Scientific Programming with Python - Exercise 1

You are tasked with creating a summary file to spec for a given CSV table file. The CSV file will contain columns which will be either categorical, textual, or numerical features. Additionally, you will be given a feature spec as a JSON file. The JSON file will define the requested output: there will be one "groupby" feature, and for each other feature it will contain the requested behavior for the "group" (i.e. what aggregate function must be run on the column per group created). Textual columns will be discarded.

Remember to document your class and functions. Functions must include a short description, as well as the input arguments and output value.

Coding Instructions:

- Your module must be named "csv_summary" and it will be imported under this name.
- Your module must contain a Class called "Summary". Its constructor will receive the CSV file as the first parameter and the JSON file as the second.
- A "summary" object will have a "getGroups()" method which will return a list of all groups created.
    - The order of the groups will be in ascending order of the group name.
- Each group will be a "Group" object, defined as follows:
    - A "Group" has a "name" parameter, which is the value of the groupby feature for this group.
- The "Summary" object will have a "getSpec()" method which will return a dictionary with the following format:
    - {feature1:aggregate1, feature2:aggregate2, …, featureN:aggregateN}
- The "Summary" object will support access using the brackets '[]' operator. (by group name)
- The "Group" object will support access using the brackets '[]' operator. (by feature name or index, including negative indexing)
- The "Summary" object will be iterable (iterate group by group).
- The "Group" object will be iterable (iterate feature by feature).
    - Iterating over a group will return the tuple (feature,value)
- The "Summary" object will have a string representation (as called by the "print()" function).
    - The representation will contain each group in a new line.
- The "Group" object will have a string representation (as called by the "print()" function).
    - The representation will be in the following format:
      group_name – feature1 (aggregate1):value1, feature2 (aggregate2):value1, …, featureN (aggregateN):valueN
- The "Summary" object will have a "saveSummary(filename,[delimiter])" method which will save the summary as a CSV file, with an optional delimiter argument (the default will be a comma ',').
    - The order of the columns must be identical to that of the source CSV file, except that the groupby feature will be first.
    - The order of the groups will be in ascending order of the group name.

    - The feature names will be the header row, and will contain the aggregate function used in the parentheses text: “featureName (aggregate)”. The aggregate function for the grouped by feature will be “groupby” (see example below).

Validation Instructions:

- You must ensure that the csv and json files sent to the constructor exist.
    - If one of the files does not exist, return an empty Summary object.
    - If both files exist, you may assume that their content is valid and in the correct format.
- Empty entries for a feature may exist in the CSV file. These will be handled as follows:
    - Categorical values: will be converted to the string “NA”
    - Numerical values: will be converted to 0.
- The optional delimiter used for the `saveSummary` method must be a single non-alphanumeric and non-quotation character (i.e. not a-z, 0-9, “, ‘). Otherwise the default is used.

Possible aggregate functions:

Categorical features:

mode – value which appears most (ties broken by first in alphabetical order)

union – string containing all unique entities in category separated by semicolon (;)

unique – number of unique entities in category

count – total number of entities in category

Numerical features:

min – minimum value in category

max – maximum value in category

median – median value in category

mean – mean of values in category

sum – sum of values in category

mode – value which appears most (ties broken by lowest value)

count – total number of entities in category

Example CSV table:

Brand	Color	Kilometers	Model	Comment
Ford	Red	10000	Focus	Like new!!!!
Ford	Red	15000	Focus	Best Value!!!
Ford	Blue	55000	Explorer	Good car.
Mazda	White	100000	Lantis	New engine.
Mazda	White	15000	CX-5	Under warranty
Mazda	Gray	15000	CX-5	Under warranty
Nissan	White	50000	Micra	Gas efficient!

Example JSON:

{

“groupby”: “Brand”,

[

{“Color”:{“type”:”categorical”, “aggregate”:”mode”},

{“Kilometers”:{“type”:”numerical”, “aggregate”:”max”},

{“Model”:{“type”:”categorical”, “aggregate”:”unique”},

{“Comment”: {“type”:”textual”}

]

}

Example result:

Brand (groupby)	Color (mode)	Kilometers (max)	Model (unique)
Ford	Red	55000	2
Mazda	White	100000	2
Nissan	White	50000	1