View on GitHub

abstractions-in-python

Programming Abstractions in Python

Luca de Alfaro

Introduction

This is the online textbook Programming Abstractions in Python, written for the class by the same name at the University of California, Santa Cruz.

The idea of the book, and of the class, consists in teaching students to think about software in terms of objects that have primitive operations. When confronted with a new problem to solve, one should consider what are the objects that are useful for representing the problem, and what operations can be defined on those objects that solve the problem. The operations on objects are often best understood as mathematical operations on the set of objects: considering the properties of the operations leads to cleaner, and more general, solutions.

The book is articulated into five parts:

- **Methods:** A first part on methods provides a refresher of the Python programming language, and provides an introduction to recursion and generators, topics that are occasionally missing from introductory Python classes.
- **Structures:** A part on structures shows how solve many interesting problems by designing the appropriate objects and operations. We build a clone of Pandas, we introduce data structures for representing subsets of real numbers, we develop a motion detection algorithm for images, and more.
- **From Symbolic Expressions to Machine Learning:** We introduce the idea of representing expressions via trees, and evaluating them with respect to a variable assignment. We then develop symbolic differentiation, autogradient, and we build a simplified clone of PyTorch.
- **Graphs and dynamic algorithms:** A part on graphs dynamic algorithms introduces graphs, along with the fundamental algorithsm for graph exploration. We then present some ideas about dynamic solution methods that are fundamental to problem solving in many areas of computer science.
- **Search:** We introduce the fundamental guess-propagate-recur paradigm that is at the core of most search procedures, and we illustrate it on Sudoku and SAT.

In the course of the book, we will provide simple implementations of well-known problems. Among other things, we will reimplement the machine-learning framework PyTorch, albeit in a simplified pure-Python setting; we will reimplement the core of Pandas, and we will write Sudoku and Boolean SAT solvers. All these implementations will be concise, minimalistic, and yet powerful, and we hope that they can serve as illustration of how to approach non-trivial software and algorithmic problems.

The book chapters consist in Jupyter notebooks. Jupyter notebooks allow literate programming: one can write, in the same place and properly interleaved, both the text and mathematics that explains the high-level concepts, and their implementations into code. The notebooks provided here form the book. In these notebooks, exercises appear in two forms: as suggestions, and as holes where code is missing, for students to fill in. Separately, upon request, are available also teacher versions of the notebooks, where all the code holes have been filled, so that the whole notebook can be run during a classroom lecture. To request these notebooks, email the author at dealfaro@alumni.stanford.edu.

The author is much indebted to Philippe Bossut for inspiration about the approach, and to Massimo Di Pierro for his many suggestions on topics to cover and methods.

Chapters

We provide each chapter as a link to a notebook hosted in Google Colab. For this to work, you might have to visit the Colab home page prior to clicking on the links here.