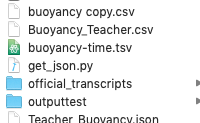
**get\_json.py**

This Python script takes a CSV file (or directory with CSV files) and creates the corresponding JSON file(s). This document is assuming Mac OS as operating system, and Python 2.7.

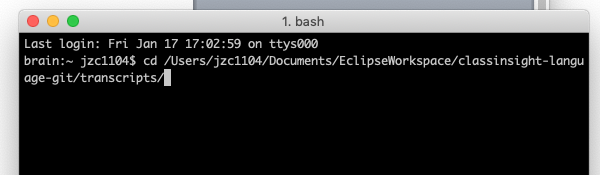
Process:

1. Place the python script and CSV files in a directory

For example, I put it in: /Users/jzc1104/Documents/EclipseWorkspace/classinsight-language-git/transcripts



1. In a terminal, navigate to that directory using the command “cd”



1. Run the script typing:

python get\_json.py

Like that, with no extra options, the script will assume that the folder where the script is contains all the CSV files to be converted to JSON. So it will try to convert all CSV files in that folder. Similarly, it will assume that the output folder is where the script is located and it will put all the json files in that folder.

If the output directory already contains some JSON files, and the script tries to create a new file with the SAME filename as an existing JSON file, then the script will OVERWRITE that file.

**FILENAMES**

For each input file, the script assumes the filename contains information. Namely, the CSV filename is expected with the following format:

perioddate + “\_” + teacherpseudonym + “\_” + extrasuffixes + “.csv”

For example:

190429\_Michelle\_Per\_5.csv

20190423\_Henry\_Per2.csv

After processing the JSON file will be created with the following format:

teacherpseudonym + “\_” + perioddate + “\_” + extrasuffixes + “.json”

So for the previous examples, the following json files will be created:

Michelle\_190429\_Per\_5.json

Henry\_20190423\_Per2.json

**OTHER OPTIONS**

**-i (or --inputdir) folder\_path**

This option tells the script the folder that contains all the CSV files to be converted

Examples:

python get\_json.py –i /Users/jzc1104/CSVFiles

which is equivalent to

python get\_json.py –-inputdir /Users/jzc1104/CSVFiles

python get\_json.py –i transcripts/CSVFiles

python get\_json.py –i transcripts . (the period “.” indicates the current directory)

If the output directory is not specified, the script will place the output files in the directory where the script is located.

**-o (or --outputdir) folder\_path**

This option tells the script the folder where the JSON files are to be placed

Examples:

python get\_json.py –o /Users/jzc1104/JSONFiles

which is equivalent to

python get\_json.py –-outputdir /Users/jzc1104/JSONFiles

python get\_json.py –o transcripts/JSONFiles

python get\_json.py –o transcripts . (the period “.” indicates the current directory)

**-f (or --filename) path\_to\_inputfile**

With this option the script will only try to convert one single CSV file, which is specified here.

Examples:

python get\_json.py –f /Users/jzc1104/CSVFiles/buoyancy.csv

python get\_json.py –f CSVFiles/buoyancy.csv

**EXPECTED FORMAT OF THE CSV FILES**

In order for the script to work, the input CSV files need to have a certain format.

**Windows Comma Separated values**

The CSV files are converted from Excel files. In that process please use the option “Windows Comma Separated” in Excel to convert to CSV. This option appears in Excel after opening the “Save as” menu.

**Headers and Columns**

The first line of the CSV file contains the names of the columns. Right now, all the CSV files are expected to have the following Headers:

[*'Speaker'*,

*'Time\_Stamp'*,

*'Transcript'*,

*'Turn-Taking Facilitation'*,

*'Metacognitive Modeling Questions'*,

*'Behavior Management Questions'*,

*'Teacher Open-Ended S/Q'*,

*'Teacher Close-Ended S/Q'*,

*'Student Open-Ended S/Q'*,

*'Student Close-Ended S/Q'*,

*'Student Close-Ended Response'*,

*'Student Open-Ended Response'*,

*'Student Explanation + Evidence'*,

*'Whole class discussion'*,

*'Lecture'*,

*'Small group + Instructor'*,

*'Individual Work'*,

*'Pair Work'*,

*'Other'*]

If the CSV file has a different order of columns, or if a column does not exist or it is duplicated, it will most likely cause an ERROR and will definitely provoke incorrect JSON Files.

There were multiple errors caused by small typos in the headers (for example, having “Lecture ” instead of “Lecture”, where there is an extra space in the former). Because of that, the script now completely replaces the headers of the original CSV files with the expected headers. This avoids errors related to typos in the column names, BUT if the columns are not in the right order, etc, the created JSON files will still have incorrect information.

**Timestamps**

The format of the timestamps can be as follows:

*“Hour:Min:Sec"* for example: 12:01:05

*"Min:Sec""* for example: 10:05

**Information out of place**

If there is **ANY** information that is not supposed to be in a column, it will most likely create an **error.** For example, some files contained annotations that were only meant to be seen by other annotators.