Create a Python program that implements a CSV (Comma separated values) file converter to JSON format. In order to carry out the intended conversion, it is important to know that the first line of the given CSV acts as a header that defines what each column represents. For example, the following file " students.csv ":

Number, Name, Course   
3162, Cândido Faísca, Theater   
7777, Cristiano Ronaldo, Sports   
264, Marcelo Sousa, Political Science

Which corresponds to a table with 3 information records: the first header line identifies the fields of each record, Number , Name , Course , and the following lines contain the information records.

However, the CSV received may contain some extensions whose semantics are explained below:

1. Lists

In the header, each field can have a number N that will represent the number of columns that this field covers. For example, let's imagine that a Grades field has been added to the previous example , with N = 5 (" students2.csv "):

Number, Name, Course, Grades{5},,,,,   
3162, Cândido Faísca, Theater, 12, 13, 14,   
15, 16 7777, Cristiano Ronaldo, Sports, 17, 12, 20, 11,   
12 264, Marcelo Sousa, Political Science, 18, 19, 19, 20, 18

This means that the Notes field covers 5 columns (note that the remaining fields were left empty, so that the **CSV matches correctly** ).

2. Lists with a range of sizes

In addition to a single size, we can also define a range of sizes {N, M} , meaning that the number of columns of a certain field can go from N to M (" students3.csv "):

Number, Name, Course, Grades{3,5},,,,,   
3162, Cândido Faísca, Theater, 12, 13, 14,,   
7777, Cristiano Ronaldo, Sport, 17, 12, 20, 11,   
12 264, Marcelo Sousa, Political Science, 18, 19, 19, 20,

Similar to the previous point, if there are empty columns, the separators must be there, the number of columns must always be equal to the maximum value of columns, they can be filled with information or not.

3. Aggregation functions

In addition to lists, we can have aggregation functions applied to these lists. See the following examples: " students4.csv "

Number, Name, Course, Grades{3,5}::sum,,,,,   
3162, Cândido Faísca, Theater, 12, 13, 14,,   
7777, Cristiano Ronaldo, Sports, 17, 12, 20, 11, 12   
264, Marcelo Sousa, Political Science, 18, 19, 19, 20,

and " students5.csv ":

Number, Name, Course, Notes{3,5}::media,,,,,   
3162, Cândido Faísca, Theater, 12, 13, 14,,   
7777, Cristiano Ronaldo, Sports, 17, 12, 20, 11, 12   
264, Marcelo Sousa, Political Science, 18, 19, 19, 20,

Expected results

The expected end result is a JSON file resulting from the conversion of a CSV file. For example, the " students.csv " (original) file , should be transformed into the following " students.json " file:

[   
{   
"Number": "3612",   
"Name": "Cândido Faísca",   
"Course": "Teatro"   
},   
{   
"Number": "7777",   
"Name": "Cristiano Ronaldo",   
"Course": "Sport"   
},   
{   
"Number": "264",   
"Name": "Marcelo Sousa",   
"Course": "Political Science"   
}   
]

In case there are lists, the fields that represent these lists must be mapped to lists in JSON (" students2.csv ==> students2.json "):

[   
{   
"Number": "3612",   
"Name": "Cândido Faísca",   
"Course": "Theatre",   
"Grades": [12,13,14,15,16]   
},   
{   
"Number": " 7777",   
"Name": "Cristiano Ronaldo",   
"Course": "Sport",   
"Grades": [17,12,20,11,12]   
},   
{   
"Number": "264",   
"Name": "Marcelo Sousa",   
"Course": "Political Science",   
"Grades": [18,19,19,20,18]   
}   
]

In cases where we have a list with an aggregation function, the processor must execute the function associated with the list, and place the result in the JSON, identifying in the key which function was executed (" students4.csv ==> students4.json " ):

[   
{   
"Number": "3612",   
"Name": "Cândido Faísca",   
"Course": "Teatro",   
"Notes\_sum": 39   
},   
{   
"Number": "7777",   
"Name": "Cristiano Ronaldo ",   
"Course": "Sport",   
"Notes\_sum": 72   
},   
{   
"Number": "264",   
"Name": "Marcelo Sousa",   
"Course": "Political Science",   
"Notes\_sum": 76   
}   
]

Others...

If you have the time and desire, you can always extend this statement by adding other aggregation functions (greater, smaller, ...), the possibility of having more than one aggregation function simultaneously in a file, ... creativity at its best... .