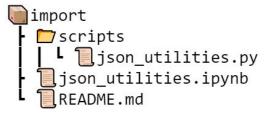
# WE1S "json\_utilities" module

A component of the WE1S Workspace (see S-2), the "json utilities" module provides a method of accessing the contents of the json folder in a Workspace project. Depending on the number of texts in a project (the Workspace imports texts in JSON format), this folder can be quite large, and will cause the user's web browser to freeze when opening it in the Jupyter notebook file interface. The "json utilities" module creates a Documents object with which a user can call methods that list or read the contents of the files in the json folder, enabling database-like queries of texts and metadata fields, filtering of results, and export of results to a zip archive.

The main Jupyter notebook in the module is json\_utilities.ipynb. It includes cells for carrying out operations on a Workspace project's json folder that include the following:

- View metadata fields
- Perform queries (including with Boolean logic)
- Export results of a query to a zip file.

## Module structure



## Screenshot from json\_utilities.ipynb (<u>larger</u>)

+	2: 2 1 + + HR.n = C H Matters M	
	JSON Utilities	
	This notebook provides a method of accessing the contents of a project's [seen folder. These folders can be quite large, and the	well store the brown
	freeze if they are opened using the Jupyter notebook file browser. This notebook creates a Document's object with which you ca	an call methods that list
	the contents of the files in the [see folder. It also allows you to perform database-like queries on the contents to filler your results a order with the contents to filler your results and the contents to filler your results and the contents of the files in the second secon	its and to export the res
	Info	
	authors = "Scott Klainman"	
	ecpyright = 'cosyright 2020, The WE1S Project' licence = 'MIT'	
	relon = "1.0" sall = "scott kleinmanilicoun edu"	
	Setup	
	90387403	
[]:	# Import the Documents class New json_utilities_scripts.py	
	# Python Imports	
	from pathlib import Fath	
	# Get the project directory	
	4	
	Create a Documents Object	
	The cell below shows you how to create a Bocument's object and use it to list or read files.	
	The get_file_list() methods can optionally take start and end values as shown in the example below.	
11	# Create a Documents abject for the project folder docs = Documents(project dir)	
	# Set the number of documents num_dock = dock.count print(Number of documents: ' + str(num_docs))	
	<pre>print('Number of documents: ' + str(num_docs)) print()</pre>	
	# Get a List of the First 5 documents in the juan folder result = doc.mptfile_list(0, 5) print("Pirst Five Films:(n") print("exist)	
	print(result) print()	
	# Count the matter of documents is a result one results = doc.count.doc.rout.loc.rout.presults)) primt("Manaber of documents in result: ' + str(num results)) primt()	
	<pre>num_results + docs.count_docs(result) print('Mamber of documents in result: ' + str(num_results))</pre>	
	# Read a document by filename doc = docs.read('myfile.json')	
	<pre># food r document by filenese doc - docs-ready (myfile)(ison') print("isit 300 document")(n') print("isit 300 document")(n') print(doc["content"][10:100] + '')</pre>	
	4	. h
	View Metadata Fields (Optional)	
	If you wish to perform a query on your documents, it can be helpful to know what metadata fields are available. The call below w	
	and extract the keys for each metadata field. Note that listed keys may not be available in all documents. If you think that your may want to run docs, get metadata, keys() without start and end values. However, this can take a long time, so it is not	etadata is very inconsis tracommended unless
	have reason to think that there are large discrepancies across your collection.	
	It is also possible to get the keys for a specific file with docs.get_metadata_keys(filelist=['file1', 'file1', etc.]).	If you have already run
	something like result = decs.get_file_list(0, 5) you can simply run_decs.get_metadata_keys(filelist=result)	
[]:	fields = docs.get_metadata_keys(0, 100)	
		h.
	You can generate a table of your documents with get_table(). It takes a list of files and a list of fields as its arguments, as in can be re-ordered, sorted, and filered. However, it is recommended that you only supply a small number of columns. The bigger	the example below. Col
	time when you scroil.	
11-	file list - decs.get file list() tath - decs are subjective to the decay to decay to	
1.15		
	4	
	If you wish to save the table after you have sorted and/or filtered it, set the Filtename in the cell below and run the cell.	
1.14	# Configuration filename = 'table.csv'	
		14/2
	4	- b
	Performing Queries	
	Although many questions about your data can be answered by working with the table above, sometimes you may need to perform database-like queries to filter the data in your project's joon tolder. The cell below provide an interface for performing these queri	mmure sophisticated

## Further Information:

\* M-15 (about Jupyter notebooks)

## Main Jupyter notebooks in this module:

\* json utilities.ipynb

Code source: [TBD] (MIT License)