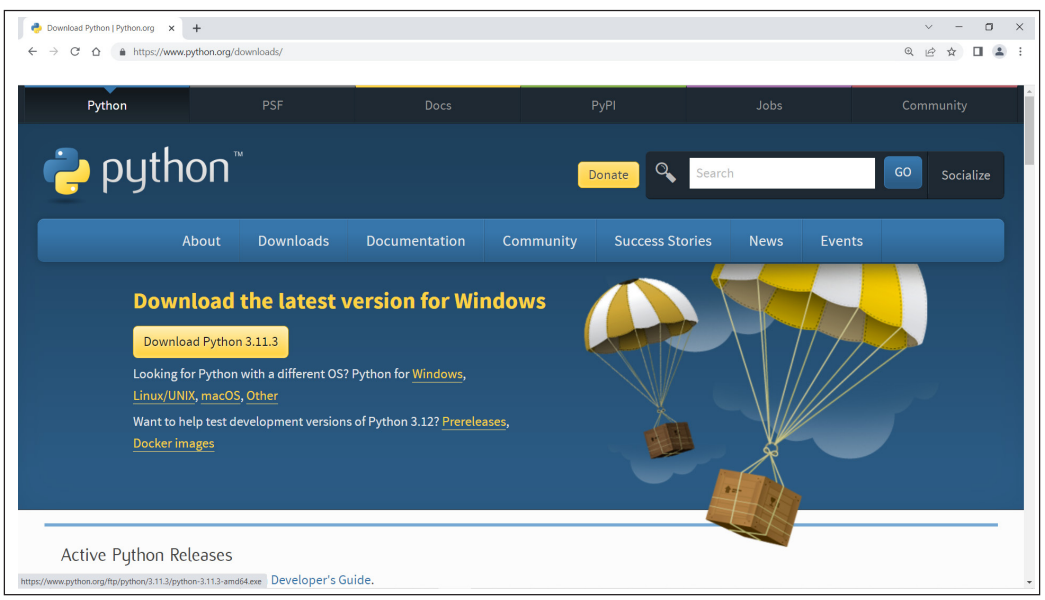

Figure 1.2: *Official website of Python*

Once the download is complete, double-click on the installer to execute it and begin the installation process. On the first screen of the installer, you will be presented with two choices: "**Install Now**" and

"**Customize Installation**." Clicking on "**Install Now**" will install Python with the default features, while clicking on "**Customize Installation**" will allow you to change the installation location or install other optional and advanced features. The defaults should work well for now, so we will go with Install Now. Before clicking on Install Now, make sure to select the **Add python.exe to PATH** checkbox, as this will add Python to your system's PATH environment variable and will enable you to run Python from the command prompt.

Figure 1.3: *Installing Python*

Click **Yes** if it asks for permission to make changes to your device. The installation begins, and all the required Python files, along with the standard library, will be installed on your system.

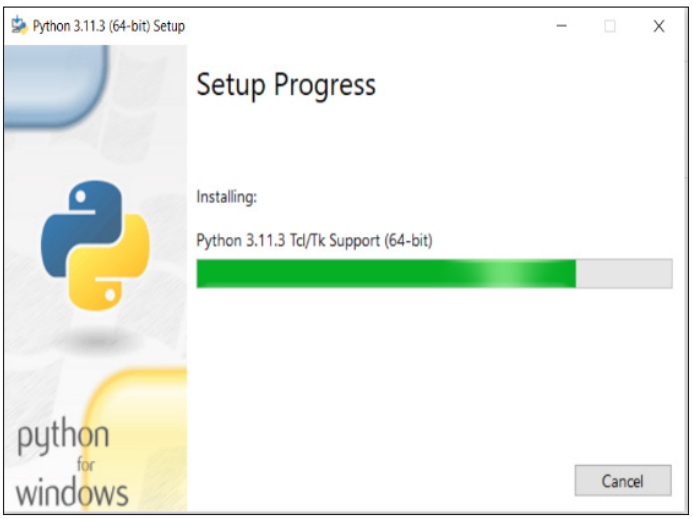

Figure 1.4: *Installation in progress*

After the installation is complete, the following pop-up box will appear. This shows that Python is installed on your system. Click on **Close** to complete the installation and exit the installer. The appearance of the images shown in the screenshots may vary depending on the version of Python that you choose to install.