Preface

Computer Science is a creative, challenging, and rewarding discipline. Computer programmers, sometimes called software engineers, solve problems involving data: computing, moving, and handling large quantities of data are all tasks made easier or possible by computer programs. Money magazine ranked software engineer as the number one job in America in terms of flexibility, creativity, low stress levels, ease of entry, compensation, and job growth within the field [4].

Learning to program a computer is a skill that can bring you great enjoyment because of the creativity involved in designing and implementing a solution to a problem. Python is a good first language to learn because there is very little overhead in learning to write simple programs. Python also has many libraries available that make it easy to write some very interesting programs including programs in the areas of Computer Graphics and Graphical User Interfaces: two topics that are covered in this text.

In this text, students are taught to program by giving them many examples and practice exercises with solutions that they can work on in an interactive classroom environment. The interaction can be accomplished using a computer or using pen and paper. By making the classroom experience active, students reflect on and apply what they have read and heard in the classroom. By using a skill or concept right away, students quickly discover if they need more reinforcement of the concept, while teachers also get immediate feedback. There is a big difference between seeing a concept demonstrated and using it yourself and this text encourages applying concepts immediately to test understanding. This is vital in Computer Science since new skills and concepts build on what we have already learned.

In several places within this book there are examples presented that highlight patterns of programming. These patterns appear over and over in programs we write. In this text, patterns like the *Accumulator Pattern* and the *Guess and Check Pattern* are presented and exercises reinforce the recognition and application of these and other abstract patterns used in problem-solving. Learning a language is certainly one important goal of an introductory text, but acquiring the necessary

problem-solving skills is even more important. Students learn to solve problems on their own by recognizing when certain patterns are relevant and then applying these patterns in their own programs.

Recent studies in Computer Science Education indicate the use of a debugger can greatly enhance a student's understanding of programming [1]. A debugger is a tool that lets the programmer inspect the state of a program at any point while it is executing. There is something about actually seeing what is happening as a program is executed that helps make an abstract concept more concrete. This text introduces students to the use of a debugger and includes exercises and examples that show students how to use a debugger to discover how programs work.

There are additional resources available for instructors teaching from this text. They include lecture slides and a sample schedule of lectures for a semester long course. Solutions to all programming exercises are also available upon request. Visit http://cs.luther.edu/~leekent/CS1 for more information.

Python is a good language for teaching introductory Computer Science because it is very accessible and can be incrementally taught so students can start to write programs before having to learn the whole language. However, at the same time, Python is also a developing language. Python 3.1 was recently released to the public. This release of Python included many performance enhancements which were very good additions to the language. There were also some language issues with version 2.6 and earlier that were cleaned up at the same time that were not backwards compatible. The result is that not all Python 2 programs are compatible with Python 3 and vice versa. Because both Python 2 and Python 3 are in use today, this text will point out the differences between the two versions where appropriate. These differences will be described by inset boxes titled **Python 2⇝3** within the text where the differences are first encountered.

It is recommended that students reading this text use Python 3.1 or later for writing and running their programs. All Python programs presented in the text are Python 3 programs. The libraries used in this text all work with Python 3. However, there may be some libraries that have not been ported to Python 3 that a particular instructor would like to use. In terms of what is covered in this text, the differences between Python 2 and 3 are pretty minor and either language implementation will work to use with the text.

Acknowledgments

I would like to thank Nathaniel Lee, who not only let his dad teach him, but was a great sounding board and test subject for this text. Thank you, Nathan, for all your valuable feedback and for your willingness to learn. I'd also like to thank my wife, Denise, for her ongoing support while I have written. Thanks Denise. I know it has been work for you too.

Credits

At times in this text Microsoft Windows is referred to when installing software. Windows is a registered trademark of Microsoft Corporation in the United States and other countries. Mac OS X is referred to at times within this text. Mac and Mac OS are trademarks of Apple Inc., registered in the U.S. and other countries.

This book also introduces readers to Wing IDE 101, which is used in examples throughout the text. Wing IDE 101 is a free simplified edition of Wing IDE Professional, a full-featured integrated development environment designed specifically for Python. For more information on Wing IDE, see www.wingware.com. Wingware and Wing IDE are trademarks or registered trademarks of Wingware in the United States and other countries.

Suggestions

I welcome suggestions for future printings of this text. If you like this text and have suggestions for future printings, please write up your suggestion(s) and email them to me. The more complete your write up, the more likely I will be to consider your suggestion. If I select your suggestion for a future printing I'll be sure to include your name in the preface as a contributor to the text. Suggestions can be emailed to kentdlee@luther.edu or kentdlee@gmail.com.

http://www.springer.com/978-1-4471-6641-2

Python Programming Fundamentals

Lee, K.D.

2014, XII, 239 p. 64 illus., 53 illus. in color., Softcover
ISBN: 978-1-4471-6641-2