Lesson 1: Week 8

Created: 2024-05-27T01:25:49.180409+10:00

Exercise 1: Obtaining a sum from a subsequence of digits

Write a program sum_of_digits.py that prompts the user for two natural numbers, say available_digits and desired_sum, and outputs the number of ways of selecting digits from available_digits that sum up to desired_sum. For instance, if available_digits is 12234 and sum is 5 then there are four (4) solutions:one solution is obtained by selecting 1 and both occurrences of 2 (1+2+2 = 5);one solution is obtained by selecting 1 and 4 (1+4 = 5);one solution is obtained by selecting the first occurrence of 2 and 3 (2+3 = 5);one solution is obtained by selecting the second occurrence of 2 and 3 (2+3 = 5).Here are possible interactions:Input a number that we will use as available digits: 12234 Input a number that represents the desired sum: 5 There are 4 solutions. Input a number that we will use as available digits: 11111 Input a number that represents the desired sum: 5 There is a unique solution. Input a number that we will use as available digits: 11111 Input a number that represents the desired sum: 6 There is no solution. Input a number that we will use as available digits: 1234321 Input a number that represents the desired sum: 5 There are 10 solutions. Insert your code into sum_of_digits.pyIf you are stuck, but only when you are stuck, then use sum_of_digits_scaffold.py.

Exercise 2: Merging two strings into a third one

Say that two strings s1 and s2 can be merged into a third string s3 if s3 is obtained from s1 by inserting arbitrarily in s1 the characters in s2, respecting their order. For instance, the two strings ab and cd can be merged into abcd, or cabd, or cdab, or acbd, or acdb, ..., but not into adbc nor into cbda. Write a program merging_strings.py that prompts the user for 3 strings and displays the output as follows:If no string can be obtained from the other two by merging, then the program outputs that there is no solution.Otherwise, the program outputs which of the strings can be obtained from the other two by merging.Here are possible interactions:Please input the first string: ab Please input the second string: cd Please input the third string: abcd The third string can be obtained by merging the other two. Please input the first string: ab Please input the second string: cdab Please input the third string: cd The second string can be obtained by merging the other two. Please input the first string: abcd Please input the second string: cd Please input the third string: ab The first string can be obtained by merging the other two. Please input the first string: ab Please input the second string: cd Please input the third string: adcb No string can be merged from the other two. Please input the first string: aaaaa Please input the second string: a Please input the third string: aaaa The first string can be obtained by merging the other two. Please input the first string: aaab Please input the second string: abcab Please input the third string: aaabcaabb The third string can be obtained by merging the other two. Please input the first string: ??got Please input the second string: ?it?go#t## Please input the third string: it### The second string can be obtained by merging the other two. Insert your code into merging_strings.py.If you are stuck, but only when you are stuck, then use merging_strings_scaffold.py.

Exercise 3: Eight puzzle

Dispatch the integers from 0 to 8, with 0 possibly changed to None, as a list of 3 lists of size 3, to represent a 9 puzzle.For instance, let [[4, 0, 8], [1, 3, 7], [5, 2, 6]] or [[4, None ,8], [1, 3, 7], [5, 2, 6]] represent the 9 puzzlewith the 8 integers being printed on 8 tiles that are placed in a frame with one location being tile free.The aim is to slide tiles horizontally or vertically so as to eventually reach the configurationIt can be shown that the puzzle is solvable iff the permutation of the integers 1, ..., 8, determined by reading those integers off the puzzle from top to bottom and from left to right, is even. This is clearly a necessary condition since:sliding a tile horizontally does not change the number of inversions;sliding a tile vertically changes the number of inversions by -2, 0 or 2;the parity of the identity is even. Complete the program eight_puzzle.py so as to have the functionality of the two functions:validate_8_puzzle(grid) that prints out whether or not grid is a valid representation of a solvable 8 puzzle;solve_8_puzzle(grid) that, assuming that grid is a valid representation of a solvable 8 puzzle, outputs a solution to the puzzle characterised as follows:the number of moves is minimal;at every stage, the preferences of the tile to slide are, from most preferred to least preferred:the tile above the empty cell (provided the latter is not in the top row), thenthe tile to the left of the empty cell (provided the latter is not in the left column), thenthe tile to the right of the empty cell (provided the latter is not in the right column), thenthe tile below the empty cell (provided the latter is not in the bottom row).

Exercise 4: Magic squares

Given a positive integer n, a magic square of order n is a matrix of size $n \times n$ that stores all numbers from 1 up to n^2 and such that the sum of the n rows, the sum of the n columns, and the sum of the two diagonals is constant, hence equal to $n(n^2+1)/2$.Implement in the file magic_squares.py the function print_square(square), that prints a list of lists that represents a square, and the function is_magic_square(square), that checks whether a list of lists is a magic square.Examples of execution:# Examples of execution print(is_magic_square([[2,7,6], [1,5,9], [4,3,8]])) # False print(is_magic_square([[2,7,6], [9,5,1], [4,3,8]])) # True print(is_magic_square([[8,1,6],[3,5,7],[4,9,2]])) # True print_square([[8,1,6],[3,5,7],[4,9,2]]) False True True 8 1 6 3 5 7 4 9 2

Exercise 5: The numbers round

On the quiz show "Letters and Numbers" there is a round in which contestants are given six ingredient numbers and one target number, and their challenge is to apply arithmetic operations (addition, subtraction, multiplication, and division) to one or more of the ingredient numbers to get the target number. For example:Ingredient numbers: 1, 2, 6, 10, 75, 100 Target number: 582 Possible answer: (100 - (2 + 1)) × 6 = 582 Possible answer: ((100 - 2) - 1)) × 6 = 582 Possible answer: (6 x 100) - (2 x (10 - 1)) = 582 The ingredient numbers are chosen randomly as follows:0-4 large numbers, chosen from {25, 50, 75, 100} with no repeatsThe remaining are all small numbers, chosen from {1, 2, 3, ..., 10} with no repeatsEach ingredient number can only be used once, but they need not all be used. No fractions are allowed at any stage of the calculation.The target number is also chosen randomly from 100 to 1000.Your challenge is to write a program that is given the ingredient numbers and the target number and calculates all possible answers (if any).