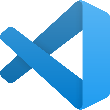
**CS506 Programming for Computing**

**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**

Pandas is a very powerful and popular framework for data analysis and manipulation. One of the most use features of Pandas is the ability to read and write various types of files including CSV.

The Pandas I/O API is a set of top level reader functions accessed like pd.read\_csv() that generally return a Pandas object. More available readers and writers can be found here <https://pandas.pydata.org/docs/user_guide/io.html>

Common functionalities:

| **Format Type** | **Data Description** | **Reader** | **Writer** |
| --- | --- | --- | --- |
| text | CSV | read\_csv | to\_csv |
| text | Fixed-Width Text File | read\_fwf |  |
| text | JSON | read\_json | to\_json |
| text | HTML | read\_html | to\_html |
| text | Local clipboard | read\_clipboard | to\_clipboard |
|  | MS Excel | read\_excel | to\_excel |

**Reading and Writing CSV**

Open Jupyter Notebook:

1. Create a new file named **sales\_data\_sample.csv** under Module folder.
2. Copy a sample CSV data from <https://raw.githubusercontent.com/stcta/CS612-Data-Analysis/master/example/data/sales_data_sample.csv> and paste into the file we just created and then save it.
3. Under module folder, create a new file called **reading\_writing.ipynb** and simply click on the file to open notebook.
4. Type the following into the file just created. Run selected cell to see each result.

**A screenshot of a computer

Description automatically generated**

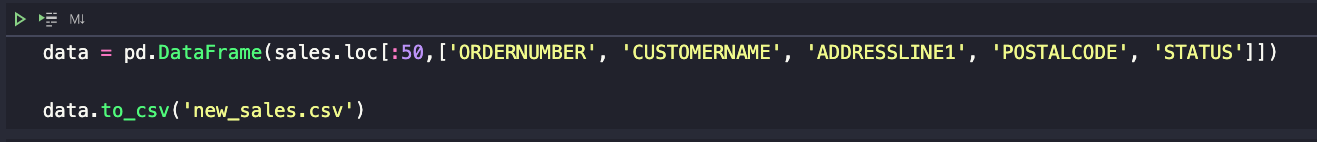
**'latin-1' or 'iso-8859-1'** is the simplest text encoding maps the code points 0–255 to the bytes 0x0–0xff, which means that a string object that contains code points above U+00FF can’t be encoded with this codec. Doing so will raise a **UnicodeEncodeError**

1. This data has 25 columns, you might not see the whole thing, but you still can see what they are with the columns DataFrame like below.

A screenshot of a cell phone

Description automatically generated

1. Now we will write new CSV file with multiple columns selected by the indexing operator by passing it a list of column names.



The **Series** and **DataFrame** objects have an instance method **to\_csv** which allows storing the contents of the object as a comma-separated-values file.

There are a couple common exceptions that arise when doing selections with just the indexing operator.

* + If you misspell a word, you will get a KeyError
  + If you forgot to use a list to contain multiple columns you will also get a KeyError

1. You just created a new CSV file named **new\_sales.csv** with **to\_csv** method of DataFrame. See the file under your module folder.