Assignment crypto

You are given a file input.json which contains data in JSON format. Take a quick peek at it. As you can see, it contains a list of cryptocoin related data represented as a list of objects, each with a currency and history field. The history is a long series of numbers representing currency rates throughout time.

We want you to write a script that summarizes this data. For each cryptocoin in the input file, determine the lowest, highest and current rate (= the last one in the history.) Write this information to a file named output.txt. For every cryptocoin, there should be one line formatted as shown below:

currency minimum maximum current

The currencies in the output file have to appear in the same order as they do in the input file.

Example

Say input.json contains the following data:

[

{

"currency": "bitcoin",

"history": [ 1, 1, 2, 3, 1000, 55000, 50000, 30000 ]

},

{

"currency": "ucll-coin",

"history": [ 1, 1, 1, 1, 1 ]

}

]

Your script needs to process each coin in turn.

The first currency is bitcoin.

* 1 is the lowest rate
* 55000 is the highest rate
* 30000 is current rate.

We output this data as

bitcoin 1 55000 50000

Similarly, the second coin can be summarized as

ucll-coin 1 1 1

These two lines need to be written to a file named output.txt:

bitcoin 1 55000 50000

ucll-coin 1 1 1

# Assignment doedel

We want to go see a movie together. Unfortunately, it is probably impossible to find a day on which we all are available. We will have to settle with trying to find a day on which as many people as possible are free.

A doedel (we don't want to get sued for trademark infringement) is a tool to help with determining this day. Each person is shown the days the movie plays in theaters and indicates which days they would be able to join.

The file input.csv contains the results. As the extension gives away, it is a CSV file. The first column contains the person's name. Each subsequent column represents a date (the first line in the file shows which dates). If a person is available on a given day, the corresponding column contains a yes, otherwise a no.

Write a script that lists all dates and the number of yes votes on that date:

date1 yescount1

date2 yescount2

date3 yescount3

...

The list should be ordered in decreasing order by number of yes votes. You can assume no two days have the same number of yes votes. The output of this script should be written to a file named output.txt.

## Example

Say the input file contains

name,01-06-2021,02-06-2021,03-06-2021

Molly Taylor,no,no,no

Joshua Wade,yes,no,yes

Traci Morris,no,yes,yes

Jennifer Harris,yes,no,yes

According to this data, the movie is only in theaters for three days: the first, second and third of June. If we count the number of yes-votes by date, we get

01-06-2021 2

02-06-2021 1

03-06-2021 3

We order by number of yes-votes:

03-06-2021 3

01-06-2021 2

02-06-2021 1

The above three lines should be written to output.txt.

# Assignment images

Typically you would know what kind of data a file contains based on its extension. For example, if a file ends on .zip, you know it's a compressed archive. A .png means the file contains an image. But what happens if the file does not have an extension?

For many file formats, it is possible to look inside the file and determine based on its contents what it actually is. For example, the 4 first bytes of a png file are always 89 50 4E 47. Other file formats can be recognized in different ways.

You are given 100 files named N.unknown where N varies from 1 to 100. The only thing you know about these files is that they contain images, but you do not know which format specifically (png, jpeg, gif, etc.) We want you to write a script that determines the image format of each of these files and generates bash commands (see example below) to rename these files so that they have the right extension. The results should be written to output.txt. The rename commands should be ordered from 1 to 100.

Hint: please do not waste time trying to manually determine the image format yourself. Look for a module.

## Example

Say you find out that 1.unknown is a jpg, then we want it to be renamed to 1.jpg. Now, do not actually rename the files. What you need to do is generate a bash command to rename 1.unknown to 1.jpg. This can be done using the following command:

mv 1.unknown 1.jpg

Produce one such mv for each file and write them (in order) to output.txt:

mv 1.unknown 1.jpg

mv 2.unknown 1.gif

mv 3.unknown 1.png

...

# Assignment layout

Take a look at the contents of input.txt. Each line contains a name, followed by an IP address and a country. Write a script that formats the contents into a nice table. The result should be written to a file named output.txt.

It's important to get the formatting exactly right. Below we list the rules.

name ip country

| | | |

<---------------> <------>

W1 W2

* There should be three columns.
  + The first column must contain the name.
  + The second column must contain the ip address.
  + The third column must contain the country.
* Each column should be just wide enough to accommodate its largest content.
  + W1 should be equal to the length of the longest name.
  + W2 should be equal to the length of the longest IP address.
  + The last column does not need padding to make it a certain width. In other words, there should be no trailing spaces.
* Columns should be separated by a single space.
* The first column should be right aligned, whereas the second and third column should be left aligned.
* Only spaces are allowed to align the text. Do not rely on tabs.
* The order in which the rows appear should be the same as in the input file.

## Example

Say input.txt contains

Sydney Clark 106.37.136.123 British Virgin Islands

Kristin Schwartz 147.202.106.81 Iceland

Steven Watkins 84.35.112.148 Brazil

Jeff Hill 74.102.135.229 Slovenia

Mr. Jesse Simpson DDS 182.157.226.16 Thailand

This should be reformatted as

Sydney Clark 106.37.136.123 British Virgin Islands

Kristin Schwartz 147.202.106.81 Iceland

Steven Watkins 84.35.112.148 Brazil

Jeff Hill 74.102.135.229 Slovenia

Mr. Jesse Simpson DDS 182.157.226.16 Thailand

# Assignment passwords

You want to hack into your mortal enemy's account. For this, you intend to try out every possible password in turn. However, you have observed him for a long time and have found out a few crucial bits of information about his password.

* His password consists of exactly 6 characters.
* This password only uses the top row letter keys on a qwerty keyboard: QWERTYUIOP.
* All letters are in uppercase.
* The password contains twice the same vowel in a row (for example, PEEQOP: the vowel E appears twice in a row). Vowels are AEIOU; we do not count Y as a vowel.
* Similarly, the password contains twice the same consonant in a row (e.g. PORRQO contains the consonant R twice in a row).
* Apart from the two vowels and two consonants mentioned above, there are no duplicate letters. In other words, after removing all duplicates, there should be 4 letters left in the password.

Produce a list of all possible passwords that satisfy these conditions. Write this list in alphabetical order to a file named output.txt.

## Hints

* Generating all passwords using the given letters is easy as there exists a function in Python's standard library that does exactly that. It has been used in one of the exercise.
* Go through all possible passwords and print out only those that satisfy the conditions.
* No complex algorithmic code is necessary.
* Hint: regex backreferences might come in handy.

## Examples

EEIOPP is one of the possible passwords:

* It consists solely of uppercase letters made from the top row keys.
* It has length 6.
* It contains the vowel E twice in a row.
* It contains the consonant P twice in a row.
* It contains four distinct letters E, I, O and P.