Content

[Crystal Grid 2](#_Toc510020527)

[Common Words 3](#_Toc510020528)

[The Angles of a Triangle 3](#_Toc510020529)

[Friendly number 4](#_Toc510020530)

[Convex Hull 5](#_Toc510020531)

# Crystal Grid

The initial crystal quality check phase (link) is not enough to satisfactorily pass a crystal for use. As such, we need to work to improve the process. Since we've checked random atomic lines/rows, perhaps it would be best to perform checks on random slices of the crystal. This crystal type contains two atoms composed of the elements "X" (Xenatom) and "Z" (Zorium). In a well grown crystal, these atoms should alternate down each row and column.

You are given a slice of crystal lattice as a grid (2D array) of the atoms "X" and "Z". A well grown grid should have proper periodic arrangements both horizontally and vertically. If one atom is found next to another atom of its element, the crystal is unusable. For example:

[["X", "Z"],

["Z", "X"]] is good

[["X", "Z", "X"],

["Z", "X", "Z"],

["X", "Z", "X"]] is good

[["X", "Z", "X"],

["Z", "Z", "Z"],

["X", "Z", "X"]] is bad

Input: Atomic grid as a list of lists with strings.

Output: The crystal quality as a boolean.

# Common Words

Let's continue examining words. You are given two string with words separated by commas. Try to find what is common between these strings. The words are not repeated in the same string.

Your function should find all of the words that appear in both strings. The result must be represented as a string of words separated by commas in alphabetic order.

Input: Two arguments as strings.

Output: The common words as a string.

# The Angles of a Triangle

You are given the lengths for each side on a triangle. You need to find all three angles for this triangle. If the given side lengths cannot form a triangle (or form a degenerated triangle), then you must return all angles as 0 (zero). The angles should be represented as a list of integers in ascending order. Each angle is measured in degrees and rounded to the nearest integer number (Standard mathematical rounding).

Input: The lengths of the sides of a triangle as integers.

Output: Angles of a triangle in degrees as sorted list of integers.

Long numbers can be made to look nicer, so let’s write some code to do just that.

# Friendly number

You should write a function for converting a number to string using several rules. First of all, you will need to cut the number with a given base (base argument; default 1000). The value is a float number with decimal after the point (decimals argument; default 0). For the value, use the rounding towards zero rule (5.6⇒5, -5.6⇒-5) if the decimal = 0, otherwise use the standard rounding procedure. If the number of decimals is greater than the current number of digits after dot, trail value with zeroes. The number should be a value with letters designating the power. You will be given a list of power designations (powers argument; default ['', 'k', 'M', 'G', 'T', 'P', 'E', 'Z', 'Y']). If you are given suffix (suffix argument; default ‘’) , then you must append it. If you don’t have enough powers - stay at the maximum. And zero is always zero without powers, but with suffix.

Let's look at examples. It will be simpler.

n=102

result: "102", the base is default 1000 and 102 is lower this base.

n=10240

result: "10k", the base is default 1000 and rounding down.

n=12341234, decimals=1

result: "12.3M", one digit after the dot.

n=12000000, decimals=3

result: "12.000M", trailing zeros.

n=12461, decimals=1

result: "12.5k", standard rounding.

n=1024000000, base=1024, suffix='iB'

result: '976MiB', the different base and the suffix.

n=-150, base=100, powers=['', 'd', 'D']

result: '-1d', the negative number and rounding towards zero.

n=-155, base=100, decimals=1, powers=['', 'd', 'D']

result: '-1.6d', the negative number and standard rounding.

n=255000000000, powers=['', 'k', 'M']

result: '255000M', there is not enough powers.

Input: A number as an integer. The keyword argument "base" as an integer, default 1000. The keyword argument "decimals" as an integer, default 0. The keyword argument "powers" as a list of string, default ['', 'k', 'M', 'G', 'T', 'P', 'E', 'Z', 'Y'].

Output: The converted number as a string.

# Convex Hull

You are given a list of points on a coordinate plane. We need you find the convex hull formed by these points. The convex hull of a set X of points in the Euclidean plane is the smallest convex set that contains X. For instance: when X is a bounded subset of the plane, the convex hull may be visualized as the shape formed by a rubber band stretched around X. If a point lies on edge, it's included.

The points are presented as a list of coordinates [x, y] in which x and y are integers. The result returns as a sequence of indexes of points in the given list; points lie on the convex hull in clockwise order (see the picture). The sequence starts from the bottom leftmost point. Remember: You should return a list of indexes--not the points themselves.

example

checkio([[7, 6], [8, 4], [7, 2], [3, 2], [1, 6], [1, 8], [4, 9]]) == [4, 5, 6, 0, 1, 2, 3]

checkio([[3, 8], [1, 6], [6, 2], [7, 6], [5, 5], [8, 4], [6, 8]]) == [1, 0, 6, 3, 5, 2]