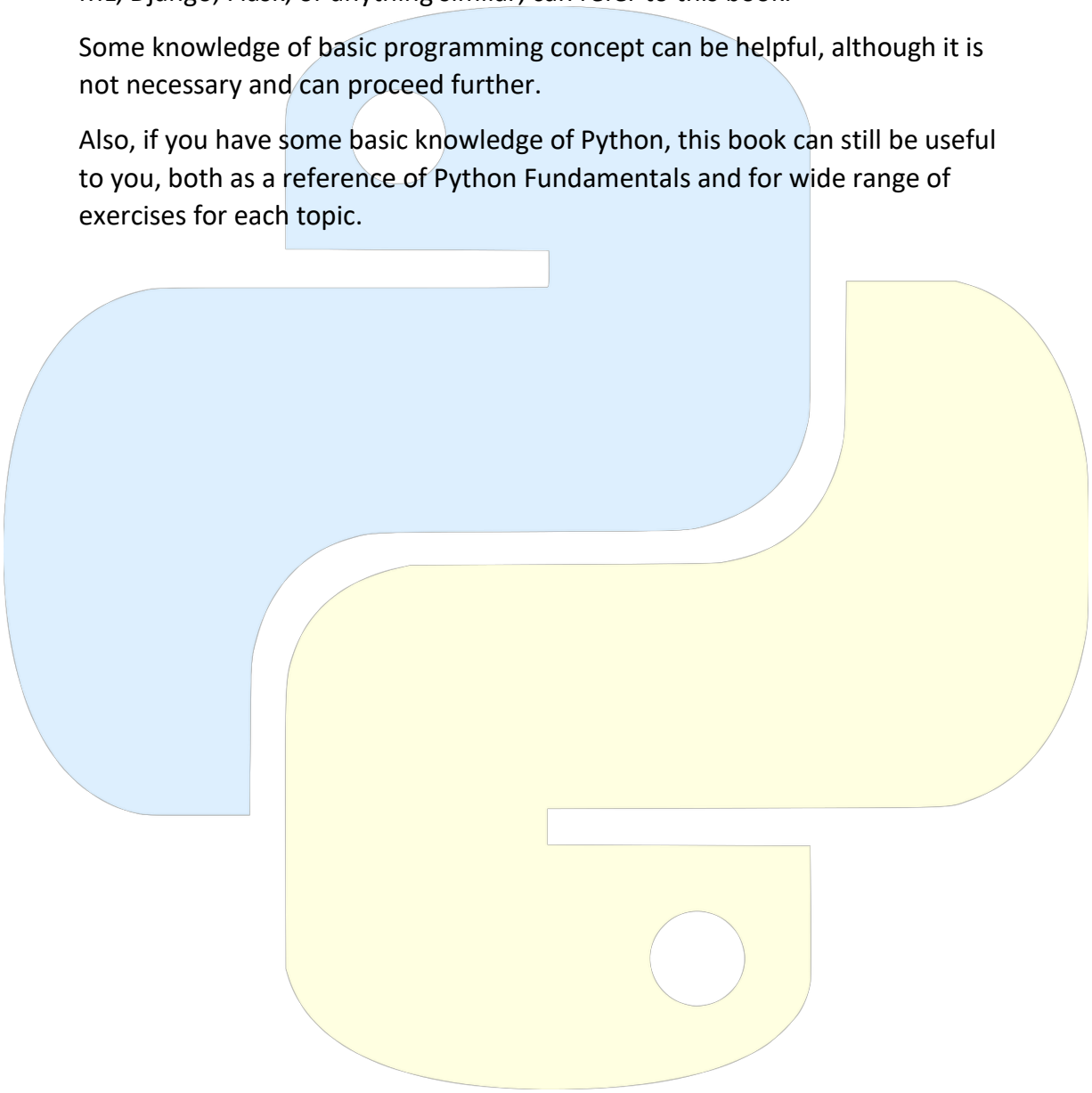
## Who this book is for

This book is for people who want to get started with Python Programming and want to make career in it. Anyone looking for going into the Data Science, AI, ML, Django, Flask, or anything similar, can refer to this book.

Some knowledge of basic programming concept can be helpful, although it is not necessary and can proceed further.

Also, if you have some basic knowledge of Python, this book can still be useful to you, both as a reference of Python Fundamentals and for wide range of exercises for each topic.



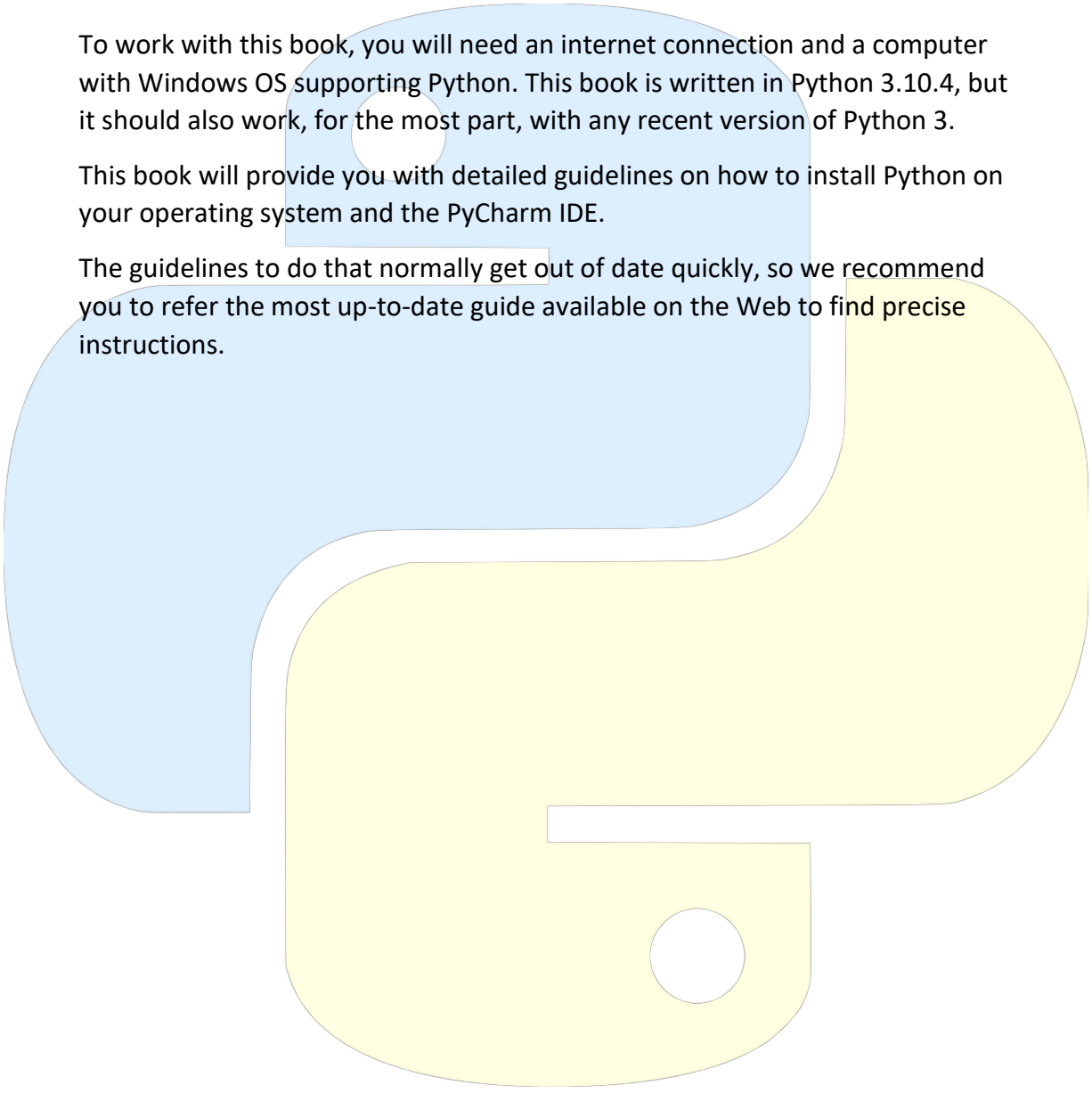
# To get the most out of this book

You are encouraged to read each topic in this book, also after every topic, a set of questions will be provided for the practice purpose along with solutions at the end of this book.

To work with this book, you will need an internet connection and a computer with Windows OS supporting Python. This book is written in Python 3.10.4, but it should also work, for the most part, with any recent version of Python 3.

This book will provide you with detailed guidelines on how to install Python on your operating system and the PyCharm IDE.

The guidelines to do that normally get out of date quickly, so we recommend you to refer the most up-to-date guide available on the Web to find precise instructions.



# Introduction to Python Language

## What is computer programming?

According to Wikipedia, the definition of **computer programming** is:

*"…the process of performing a particular computation (or more generally, accomplishing a specific computing result), usually by designing and building an executable computer program. Programming involves tasks such as analysis, generating algorithms, profiling algorithms' accuracy and resource consumption, and the implementation of algorithms (usually in a chosen programming language, commonly referred to as coding)."*

(https://en.wikipedia.org/wiki/Computer_programming)

To be precise, **computer programming** or **coding,** is sometimes can be referred to as giving set of instructions to a computer in a language that it understands.

A computer is one of the most powerful tools made by Humans, but unfortunately, it cannot think by themselves. They need to be told everything: how to evaluate a condition, how to perform a task, how to handle data, how to read a drive, and more.

You can code in many different styles and languages. The main question comes: **Is it hard?** Well, it depends on what coding language you pick up. It is a bit like writing—it is something that everybody can learn. But what if you want to become a poet? Writing alone is not enough. You must acquire a whole other set of skills, and this will involve a longer and greater effort.

In the end, it all comes down to how far you want to go down the road. Coding is not just putting together some instructions that work. It is so much more!

In the end, it all comes down to how far you want to go down the road. Coding is not just putting together some instructions that work. It is so much more! Good code is short, fast, elegant, easy to read and understand, simple, easy to modify and extend, easy to scale and refactor, and easy to test. It takes time to be able to write code that has all these qualities at the same time, but the good news is that you are taking the first step towards it at this very moment by reading this book. And we have no doubt you can do it. Anyone can; in fact, we all program all the time, only we are not aware of it. Take the following example…

Say you want to make instant coffee. You must get a mug, the instant coffee jar, a teaspoon, water, and the kettle. Even if you are not aware of it, you are evaluating a lot of data. You are making sure that there is water in the kettle and that the kettle is plugged in, that the mug is clean, and that there is enough coffee in the jar. Then, you boil the water and maybe, in the meantime, you put some coffee in the mug. When the water is ready, you pour it into the mug, and stir.

So, how is this programming?

Well, we gathered resources (the kettle, coffee, water, teaspoon, and mug) and we verified some conditions concerning them (the kettle is plugged in, the mug is clean, and there is enough coffee). Then we started two actions (boiling the water and putting coffee in the mug), and when both were completed, we finally ended the procedure by pouring water into the mug and stirring.

## Why was Python created in the first place?

Here is a very brief summary of what started it all, written by Guido van Rossum:

- I had extensive experience with implementing an interpreted language in the ABC group at CWI, and from working with this group I had learned a lot about language design. This is the origin of many Python features, including the use of indentation for statement grouping and the inclusion of very-high-level data types (although the details are all different in Python).
- I had several gripes about the ABC language, but also liked many of its features. It was impossible to extend the ABC language (or its implementation) to remedy my complaints – in fact its lack of extensibility was one of its biggest problems. I had some experience with using Modula-2+ and talked with the designers of Modula-3 and read the Modula-3 report. Modula-3 is the origin of the syntax and semantics used for exceptions, and some other Python features.
- I was working in the Amoeba distributed operating system group at CWI. We needed a better way to do system administration than by writing either C programs or Bourne shell scripts, since Amoeba had its own system call interface which was not easily accessible from the Bourne shell. My experience with error handling in Amoeba made me acutely

aware of the importance of exceptions as a programming language feature.

- It occurred to me that a scripting language with a syntax like ABC but with access to the Amoeba system calls would fill the need. I realized that it would be foolish to write an Amoeba-specific language, so I decided that I needed a language that was generally extensible.

## What is Python good for?

Python is a high-level general-purpose programming language that can be applied to many different classes of problems.

The language comes with a large standard library that covers areas such as string processing (regular expressions, Unicode, calculating differences between files), internet protocols (HTTP, FTP, SMTP, XML-RPC, POP, IMAP, CGI programming), software engineering (unit testing, logging, profiling, parsing Python code), and operating system interfaces (system calls, filesystems, TCP/IP sockets).

## Features of Python

### Easy to Learn

Python is one of the most user-friendly programming languages. One can easily learn the basics of Python and become familiar with its syntax and be able to write basic programs in a few days. However, learning the advanced concepts and mastering python may take you some time. Compared to other languages like C, C++, Java, etc., Python is the easiest language to learn and master.

### Free and Open Source

Python is a free and open-source programming language, which means that it can be used for free of cost on any operating system and without any copyright issues. Anyone can download Python from its official website along with its libraries and documentation. You can download, but it also allows you to make your own modules or libraries and distribute them.

### Portability

Python runs everywhere, and porting a program from Linux to Windows or Mac is usually just a matter of fixing paths and settings. Python is designed for portability and it takes care of specific **operating system (OS)** quirks behind interfaces that shield you from the pain of having to write code tailored to a specific platform.

### Software quality

Python is heavily focused on readability, coherence, and quality. The language's uniformity allows for high readability, and this is crucial nowadays, as coding is more of a collective effort than a solo endeavor. Another important aspect of Python is its intrinsic multiparadigm nature. You can use it as a scripting language, but you can also exploit object oriented, imperative, and functional programming styles—it is extremely versatile.

### Software integration

Another important aspect is that Python can be extended and integrated with many other languages, which means that even when a company is using a different language as their mainstream tool, Python can come in and act as a gluing agent between complex applications that need to talk to each other in some way.

**Databases Support**

Today almost every application we need to develop certainly requires a database and here comes the Python Database API (DB-API) that provides an interface to almost most of the major commercial databases. Some of the databases supported by standard python are MySQL, PostgreSQL, Microsoft SQL, Oracle, Informix, etc. We need to import the interface for the database to use it. Using Python, you can deal with both relational as well as non-relational databases.

**Large Standard Library**

Python consists of a bulk of libraries that are cross-platform and provides a rich set of modules and functions. These libraries are compatible with various operating systems like UNIX, Mac, windows, etc. Due to the large number of libraries, we need not write code for everything instead import and use the functionality required. For example, if you need to access some websites and want to scrape data from them, then you do not need to write the functions for request, response, and other things, from scratch. There are various libraries available for this purpose which you can use.

**Dynamic Typed Language**

Python is a Dynamically-Typed language. Dynamically-typed means that, unlike other programming languages, in Python, we need not declare the data type (for example, int, float, double, char, etc.) of a variable explicitly. The data type of the variable is decided at the run time. Apart from this one variable can be used to store different types of data at different instances in the program. This feature of python saves a lot of time and helps us to avoid pitfalls that might have occurred if it required the datatype to be mentioned explicitly.

## What are the drawbacks?

**Speed:** Probably, the only drawback that one could find in Python, which is not due to personal preferences, is its execution speed. Typically, Python is slower than its compiled siblings. The standard implementation of Python produces, when you run an application, a compiled version of the source code called byte code (with the extension .pyc), which is then run by the Python interpreter. The advantage of this approach is portability, which we pay for with increased runtimes since Python is not compiled down to the machine level, as other languages are.

**Memory Consumption:** Python has a very high memory consumption. This is because it is flexible to the data types. It uses large amounts of memory. Python is not a good choice for tasks where the user wants to optimize memory, i.e., a memory-intensive language.

**Mobile development:** Python is strong in server platforms and desktops, and hence it is a fantastic server-side programming language. But it is not appropriate for mobile development. For mobile development, Python is a fragile language. Since it is not memory efficient and has a prolonged power for processing, due to these reasons, Python does not have many built-in mobile applications. Carbonnelle is a built-in application present in Python.

**Runtime errors:** The users of Python mentioned various issues they faced with the language design. Since the language of Python is dynamically typed, there can be changes in the data type of a variable at any time. Therefore, it needs to be tested more often, and, there are errors in the language displayed during runtime.

**Developer's Restrictions:** Once a developer gets used to the ease and simplicity of this language, it becomes difficult for them to switch back to other languages.

## Who is using Python today?

The Python programming language is widely used by companies around the world to build web apps, analyze data, automate operations via DevOps and create reliable, scalable enterprise applications.

Many companies do not even realize they are using Python across their organizations. For example, if a company is a "Java-only shop" but they use IBM WebSphere as a web application server then they must use Python to script the server's configuration! Python has a habit of getting in everywhere regardless of whether the usage is intentional.

**Uber's tech stack** contains a significant amount of Python, which they documented in a series of engineering posts. Part, one describes the lower backend levels, which are written in Python, with Node.js, Go and Java mixed in. Part two explains the higher levels of the marketplace and user interfaces.

**Twilio** uses Python with Django and the Wagtail content management system to power the amazing Twilio documentation as well as TwilioQuest. They wrote a post about how TwilioQuest was built that goes into detail on the code including the usage of the front-end Vue.js framework. Twilio also uses Flask to run the REST API endpoints and open sourced the Flask-RESTful framework so other developers could cut down the boilerplate in their web APIs.

**Netflix** uses Python throughout their organization to run chaos engineering tests and generally glue together the code from their high-functioning polyglot teams. Netflix also wrote a 2019 update for PyCon US to give more detail on what teams and projects work in Python.

**Google** uses Python extensively and officially supports it internally as one of their three core languages, the other two being Java and Golang. While Google likely has every programming language running somewhere in their infrastructure, Python receives priority support due to its core language status.

Python is used in many different contexts, such as system programming, web and API programming, GUI applications, gaming and robotics, rapid prototyping, system integration, data science, database applications, real time communication, and much more. Several prestigious universities have also adopted Python as their main language in computer science courses.