Data Structures and Algorithms in Python

Michael T. Goodrich
Department of Computer Science
University of California, Irvine

Roberto Tamassia
Department of Computer Science
Brown University

Michael H. Goldwasser
Department of Mathematics and Computer Science
Saint Louis University

Study Guide: Hints to Exercises

WILEY

Chapter 1 Python Primer

Hints

Reinforcement

R-1.1) The modulo operator could be useful here.

R-1.2) Use bit operations.

R-1.3) Keep track of the smallest and largest value while looping.

R-1.4) Although there is a formula for this, the easy thing to do is to write a loop.

R-1.5) How can you describe the range of integers for the sum?

R-1.6) Consider modifying the range over which you loop.

R-1.7) How can you describe the range of integers for the sum?.

R-1.8) Give your answer in terms of n and k.

R-1.9) Where does the sequence start and end? What is the step size?

R-1.10) Use a negative step size.

R-1.11) Those look like powers of two!

R-1.12) Use randrange to pick the index of the chosen element.

Creativity

C-1.13) The Python function does not need to be passed the value of n as an argument.

C-1.14) Note that both numbers in the pair must be odd.

C-1.15) The simple solution just checks each number against every other one, but we will discuss better solutions later in the book. But make sure you don't compare a number to itself.

C-1.16) Think about the semantics of $\mathsf{data[j]} = \mathsf{data[j]} * \mathsf{factor}$.

C-1.17) Try it out and see if it works!

C-1.18) What are the factors of each number?

C-1.19) Use the chr function with appropriate range

C-1.20) Consider randomly swapping an element to the first position, then randomly swapping a remaining element to the second position, and so on.

C-1.21) Use a list to store all the lines.

C-1.22) Go back to the definition of dot product and write a for loop that matches it.

C-1.23) Use a try-except structure.

C-1.24) You can use the condition `ch in 'aeiou'` to test if a character is a vowel.

C-1.25) Consider each character one at a time.

C-1.26) Try a case analysis for each pair of integers and an operator.

C-1.27) Either buffer the bigger value from each pair of factors, or repeat the loop in reverse to avoid the buffer.

C-1.28) Use the `**` operator to compute powers.

Projects

P-1.29) There are many solutions. If you know about recursion, the easiest solution uses this technique. Otherwise, consider using a list to hold solutions. If this still seems to hard, then consider using six nested loops (but avoid repeating characters and make sure you allow all string lengths).

P-1.30) This is the same as the logarithm, but you can use recursion here rather than calling the log function.

P-1.31) While not always optimal, you can design your algorithm so that it always returns the largest coin possible until the value of the change is met.

P-1.32) Do a case analysis to categorize each line of input.

P-1.33) Write your program to loop continually until a quit operation is entered. In each iteration, collect a sequence of button pushes, and then output the result from processing that sequence of pushes.

P-1.34) Define a way of indexing all the sentences and the location in each one and then work out a way of picking eight of these locations for a typo.

P-1.35) Use a two-dimensional list to keep track of the statistics and a one-dimensional list for each experiment.

P-1.36) You need some way of telling when you have seen the same word you have before. Feel free to just search through your list of words to do this here.