Chapter 1
Getting Started with Python

This book teaches the Python programming language, which is one of the most popular languages for introductory programming courses. An advantage of Python is that it is a so-called high-level language, with simple and intuitive syntax that makes it easy to get started. However, although it works well as a beginner's language, Python is also suitable for more advanced tasks, and it is currently one of the most widely used programming languages worldwide.

1.1 The First Example: Hello, World!

Most introductory books on programming start with a so-called Hello, World! program, which is a program that simply writes *Hello, World!* to the screen. In Python, this program is just a single line;

```
print("Hello, World!")
```

To actually write and run such a program, Python offers a number of different options. Throughout this book we will mostly apply the classical programming approach, where a program is written in a text editor and stored as a file that is then run from the command line window or an integrated development environment (IDE). To write and run the "Hello, World!"-program above, open your favorite editor (Atom, gedit, Emacs etc.), type the given line and save the file with a suitable filename, for instance, `hello.py`.[1] Then, open a terminal or an iPython window, navigate to the directory where you saved the file, and type `python hello.py`, if you are using a regular terminal, or `run hello.py` if you are using iPython. The output *Hello, World!* should

[1] We do not describe the technical details of acquiring and installing Python here, since this information is platform dependent and becomes outdated very quickly. For updated hints on installing Python, see the web page for the IN1900 course at the University of Oslo (`https://www.uio.no/studier/emner/matnat/ifi/IN1900/index-eng.html`), or to the numerous other resources found online.

J. Sundnes, *Introduction to Scientific Programming with Python*, Simula SpringerBriefs on Computing 6,
https://doi.org/10.1007/978-3-030-50356-7_1

appear in the terminal right after the command. If you are using an IDE, it is essentially an editor and an iPython/terminal window combined. For instance, in the popular Spyder IDE the editor is usually in the upper left window, while the window in the lower right corner is the iPython window where you run the program. [2]

Although the "Hello, World!"-program could seem like a silly example, it serves a number of useful purposes. First of all, running this small program will verify that you have installed Python properly, and that you have installed the right version. It also introduces the function `print`, which will be used virtually every time we program, and it illustrates how we use quotes to define a *string* in Python. While `print` is a word that Python understands, the words "Hello" and "World" are not. By using the quotes, we tell Python that it should not try to understand (or interpret) these words, but, rather, treat them as simple text that, in this case, is to be printed to the screen. We will come back to this topic in more detail later.

1.2 Different Ways to Use Python

As briefly mentioned above, Python offers some alternatives to the traditional style of programming using a text editor and a terminal window, and some of these alternatives can be very useful when learning to program. For instance, we can use Python interactively by simply typing `python` or `ipython` in a terminal window, without a subsequent file name. This will open an environment for typing and running Python commands, which is not very suitable for writing programs over several lines, but extremely useful for testing Python commands and statements, or simply using Python as a calculator. In a regular terminal window on macOS or Linux, an interactive version of the Hello, World! example would look something like

Terminal

```
Terminal> ipython
Python 3.7.3 (default, Mar 27 2019, 16:54:48)
Type 'copyright', 'credits' or 'license' for more information
IPython 7.4.0 -- An enhanced Interactive Python.

In [1]: print("Hello, World!")
Hello, World!

In [2]:
```

The two versions `python` and `ipython` work largely the same way, but `ipython` has a number of additional features and is recommended.

[2] For details, see, for instance, `https://www.spyder-ide.org/`.

A third way to use Python is through *Jupyter notebooks*, which are a form of interactive notebooks that combine code and text. The notebooks are viewed through a browser and look quite similar to a simple web page, but with the important difference that the code segments are "live" Python code that can be run, changed, and re-run while reading the document. These features are particularly useful for teaching purposes, since detailed explanations of new concepts are easily combined with interactive examples. All the chapters of this book are also available as Jupyter notebooks.

Minor drawbacks of the Python language. Although Python is a very popular and suitable language for learning to program, it also has some minor drawbacks. One of the more important is tightly linked to its advantage of being a flexible high-level language with a short and intuitive syntax. Writing small programs in Python can be very efficient, and beginners can quickly start writing useful programs, but the downside is that the code can become messy as the programs grow in size and complexity. Other languages such as C, C++, and Java tend, to enforce more structure in the code, which can be confusing for beginners and annoying when you want to write a small program quickly, but it can be more efficient in the long run when writing larger programs. However, it is certainly possible to write neat and nicely structured programs in Python as well, but this requires a choice by the programmer to follow certain principles of coding style, and is not enforced by the language itself.

Another slightly annoying aspect of Python is that it exists in different versions. At the time of this writing, Python 3 has been dominant for quite a while, but if you look for programming resources online or read older textbooks, you will find many examples using Python 2. For the mathematics-centered programming covered in this book, the difference between Python 2 and Python 3 is actually quite small, but some differences are important to be aware of. The most obvious one is how `print` works. In Python 2, the program above would read `print "Hello, World!"`, that is, without the parentheses. Since nearly all code examples use `print` to some extent, programs written in Python 2 will typically not run in Python 3. One particularly relevant resource for scientific Python (on which this book is largely based) is "A Primer on Scientific Programming with Python", by Hans Petter Langtangen[3]. However, the latest version of that book was written in 2016, and all the code examples are in Python 2 and will stop with an error message if they are run in Python 3. In most cases, the only error is the missing parentheses; so the addition of parentheses to all the print statements will make most of the examples run fine in Python 3. We will comment on some of the other differences between the Python versions later.

[3]Hans Petter Langtangen, *A Primer on Scientific Programming with Python*, 5th edition, Springer-Verlag, 2016.