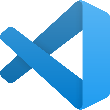
**HOP06D – Reading and Writing Data**

06/06/2020 Created by Apiwat Chuaphan

11/08/2020 Revised by Kim Nguyen

Center for Information Assurance (CIAE) @City University of Seattle (CityU)

 A close up of a screen

Description automatically generated A close up of a sign

Description automatically generated

**Before You Start**

* The directory path shown in screenshots may be different from yours.
* Some steps might not be explained in the tutorial.  If you are not sure what to do:
  + Consult the resources listed below.
  + If you cannot solve the problem after a few tries, ask a TA for help.

**Learning Outcomes**

* Learn how to use Python's Pandas library to read and write CSV files.
* Learn how to read and write JSON files.
* Learn how to read API data and serialize JSON file.

**Resources**

* Pandas Documentation: <https://pandas.pydata.org/docs/user_guide/io.html?highlight=json#io-json-writer>
* Data from Kaggle: <https://www.kaggle.com/datasets>

**Read and Write – Extra Resources**

**Reading and Writing JSON**

1. Create a new file named **users.json** under Module folder
2. Copy a sample JSON file from <https://raw.githubusercontent.com/stcta/CS612-Data-Analysis/master/example/data/users.json> and paste into the file we just created and then save it.
3. Go back to your notebook and add the following cell to the file for reading JSON file.

A screenshot of a cell phone

Description automatically generated

1. In this step, we will create some random data and save it to JSON format object. We will import **numpy** for random numbers.

A close up of a screen

Description automatically generated

1. You will notice that you have new JSON file named **new\_users.json** under module folder.

**Reading and Writing API**

1. We will read data from API which contains JSON data format.
2. In the same file, type the following at the end of the same file.

A close up of a screen

Description automatically generated

**Requests** allows you to send HTTP/1.1 requests extremely easily. It returns a Response Object with all the response data (content, encoding, status, etc). You will also be able to perform web scraping using Python.

If you do not have requests, install it by running **pip install requests** in terminal.

1. Next, we have a response data from API and we will encode that to json format.A screenshot of a cell phone

   Description automatically generated

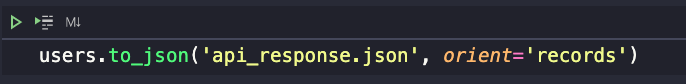
As you can see, we print out one user from the 10 users in this case. This API has 10 user’s data in total. **all\_usere** is a list of multiple JSON object or dict-like in Python.

1. Now, we formatted that data into DataFrame object and it is much easily to see and you can manipulate it with pandas later.

A screenshot of text

Description automatically generated

1. We can also save the response to JSON file like what we did earlier. However, we changed the orient parameter to records.

****

What is orient parameter?

* **Column** oriented (the default for DataFrame) serializes the data as nested JSON objects with column labels acting as the primary index:
* **Index** oriented (the default for Series) similar to column oriented but the index labels are now primary:
* **Record** oriented serializes the data to a JSON array of column -> value records, index labels are not included. This is useful for passing DataFrame data to plotting libraries, for example the JavaScript library d3.js:
* **Value** oriented is a bare-bones option which serializes to nested JSON arrays of values only, column and index labels are not included:
* **Split** oriented serializes to a JSON object containing separate entries for values, index and columns. Name is also included for Series:
* **Table** oriented serializes to the JSON Table Schema, allowing for the preservation of metadata including but not limited to dtypes and index names.

You can play around with these options to get different format of serializing