Introduction

What Is This Book About?

Modern biology has been undergoing a dramatic revolution in recent decades. Enormous amounts of data are produced at an unprecedented rate. These data come in various forms, such as gene and protein expression level data, DNA, RNA, and protein sequencing, high quality biological and medical images. One consequence of this "data explosion" is that computational methods are increasingly being used in life science research. Computational methods in this context are not the mere use of tools, but the integration of computational and algorithmic thinking to lab-experiment design; to data generation, integration, and analyses; and to modeling and simulation. It is becoming widely recognized that such thinking skills should be incorporated into the standard training of life scientists in the 21st century.

This book presents some fundamental ideas and notions from computer science in a biological context. Each chapter covers a biologically motivated challenge, and presents the relevant computational notions in a "user-friendly", non-formal manner. Each topic is accompanied by implementations in Python. This programming language has gained considerable popularity among biologists in recent years, as reflected by the proliferation of "Python for biologists" courses at university level around the world. This book also provides hands-on programming practice in the form of exercises throughout and at the end of each chapter.

Who Is This Book Aimed At?

This book is designed specifically for life scientists – students, teachers, researchers, and lab assistants – who wish to acquaint themselves with the computational "culture," including (but not limited to) basic programming skills.

This book assumes basic knowledge in biology. Any introductory course in biology should suffice. Additionally, no background in computer science or programming is assumed, as basic programming is introduced in chapter 1.

We note that this book does not cover any specific bioinformatics tool handling content. Such tools are covered in dedicated guides. Instead, our main goal is to develop fundamental computational thinking skills, which are still absent from many biology education programs.

What Skills Should I Expect to Gain from This Book?

You should be able to:

- identify problems for which a manual solution is not viable, or reaching such a solution is very tedious and error prone, yet such problems are amenable to a computational solution
- implement simple solutions to some biologically motivated problems using the programming language Python
- understand various fundamental notions in computer science and their applicability to the life sciences
- more effectively communicate and collaborate with experienced programmers, computer scientists, and computational biologists

What Topics Are Covered?

The book is divided into five parts. Part I starts with an introduction to programming in Python (chapter 1), which includes most of the programming background needed for later chapters. The important notion of computational complexity as a measure for the efficiency of algorithms is then presented (chapter 2). Part II deals with problems that involve biological sequences (e.g., DNA, RNA, proteins), and familiarizes the reader with two related notions in computing: hashing (chapter 3) and regular expressions (chapter 4). In Part III, we focus on biological networks, and introduce some basic notions in the important field of graph theory (chapter 5), provide an example for a graph-based algorithm – breadth-first-search (chapter 6), and showcase a simulation methodology for regulatory networks (chapter 7). Part IV deals with the representation and processing of biological images (chapters 8 and 9). Finally, part V exposes some fundamental limitations of computing – such as the P vs. NP open problem and the inability to solve many well-defined problems – and offers some deeper concepts from the theory of computer science.

Do You Have a Website?

Yes, at www.cs.tau.ac.il/~amirr/book.

On the book's website, you will find all the Python code demonstrated in the book, as well as some solutions to exercises, notes, and additional recommended resources.

Notes for the Reader

Practical experience is essential for the acquisition of programming skills. Therefore, we advise having a Python session (more details below) open alongside reading the book. We encourage you to download the code from our website, and "play" with it while reading: for example, execute it with various inputs, change something and inspect the effect, and try to solve a problem in a different way. You may find yourself struggling with the code to make it work properly, which can be frustrating. However, we think this is not a waste of time – it is a necessary trial-and-error-based learning process.

Each chapter includes various exercises, of two types: short questions embedded in the chapter, and "challenge yourself" problems at the end of the chapter that explore specific aspects in more depth. Together these should provide you with "hands-on" practice, as well as encourage and develop your computational thinking.

The parts of this book are mostly independent. Therefore, you may choose to focus on specific topics, without having to read all the previous chapters (with the exception of chapter 1 that provides the programming background for the rest of the book and chapter 2 that introduces the notion of complexity).

Notes for the Teacher

Biology teachers and lecturers who are interested in incorporating some computational content into their courses may find some ideas in this book. From our experience, the whole content of this book fits within a 13-week semester at university. If you wish to devote only a small number of hours for that in your course, just pick a single part in the book (or even chapter) that is most appropriate to your audience in terms of background and pedagogical interests. We kept “backwards dependencies” minimal, and you should be able to pick any part without having to cover previous ones for background.

The problems presented at the end of each chapter (“challenge yourself”) may be used as a basis for small projects, which explore the topic even deeper.

Python Preliminaries

Computer programs are sequences of commands that follow a very strict and formal set of rules (syntax). A specific syntax defines what we call a programming language, such as Python, Java, or C. Programming languages are also termed “high-level” languages, because their execution involves, as a first step, their transformation into a language that a computer can understand and execute, termed “low-level” or machine language (this is basically a sequence of 0s and 1s, corresponding to two separable physical states, e.g., voltage). In Python, this transformation is done in a process called “interpretation” (in languages such as Java or C the process is somewhat different and is termed “compilation”). Interpretation is done by special software called an *interpreter*. There are several interpreters for Python which you can download. Possibly the simplest one is called IDLE, the standard Python interpreter. On the course website, you can find links and download instructions.

Executing commands in Python can be done in two modes: *interactive* mode, in which a single command is written, executed, and immediately returns the result to the user, and then the user is prompted for the next command, and so on; and *script* mode, in which a program is written in an editor, stored in a file with a .py extension (e.g., my_program.py), and then executed as a batch of commands one by one. The former mode is convenient for the execution of short and specific computations, while the latter is suitable for writing programs that we intend to execute multiple times.

In this book, we will use the following format conventions regarding these two modes. Commands in interactive mode are presented in a shell-like format, in which lines start with a prompt sign >>>, and their results appear in the next line or lines:

```
>>> print(3+4)
7
```

Programs written in script mode will be presented in an editor-like format, with lines numbered so we can refer to specific lines in the text:

```
1 def max(x, y):
2     if x>y:
3         return x
4     else:
5         return y
```

The execution of such scripts will be presented as interactive mode executions:

```
>>> max(8, 3)
8
>>> max(2, 4+1)
5
```

We follow the default syntactic color highlighting of IDLE.

We hope this book will expose to you the beauty of the computational “culture,” by introducing a diverse range of computational concepts that are applicable to different biological settings.

Finally, we wish to deeply thank Smadar Gazit for her support in writing and organizing this book, always with a smile and good will !

Cheers,
Benny and Amir

https://doi.org/10.1017/9781108178327.001 Published online by Cambridge University Press