Lesson 1: Week 5

Created: 2024-05-27T01:25:48.176957+10:00

Exercise 1: Fibonacci maker

Recall the Fibonacci numbers from practice exercise Week 3 Exercise 3: Fibonacci lister:0, 1, 1, 2, 3, 5, 8, 13, ...Write a program that returns a requested Fibonacci number, this time using a recursive function.Examples:Which Fibonacci number would you like? 1 It is 0.Which Fibonacci number would you like? 8 It is 13.Which Fibonacci number would you like? 12 It is 89.

Exercise 2: Prime factoriser

Write a program that asks the user for a number and then factorises the number into primes.Examples:Enter a number: 345 345 = 3 x 5 x 23 Enter a number: 612 612 = 2 x 2 x 3 x 3 x 17Enter a number: 127 127 = 127

Exercise 3: Longest sequence of consecutive letters

Write a program longest_sequence.py that prompts the user for a string w of lowercase letters and outputs the longest sequence of consecutive letters that occur in w, but with possibly other letters in between, starting as close as possible to the beginning of w.Insert your code into longest_sequence.py.If you are stuck, but only when you are stuck, then use longest_sequence_scaffold.py.Examples:Please input a string of lowercase letters: a The solution is: a Please input a string of lowercase letters: abcefgh The solution is: efgh Please input a string of lowercase letters: abcefg The solution is: abc Please input a string of lowercase letters: ablccmdnneofffpg The solution is: abcdefg Please input a string of lowercase letters: abcdiivjwkaalbmmbz The solution is: ijklm Please input a string of lowercase letters: abcpqrstuvwxbcbcddddeffghijklrst The solution is: abcdefghijkl

Exercise 4: A triangle of characters

Write a program characters_triangle.py that gets a strictly positive integer N as input and outputs atriangle of height N. For instance, when N = 5, the triangle looks like this:Two built-in functions are useful for this exercise:ord() returns the integer that encodes the character provided as argument;chr() returns the character encoded by the integer provided as argument.For instance:>>> ord('A') 65 >>> chr(65) 'A' Consecutive uppercase letters are encoded by consecutive integers. For instance:>>> ord('A'), ord('B'), ord('C') (65, 66, 67) Insert your code into characters_triangle.py. If you are stuck, but only when you are stuck, then use characters_triangle_scaffold_1.py.

Exercise 5: Pascal triangle

Write a program pascal_triangle.py that prompts the user for a number N and prints out the first N + 1 lines of Pascal triangle, making sure the numbers are nicely aligned, as illustrated below for N = 3, 7 and 11 respectively:Insert your code into pascal_triangle.pyIf you are stuck, but only when you are stuck, then use pascal_triangle_scaffold_1.py.

Exercise 6: Hasse diagrams

Let a strictly positive integer n be given. Let D be the set of divisors of n. Let k be the number of prime divisors of n (that is, the number of prime numbers in D). The members of D can be arranged as the vertices of a solid in a k-dimensional space as illustrated below for n = 12 (in which case D = {1,2,3,4,6,12} and k = 2) and for n = 30 (in which case D = {1,2,3,5,6,10,15,30} and k = 3).Each of the solids' vertices is associated with two collections of nodes: those "directly below" it, and those "directly above" it. In particular, the prime divisors of n are "directly above" 1, and no vertex is below 1; n has exactly k vertices "directly below" it, and no vertex is above n. This suggests considering a dictionary whose keys are the members of D (inserted from smallest to largest), and as value for a given key d, the pair of ordered lists of members of D "directly below" d and "directly above" d, respectively.The solids exhibit k distinct "edge directions", one for each prime divisor of n, defining a partition of the solids' edges. One can represent this partition as a dictionary whose keys are the prime divisors of n (inserted from smallest to largest), and as value for a given key p, the ordered list of ordered pairs of members of D that make up the endpoints of the edges whose "direction" is associated with p.The program hasse_diagram.py defines a function make_hasse_diagram() that returns a named tuple HasseDiagram with three attributes:factors, for a dictionary whose keys are the members of D, and as value for a given key d (1 excepted), a string that represents the prime decomposition of d, using x for multiplication and ^ for exponentiation, displaying only exponents greater than 1;vertices, for the first dictionary previously defined;edges, for the second dictionary previously defined.Replace pass in hasse_diagram.py with your code.Except for namedtuple, hasse_diagram.py imports a number of classes and functions from various modules that are used in the solution, but that other good solutions will make no use of.

Exercise 7: Encoding pairs of integers as natural numbers

Complete the program plane_encoding.py that implements a function encode(a, b) and a function decode(n) for the one-to-one mapping from the set of pairs of integers onto the set of natural numbers, that can be graphically described as follows:That is, starting from the point (0, 0) of the plane, we move to (1, 0) and then spiral counterclockwise:encode(0,0) returns 0 and decode(0) returns (0,0)encode(1,0) returns 1 and decode(1) returns (1,0)encode(1,1) returns 2 and decode(2) returns (1,1)encode(0,1) returns 3 and decode(3) returns (0,1)encode(-1,1) returns 4 and decode(4) returns (-1,1)encode(-1,0) returns 5 and decode(5) returns (-1,0)encode(-1,-1) returns 6 and decode(6) returns (-1,-1)encode(0,-1) returns 7 and decode(7) returns (0,-1)encode(1,-1) returns 8 and decode(8) returns (1,-1)encode(2,-1) returns 9 and decode(9) returns (2,-1) . . .

Exercise 8: Decoding a multiplication

We want to decode all multiplications of the formsuch that the sum of all digits in all 4 columns is constant.Insert your code into decoded_multiplication.py. There are actually two solutions, see expected output for details on what it should be. If you are stuck, but only when you are stuck, then use decoded_multiplication_scaffold.py.