



Table of Contents

[Copyright](#)

[Brief Table of Contents](#)

[Table of Contents](#)

[Preface](#)

[Acknowledgments](#)

[About this Book](#)

[About the author](#)

[Unit 0. Learning how to program](#)

[Lesson 1. Why should you learn how to program?](#)

[1.1. Why programming matters](#)

[1.1.1. Programming isn't just for professionals](#)

[1.1.2. Improve your life](#)

[1.1.3. Challenge yourself](#)

[1.2. Where you are now and where you'll be](#)

[1.3. Our plan for learning how to program](#)

[1.3.1. First steps](#)

[1.3.2. Practice, practice, practice, practice](#)

[1.3.3. Think like a programmer](#)

[Summary](#)

[Lesson 2. Basic principles of learning a programming language](#)

[2.1. Programming as a skill](#)

[2.2. A parallel with baking](#)

[2.2.1. Understand the task "bake a loaf of bread"](#)

[2.2.2. Find a recipe](#)

[2.2.3. Visualize the recipe with flowcharts](#)

[2.2.4. Use an existing recipe or make one up?](#)

[2.3. Think, code, test, debug, repeat](#)

[2.3.1. Understanding the task](#)

[2.3.2. Visualizing the task](#)

[2.3.3. Writing pseudocode](#)

[2.4. Writing readable code](#)

[2.4.1. Using descriptive and meaningful names](#)

[2.4.2. Commenting your code](#)

[Summary](#)

[Unit 1. Variables, types, expressions, and statements](#)

[Lesson 3. Introducing Python: a programming language](#)

[3.1.1. What is Python?](#)[3.1.2. Downloading Python version 3.5](#)[3.1.3. Anaconda Python Distribution](#)[3.1.4. Integrated development environments](#)[3.2. Setting up your workspace](#)[3.2.1. The IPython console](#)[3.2.2. The file editor](#)[Summary](#)[Lesson 4. Variables and expressions: giving names and values to things](#)[4.1. Giving names to things](#)[4.1.1. Math vs. programming](#)[4.1.2. What the computer can and can't do](#)[4.2. Introducing variables](#)[4.2.1. Objects are things that can be manipulated](#)[4.2.2. Objects have names](#)[4.2.3. What object names are allowed?](#)[4.2.4. Creating a variable](#)[4.2.5. Updating a variable](#)[Summary](#)[Lesson 5. Object types and statements of code](#)[5.1. Types of things](#)[5.2. Basic type of objects in programming](#)[5.2.1. Integers as whole numbers](#)[5.2.2. Floating point as decimal numbers](#)[5.2.3. Booleans as true/false data](#)[5.2.4. Strings as sequences of characters](#)[5.2.5. The absence of a value](#)[5.3. Working with basic types of data values](#)[5.3.1. Building blocks of expressions](#)[5.3.2. Converting between different types](#)[5.3.3. How arithmetic impacts object types](#)[Summary](#)[Lesson 6. Capstone project: your first Python program—convert hours to minutes](#)[6.1. Think-code-test-debug-repeat](#)[6.2. Divide your task](#)[Code to set up the input](#)[Code to set up the output](#)[6.3. Implement the conversion formula](#)[6.3.1. How many hours?](#)[6.3.2. How many minutes?](#)[6.4. Your first Python program: one solution](#)[6.5. Your first Python program: another solution](#)[Summary](#)[Unit 2. Strings, tuples, and interacting with the user](#)[Lesson 7. Introducing string objects: sequences of characters](#)[7.1. Strings as sequences of characters](#)[7.2. Basic operations on strings](#)

[7.2.2. Understanding indexing into a string](#)[7.2.3. Understanding slicing a string](#)[7.3. Other operations on string objects](#)[7.3.1. Getting the number of characters in a string with len\(\)](#)[7.3.2. Converting between letter cases with upper\(\) and lower\(\)](#)[Summary](#)[Lesson 8. Advanced string operations](#)[8.1. Operations related to substrings](#)[8.1.1. Find a specific substring in a string with find\(\)](#)[8.1.2. Find out whether a substring is in the string with “in”](#)[8.1.3. Count the number of times a substring occurs with count\(\)](#)[8.1.4. Replace substrings with replace\(\)](#)[8.2. Mathematical operations](#)[Summary](#)[Lesson 9. Simple error messages](#)[9.1. Typing up statements and trying things out](#)[9.2. Understanding string error messages](#)[Summary](#)[Lesson 10. Tuple objects: sequences of any kind of object](#)[10.1. Tuples as sequences of data](#)[10.1.1. Creating tuple objects](#)[10.2. Understanding operations on tuples](#)[10.2.1. Getting the tuple length with len\(\)](#)[10.2.2. Indexing into and slicing a tuple with \[\]](#)[10.2.3. Performing mathematical operations](#)[10.2.4. Swapping objects inside tuples](#)[Summary](#)[Lesson 11. Interacting with the user](#)[11.1. Showing output](#)[11.1.1. Printing expressions](#)[11.1.2. Printing multiple objects](#)[11.2. Getting user input](#)[11.2.1. Prompting the user](#)[11.2.2. Reading the input](#)[11.2.3. Storing the input in a variable](#)[11.2.4. Converting the user input to a different type](#)[11.2.5. Asking for more input](#)[Summary](#)[Lesson 12. Capstone project: name mashup](#)[12.1. Understanding the problem statement](#)[12.1.1. Drawing a sketch of the problem](#)[12.1.2. Coming up with a few examples](#)[12.1.3. Abstracting the problem into pseudocode](#)[12.2. Splitting up first and last names](#)[12.2.1. Finding the space between the first and last name](#)[12.2.2. Using variables to save calculated values](#)[12.2.3. Testing what you know so far](#)

[12.3.1. Finding the midpoint of names](#)

[12.4. Combining the halves](#)

[Summary](#)

[Unit 3. Making decisions in your programs](#)

[Lesson 13. Introducing decisions in programs](#)

[13.1. Making decisions with conditionals](#)

[13.1.1. Yes/no questions and true/false statements](#)

[13.1.2. Adding a condition to a statement](#)

[13.2. Writing the code to make the decision](#)

[13.2.1. Coding up a decision—an example](#)

[13.2.2. Coding up a decision—a general way](#)

[13.3. Structuring your programs](#)

[13.3.1. Making many decisions](#)

[13.3.2. Making decisions based on another decision's outcomes](#)

[13.3.3. A more complicated example with nested conditionals](#)

[Summary](#)

[Lesson 14. Making more-complicated decisions](#)

[14.1. Combining multiple conditions](#)

[14.1.1. Conditionals are made up of true/false expressions](#)

[14.1.2. Operator precedence rules](#)

[14.2. Choosing which lines to execute](#)

[14.2.1. Do this or that](#)

[14.2.2. Putting it all together](#)

[14.2.3. Thinking in terms of code blocks](#)

[Summary](#)

[Lesson 15. Capstone project: choose your own adventure](#)

[15.1. Outlining the game rules](#)

[15.2. Creating different paths](#)

[15.3. More choices? Yes, please!](#)

[Summary](#)

[Unit 4. Repeating tasks](#)

[Lesson 16. Repeating tasks with loops](#)

[16.1. Repeating a task](#)

[16.1.1. Adding nonlinearity to programs](#)

[16.1.2. Infinite repetitions](#)

[16.2. Looping a certain number of times](#)

[16.2.1. for loops](#)

[16.3. Looping N times](#)

[16.3.1. Loops over the common sequence 0 to N – 1](#)

[16.3.2. Unrolling loops](#)

[Summary](#)

[Lesson 17. Customizing loops](#)

[17.1. Customizing loops](#)

[17.2. Looping over strings](#)

[Summary](#)

[Lesson 18. Repeating tasks while conditions hold](#)

[18.1. Looping while a condition is true](#)

[18.1.2. while loops](#)

[18.1.3. Infinite loop](#)

[18.2. Using for loops vs. while loops](#)

[18.3. Manipulating loops](#)

[18.3.1. Exiting early out of a loop](#)

[18.3.2. Going to the beginning of a loop](#)

[Summary](#)

[Lesson 19. Capstone project: Scrabble, Art Edition](#)

[19.1. Understanding the problem statement](#)

[19.1.1. Change the representation of all valid words](#)

[19.1.2. Making a valid word with the given tiles](#)

[19.2. Dividing your code into pieces](#)

[Summary](#)

[Unit 5. Organizing your code into reusable blocks](#)

[Lesson 20. Building programs to last](#)

[20.1. Breaking a big task into smaller tasks](#)

[20.1.1. Ordering an item online](#)

[20.1.2. Understanding the main points](#)

[20.2. Introducing black boxes of code in programming](#)

[20.2.1. Using code modules](#)

[20.2.2. Abstracting code](#)

[20.2.3. Reusing code](#)

[20.3. Subtasks exist in their own environments](#)

[Summary](#)

[Lesson 21. Achieving modularity and abstraction with functions](#)

[21.1. Writing a function](#)

[21.1.1. Function basics: what the function takes in](#)

[21.1.2. Function basics: what the function does](#)

[21.1.3. Function basics: what the function returns](#)

[21.2. Using functions](#)

[21.2.1. Returning more than one value](#)

[21.2.2. Functions without a return statement](#)

[21.3. Documenting your functions](#)

[Summary](#)

[Lesson 22. Advanced operations with functions](#)

[22.1. Thinking about functions with two hats](#)

[22.1.1. Writer hat](#)

[22.1.2. User hat](#)

[22.2. Function scope](#)

[22.2.1. Simple scoping example](#)

[22.2.2. Scoping rules](#)

[22.3. Nesting functions](#)

[22.4. Passing functions as parameters](#)

[22.5. Returning a function](#)

[22.6. Summary](#)

[Lesson 23. Capstone project: analyze your friends](#)

[23.1. Reading a file](#)

[23.1.1. File format](#)

[23.1.3. Remove the newline character](#)

[23.1.4. Using tuples to store information](#)

[23.1.5. What to return](#)

[23.2. Sanitizing user inputs](#)

[23.3. Testing and debugging what you have so far](#)

[23.3.1. File objects](#)

[23.3.2. Writing a text file with names and phone numbers](#)

[23.3.3. Opening files for reading](#)

[23.4. Reusing functions](#)

[23.5. Analyzing the information](#)

[23.5.1. The specification](#)

[23.5.2. Helper functions](#)

[Summary](#)

[Unit 6. Working with mutable data types](#)

[Lesson 24. Mutable and immutable objects](#)

[24.1. Immutable objects](#)

[24.2. The need for mutability](#)

[Summary](#)

[Lesson 25. Working with lists](#)

[25.1. Lists vs. tuples](#)

[25.2. Creating lists and getting elements at specific positions](#)

[25.3. Counting and getting positions of elements](#)

[25.4. Adding items to lists: append, insert, and extend](#)

[25.4.1. Using append](#)

[25.4.2. Using insert](#)

[25.4.3. Using extend](#)

[25.5. Removing items from a list: pop](#)

[25.6. Changing an element value](#)

[Summary](#)

[Lesson 26. Advanced operations with lists](#)

[26.1. Sorting and reversing lists](#)

[26.2. Lists of lists](#)

[26.3. Converting a string to a list](#)

[26.4. Applications of lists](#)

[26.4.1. Stacks](#)

[26.4.2. Queues](#)

[Summary](#)

[Lesson 27. Dictionaries as maps between objects](#)

[27.1. Creating dictionaries, keys, and values](#)

[27.2. Adding key-value pairs to a dictionary](#)

[27.2.1. Short diversion into restrictions on keys](#)

[27.3. Removing key-value pairs from a dictionary](#)

[27.4. Getting all the keys and values in a dictionary](#)

[27.4.1. No ordering to dictionary pairs](#)

[27.5. Why should you use a dictionary?](#)

[27.5.1. Keeping count with frequency dictionaries](#)

[27.5.2. Building unconventional dictionaries](#)

[Summary](#)

28.1. Using object aliases28.1.1. Aliases of immutable objects28.1.2. Aliases of mutable objects28.1.3. Mutable objects as function parameters28.2. Making copies of mutable objects28.2.1. Commands to copy mutable objects28.2.2. Getting copies of sorted lists28.2.3. A word of caution when iterating over mutable objects28.2.4. Why does aliasing exist?SummaryLesson 29. Capstone project: document similarity29.1. Breaking the problem into tasks29.2. Reading file information29.3. Saving all words from the file29.4. Mapping words to their frequency29.5. Comparing two documents by using a similarity score29.6. Putting it all together29.7. One possible extensionSummaryUnit 7. Making your own object types by using object-oriented programmingLesson 30. Making your own object types30.1. Why do you need new object types?30.2. What makes up an object?30.2.1. Object properties30.2.2. Object behaviors30.3. Using dot notationSummaryLesson 31. Creating a class for an object type31.1. Implementing a new object type by using a class31.2. Data attributes as object properties31.2.1. Initializing an object with `__init__`31.2.2. Creating an object property inside `__init__`31.3. Methods as object operations and behaviors31.4. Using an object type you defined31.5. Creating a class with parameters in `__init__`31.6. Dot notation on the class name, not on an objectSummaryLesson 32. Working with your own object types32.1. Defining a stack object32.1.1. Choosing data attributes32.1.2. Implementing methods32.2. Using a Stack object32.2.1. Make a stack of pancakes32.2.2. Make a stack of circlesSummaryLesson 33. Customizing classes33.1. Overriding a special method

[33.3. Behind the scenes](#)

[33.4. What can you do with classes?](#)

[33.4.1. Scheduling events](#)

[Summary](#)

[Lesson 34. Capstone project: card game](#)

[34.1. Using classes that already exist](#)

[34.2. Detailing the game rules](#)

[34.3. Defining the Player class](#)

[34.4. Defining the CardDeck class](#)

[34.5. Simulate the card game](#)

[34.5.1. Setting up the objects](#)

[34.5.2. Simulating rounds in the game](#)

[34.6. Modularity and abstraction with classes](#)

[Summary](#)

[Unit 8. Using libraries to enhance your programs](#)

[Lesson 35. Useful libraries](#)

[35.1. Importing libraries](#)

[35.2. Doing mathematical operations with the math library](#)

[35.3. Random numbers with the random library](#)

[35.3.1. Randomizing lists](#)

[35.3.2. Simulating games of chance](#)

[35.3.3. Replicating results by using a seed](#)

[35.4. Timing programs with the time library](#)

[35.4.1. Using the clock](#)

[35.4.2. Pausing your program](#)

[Summary](#)

[Lesson 36. Testing and debugging your programs](#)

[36.1. Working with the unittest library](#)

[36.2. Separating the program from the tests](#)

[36.2.1. Types of tests](#)

[36.3. Debugging your code](#)

[36.3.1. Using tools to help you step through code](#)

[Summary](#)

[Lesson 37. A library for graphical user interfaces](#)

[37.1. A library for graphical user interfaces](#)

[37.2. Setting up a program using the tkinter library](#)

[37.3. Adding widgets](#)

[37.4. Adding event handlers](#)

[Summary](#)

[Lesson 38. Capstone project: game of tag](#)

[38.1. Identifying the parts to the problem](#)

[38.2. Creating two shapes in a window](#)

[38.3. Moving shapes inside the canvas](#)

[38.4. Detecting a collision between shapes](#)

[38.5. Possible extensions](#)

[Summary](#)

[A. Answers to lesson exercises](#)

[Lesson 2](#)

Lesson 3

Answers to quick checks

Lesson 4

Answers to quick checks

Answers to summary questions

Lesson 5

Answers to quick checks

Lesson 6

Answers to quick checks

Answers to summary questions

Lesson 7

Answers to quick checks

Answers to summary questions

Lesson 8

Answers to quick checks

Answers to summary questions

Lesson 9

Answers to summary questions

Lesson 10

Answers to quick checks

Answers to summary questions

Lesson 11

Answers to quick checks

Answers to summary questions

Lesson 13

Answers to quick checks

Answers to summary questions

Lesson 14

Answers to quick checks

Answers to summary questions

Lesson 16

Answers to quick checks

Answers to summary questions

Lesson 17

Answers to quick checks

Answers to summary questions

Lesson 18

Answers to quick checks

Answers to summary questions

Lesson 20

Answers to quick checks

Answers to summary questions

Lesson 21

Answers to quick checks

Answers to summary questions

Lesson 22

Answers to quick checks

Answers to summary questions

[Answers to quick checks](#)[Answers to summary questions](#)

[Lesson 25](#)

[Answers to quick checks](#)[Answers to summary questions](#)

[Lesson 26](#)

[Answers to quick checks](#)[Answers to summary questions](#)

[Lesson 27](#)

[Answers to quick checks](#)[Answers to summary questions](#)

[Lesson 28](#)

[Answers to quick checks](#)[Answers to summary questions](#)

[Lesson 30](#)

[Answers to quick checks](#)

[Lesson 31](#)

[Answers to quick checks](#)[Answers to summary questions](#)

[Lesson 32](#)

[Answers to quick checks](#)[Answers to summary questions](#)

[Lesson 33](#)

[Answers to quick checks](#)[Answers to summary questions](#)

[Lesson 35](#)

[Answers to quick checks](#)[Answers to summary questions](#)

[Lesson 36](#)

[Answers to quick checks](#)[Answers to summary questions](#)

[Lesson 37](#)

[Answers to quick checks](#)[Answer to summary questions](#)

[B. Python cheat sheet](#)

[Variable names](#)[Mutable vs. immutable](#)[Dictionaries](#)

[C. Interesting Python libraries](#)

[Thinking like a programmer: big ideas](#)[Index](#)[List of Figures](#)[List of Tables](#)[List of Listings](#)

