PYTHON PROGRAMMING (EDIT): AN INTRODUCTION TO COMPUTER SCIENCE Free

- John Zelle,Michael Smith
- 432 pages
- 07 May 2010
- Franklin, Beedle & Associates Inc
- 9781590282410
- English
- Wilsonville, United States

One good way to learn about Python is to use an interac- tive shell for experimentation. Python includes statements to do things such as print output to the screen, get input from the user, calculate the value of a mathematical expression, and perform a sequence of statements multiple times loop. The models of many real-world phenomena exhibit chaotic behavior, which places some limits on the power of computing. Exercises 17 1. Computer science is the study of computers.

Secondary memory is also called RAM. All information that a computer is currently working on is stored in main memory. The syntax of a language is its meaning, and semantics is its form. A programming environment refers to a place where programmers work. A variable is used to give a name to a value so it can be referred to in other places. A loop is used to skip over a section of a program. Multiple Choice 1. What is the fundamental question of computer science? A problem is intractable when a you cannot reverse its solution b it involves tractors c it has many solutions d it is not practical to solve 4.

Which of the following is not an example of secondary memory? Computers and Programs 5. Computer languages designed to be used and understood by humans are a natural languages b high-level computer languages c machine languages d fetch-execute languages 6. A statement is a a translation of machine language b a complete computer command c a precise description of a problem d a section of an algorithm 7. One difference between a compiler and an interpreter is a a compiler is a program b a compiler is used to translate high-level language into machine language c a compiler is no longer needed after a program is translated d a compiler processes source code 8.

By convention, the statements of a program are often placed in a function called a import b main c program d IDLE 9. Which of the following is not true of comments? Compare and contrast the following pairs of concepts from the chapter: a Hardware vs. Software b Algorithm vs. Program c Programming Language vs. Natural Language d High-Level Language vs. Machine Language e Interpreter vs. Compiler f Syntax vs. Semantics 2. Exercises 19 3. Write a detailed algorithm for making a peanut butter and jelly sandwich or some other everyday activity. You should assume that you are talking to someone who is conceptually able to do the task, but has never actually done it before. For example, you might be telling a young child. As you will learn in a later chapter, many of the numbers stored in a computer are not ex- act values, but rather close approximations.

For example, the value 0. Usually, such small differences are not a problem; however, given what you have learned about chaotic behavior in Chapter 1, you should realize the need for caution in certain situations. Can you think of examples where this might be a problem? Trace through the Chaos program from Section 1. Show the sequence of output that results. Programming Exercises 1. Start up an interactive Python session and try typing in each of the following commands. Write down the results you see. Enter and run the Chaos program from Section 1.

Try it out with various values of input to see that it functions as described in the chapter. Modify the Chaos program using 2. Write a short paragraph describing any differences that you notice in the behavior of the two versions. Computers and Programs 4. Modify the Chaos program so that it prints out 20 values instead of Modify the Chaos program so that the number of values to print is determined by the user. The calculation performed in the chaos program can be written in a number of ways that are algebraically equivalent. Write a version of the chaos program for each of the following ways of doing the computation. Hint: see discussion question number 4, above. Advanced Modify the Chaos program so that it accepts two inputs and then prints a table with two columns similar to the one shown in Section 1.

Note: You will probably not be able to get the columns to line up as nicely as those in the example. Computers are very literal, and they must be told what to do right down to the last detail. Writing large programs is a daunting challenge. It would be almost impossible without a systematic approach. The process of creating a program is often broken down into stages according to the information that is produced in each phase. Try to understand as much as possible about it. Until you really know what the problem is, you cannot begin to solve it. Determine Specifications Describe exactly what your program will do. At this point, you should not worry about how your program will work, but rather about deciding exactly what it will accomplish. For simple programs this involves carefully describing what the inputs and outputs of the program will be and how they relate to each other. This is where the how of the program gets worked out.

Implement the Design Translate the design into a computer language and put it into the com- puter. In this book, we will be implementing our algorithms as Python programs. Susan is spending a year studying in Germany. Susan listens to the weather report each morning, but the temperatures are given in degrees Celsius, and she is used to Fahrenheit. Fortunately, Susan has an idea to solve the problem. Being a computer science major, she never goes anywhere without her laptop computer. She thinks it might be possible that a computer program could help her out. Susan begins with an analysis of her problem. In this case, the problem is pretty clear: the radio announcer gives temperatures in degrees Celsius, but Susan only comprehends temperatures that are in degrees Fahrenheit. What should the input be? She decides that her program will allow her to type in the temperature in degrees Celsius.

And the output? The program will display the temperature converted into degrees Fahrenheit. Now she needs to specify the exact relationship of the output to the input. She knows that 0 degrees Celsius freezing is equal to 32 degrees Fahrenheit, and Celsius boiling is equal to Fahrenheit. Example Program: Temperature Converter 23 Notice that this describes one of many possible programs that could solve this problem.

This would be a much more ambitious project, to say the least! Susan is now ready to design an algorithm for her problem. Her program will prompt the user for some input information the Celsius temperature , process it to convert to a Fahrenheit temperature, and then output the result by displaying it on the computer screen. Susan could write her algorithm down in a computer language. Instead, she writes her algorithm using pseudocode. Pseudocode is just precise English that describes what a program does. It is meant to communicate algorithms without all the extra mental overhead of getting the details right in any particular programming language.

This is straightforward, as each line of the algorithm turns into a corresponding line of Python code. They will be discussed in detail in the next section. After completing her program, Susan tests it to see how well it works. She uses inputs for which she knows the correct answers. Writing Simple Programs What is the Celsius temperature? What is the Celsius temperature? You can see that Susan used the values of 0 and to test her program.

She is especially pleased that no debugging seems necessary which is very unusual. Before doing that, though, you need a more complete grounding in the fundamentals of Python. The next few sections will discuss technical details that are essential to writing correct programs. This material can seem a bit tedious, but you will have to master these basics before plunging into more interesting waters. We give names to modules e. Variables are used to give names to values e. Technically, all these names are called identifiers. For the most part, programmers are free to choose any name that conforms to these rules. Good programmers always try to choose names that describe the thing being named. The complete list of Python keywords is shown in Table 2. Elements of Programs 25 False class finally is return None continue for lambda try True def from nonlocal while and del global not with as elif if or yield assert else import pass break except in raise Table 2.

So far, we have seen two different kinds of data in our example pro- grams: numbers and text. For now, you just need to keep in mind that all data has to be stored on the computer in some digital format, and different types of data are stored in different ways. The fragments of program code that produce or calculate new data values are called expressions. The simplest kind of expression is a literal. In chaos. The convert. These are all examples of numeric literals, and their meaning is obvious: 32 represents, well, 32 the number Our programs also manipulated textual data in some simple ways. Computer scientists refer to textual data as strings.

You can think of a string as just a sequence of printable characters. A string literal is indicated in Python by enclosing the characters in quotation marks '"'. These literals produce strings containing the quoted characters. Note that the quotes themselves are not part of the string. They are just the mechanism to tell Python to create a string. The process of turning an expression into an underlying data type is called evaluation. When you type an expression into a Python shell, the shell evaluates the expression and prints out a textual representation of the result. This is a way of letting us know that the value is actually text, not a number or other data type. In the last interaction, we see that the expression "32" produces a string, not a number. Writing Simple Programs In the second line of interaction, we are asking Python to evaluate the expression x.

In response, the Python shell prints out 5, which is the value that was just assigned to x. Of course, we get the same result when we explicitly ask Python to print x using a print statement. The last interaction shows what happens when we try to use a variable that has not been assigned a value. This says that there is no value with that name. The important lesson here is that a variable must always be assigned a value before it can be used in an expression. More complex and interesting expressions can be constructed by combining simpler expressions with operators. For numbers, Python provides the normal set of mathematical operations: addition, subtraction, multiplication, division, and exponentiation. Here are some examples of complex expressions from chaos. You should have little trouble constructing complex expressions in your own programs. Do keep in mind that only the round parentheses are allowed in numeric expressions, but you can nest them if necessary to create expressions like this.

So far, we have looked at a few exam- ples, but I have not yet explained the print function in detail. Like all programming languages, Python has a precise set of rules for the syntax form and semantics meaning of each statement. Computer scientists have developed sophisticated notations called meta-languages for describing programming languages. In this book we will rely on a simple template notation to illustrate the syntax of various statements. Since print is a built-in function, a print statement has the same general form as any other function invocation. We type the function name print followed by parameters listed in parenthe- ses.

The name inside the brackets indicates what is missing; expr stands for an expression. As far as semantics are concerned, a print statement displays information in textual form. Any supplied expressions are evaluated left to right, and the resulting values are displayed on a line of output in a left-to-right fashion. By default, a single blank space character is placed between the displayed values. Writing Simple Programs The answer is 7 The last statement illustrates how string literal expressions are often used in print statements as a convenient way of labeling output.

Notice that successive print statements normally display on separate lines of the screen. A bare print no parameters produces a blank line of output. We can modify that behavior by including an additional parameter that explicitly overrides this default. This is done using a special syntax for named or keyword parameters. Notice in the template I have shown its default value, the end-of-line character.

This is a standard way of showing what value a keyword parameter will have when it is not explicitly given some other value. One common use of the end parameter in print statements is to allow multiple prints to build up a single line of output. The output from the second statement appears immediately following the space. The semantics of the assignment is that the expression on the right side is evaluated to produce a value, which is then associated with the variable named on the left side. It always retains the value of the most recent assign- ment.

In this case I simply added one to the previous value. The chaos. When the variable changes, the old value is erased and a new one written in. Figure 2. This is exactly the way assignment works in some computer languages. In Python, values may end up anywhere in memory, and variables are used to refer to them. Writing Simple Programs is x. An arrow is used to show which value a variable refers to. The effect is like moving the sticky note from one object to another. When a value is no longer referred to by any variable, it is no longer useful. Python will automatically clear these values out of memory so that the space can be used for new values. In fact, this process of automatic memory management is actually called garbage collection.

Some programming languages have a special statement to do this. In Python, input is accomplished using an assignment statement combined with a built-in function called input. The exact form of an input statement depends on what type of data you are trying to get from the user. When Python encounters a call to input, it prints the prompt on the screen. Whatever the user types is then stored as a string. In this example, I typed John Yaya. Evaluating name gives back the string of characters that I typed. So, for example, the string "32" becomes the number If you are reading the example programs carefully, you probably noticed the blank space inside the quotes at the end of all these prompts.

I usually put a space at the end of a prompt so that the input that the user types does not start right next to the prompt. Putting a space in makes the interaction easier to read and understand. In fact, any valid expression would be just as acceptable. When printed, we see that ans got the value 23 as expected. In a sense, the input-eval combination is like a delayed expression. The difference is that the expression was supplied by the user at the time the statement was executed instead of being determined when the statement was written by the programmer. Thus, the user can supply formulas for a program to evaluate. Writing Simple Programs 2. Semantically, this tells Python to evaluate all the expressions on the right-

hand side and then assign these values to the corresponding variables named on the left-hand side.

That is, you want the value currently stored in x to be in y and the value that is currently in y to be stored in x. We can trace the execution of these statements step-by-step to see why. Suppose x and y start with the values 2 and 4. When we then assign x to y in the second step, we just end up with two copies of the original y value. One way to make the swap work is to introduce an additional variable that temporarily remem- bers the original value of x. In Python, the simul- taneous assignment statement offers an elegant alternative. Simultaneous assignment can also be used to get multiple numbers from the user in a single input. Consider this program for averaging exam scores: avg2.

Suppose the user types 86, This example used just two values, but it could be generalized to any number of inputs. Of course, we could have just gotten the input from the user using separate input statements. Writing Simple Programs In some ways this may be better, as the separate prompts are more informative for the user. In this example the decision as to which approach to take is largely a matter of taste.

Just remember that the multiple values trick will not work for string non- evaled input; when the user types a comma it will be just another character in the input string. The comma only becomes a separator when the string is subsequently evaluated. The simplest kind of loop is called a definite loop. That is, at the point in the program when the loop begins, Python knows how many times to go around or iterate the body of the loop. For example, the chaos program in Chapter 1 used a loop that always executed exactly ten times. The variable after the keyword for is called the loop index. It takes on each successive value in the sequence, and the statements in the body are executed once for each value. Often the sequence portion consists of a list of values.

Lists are a very important concept in Python, and you will learn more about them in upcoming chapters. The body of the loop is executed using each successive value in the list. The length of the list determines the number of times the loop executes. The expression range 10 produces the sequence of numbers 0 through 9. The loop using range 10 is equivalent to one using a list of those numbers. If you think about it, you will see that the value of the expression determines the number of items in the resulting sequence. We just needed a sequence length of 10 to make the body execute 10 times. When you want to do something in your program a certain number of times, use a for loop with a suitable range. Writing Simple Programs counted loops. Otherwise you might accidentally wipe out a value that you will need later.

Usually we think of computers as executing a series of instructions in strict sequence. Introducing a loop causes Python to go back and do some statements over and over again. State- ments like the for loop are called control structures because they control the execution of other parts of the program. When Python gets to the loop heading, it checks to see if there are any more items left in the sequence. If the answer is yes, the loop index variable is assigned the next item in the sequence, and then the loop body is executed.

Once the body is complete, the program goes back to the loop heading and checks for another value in the sequence. The loop quits when there are no more items, and the program moves on to the statements that come after the loop. Example Program: Future Value 37 2. We want to develop a program to determine the future value of an investment. You know that money deposited in a bank account earns interest, and this interest accumulates as the years pass. How much will an account be worth ten years from now?

Obviously, it depends on how much money we start with the principal and how much interest the account earns. Given the principal and the interest rate, a program should be able to calculate the value of the investment ten years into the future. Remember, this is a de- scription of what the program will do. What exactly should the inputs be? We need the user to enter the initial amount to invest, the principal. We will also need some indication of how much interest the account earns. This depends both on the interest rate and how often the interest is compounded. One simple way of handling this is to have the user enter an annual percentage rate. Whatever the actual interest rate and compounding frequency, the annual rate tells us how much the investment accrues in one year.

There are a num- ber of reasonable choices. Output The value of the investment 10 years into the future. This formula needs to be applied 10 times. Next we design an algorithm for the program. In any case, this design illustrates that sometimes an algorithmic approach to a calculation can make the mathematics easier. Knowing how to calculate the interest for just one year allows us to calculate any number of years into the future. Each line of the algorithm translates into a statement of Python. Print an introduction print statement, Section 2.

If you have any questions, you should go back and review the relevant descriptions. Notice especially the counted loop pattern is used to apply the interest formula 10 times. That about wraps it up. Here is the completed program: futval. Problem Analysis: Studying the problem to be solved. Design: Writing an algorithm in pseudocode.

Jun 26, Mohammad Elsheimy rated it it was amazing Shelves: programming. Lots of examples and sample projects. Feb 23, Jeremy Anifacc rated it really liked it. Jun 02, Bhakta Kishor rated it really liked it Shelves: study , coding. This book is a very good first programming book for beginners. On reading the title, my first cynical reaction was, "Python as an introduction to computer science? Why not RPG or Snobol? For that matter, what does computer science as rigorously construed have to do with the languages used in modern softwar This book is a very good first programming book for beginners.

For that matter, what does computer science as rigorously construed have to do with the languages used in modern software engineering? Computer programmers are evermore dealing less and less in computer science: We're becoming tailors using a set of patterns preordained by the language designer, patterns to which we cut our fabric all day. Like fish, we hardly notice the water in which we swim, live, and breathe.

Oct 17, Will rated it really liked it. I really enjoyed this book as an introduction to Computer Science. It is not meant to teach Python as much as it is to introduce concepts that can be applied to any language you may decide to pursue. The use of Python is advantageous because of its relative ease for new programmers and its intuitive syntax. After learning these basics, I have had a much easier time dipping my toes into other languages. A fantastic read if you want to learn the basics of Computer Science as it is written in an un I really enjoyed this book as an introduction to Computer Science. A fantastic read if you want to learn the basics of Computer Science as it is written in an understandable, friendly style and also

includes challenging exercises. Dec 02, Barry rated it did not like it. If you think that this book can teach you to program with Python, you are going to be disappointed.

This book teaches about computer science with Python, not how to program with Python. I think John Zelle either needs to start a new line of work or go back to the computer programming classroom, learn how to program with Python, and write a real instruction book on how to program with Python because he is another mediocre amateur.

I found this book very useful. I took Python as an introductory programming course at college. This book was a great resource for homework assignments and projects. Instructions and code were very clear and concise, and I enjoyed making my own way through the book, playing with Python outside of class time. Aug 09, Steve rated it liked it. This is the required textbook for the first two introductory courses to computer science at Dixie State College of Utah. Aug 09, Christopher Eckman rated it liked it Shelves: programming. This could actually be a really good book for a person just learning to program. It is not really a book for someone who is an intermediate programmer or better. Nov 23, Amy rated it really liked it. Helped me out for the start of Uni :. Jul 11, Hamilton Carvalho rated it it was amazing.

Best introductory book on Python in the market. Aug 15, Rick Sam rated it really liked it Shelves: computer-science. A great refresher and introduction for Python programming. Very readable, fun projects, and an excellent index. Vivek rated it really liked it May 17, John rated it really liked it Aug 07, Jameson Heath Booton rated it really liked it Nov 12, Khalid Hubail rated it really liked it Oct 21, Looking to get started with computer science while learning to program in Python?

What you'll learn Skip What you'll learn. Basic Python Programming Design, implementation, documentation, and testing skills Strategies for solving computational problems Applications of CS in society and real world context. Meet your instructors Harvey Mudd College. Unfortunately, learners from one or more of the following countries or regions will not be able to register for this course: Iran, Cuba and the Crimea region of Ukraine. While edX has sought licenses from the U. Office of Foreign Assets Control OFAC to offer our courses to learners in these countries and regions, the licenses we have received are not broad enough to allow us to offer this course in all locations. EdX truly regrets that U. Computer Science. Video Transcript:. Course Type:. Associated Programs:. Computational Thinking using Python.

Share this course Share this course on facebook Share this course on twitter Share this course on linkedin Share this course via email. Prerequisites High school algebra and a reasonable aptitude for mathematics. Interested in this course for your Business or Team? Train your employees in the most in-demand topics, with edX for Business. Purchase now Request Information.