CS506 Programming for Computing

HOS06D– Reading and Writing Data

06/06/2020 Created by Apiwat Chuaphan

3/28/2022 Revised by Cedric Huang

03/08/2023 Reviewed by Maram Elwakeel

School of Technology & Computing (STC) @ City University of Seattle (CityU)


Before You Start

- Version numbers may not match with the most current version at the time of writing. If given the option to choose between stable release (long-term support) or most recent, please select the stable release rather than the beta-testing version.
- There might be subtle discrepancies along with the steps. Please use your best judgment while going through this cookbook-style tutorial to complete each step.
- For your working directory, use your course number. This tutorial may use a different course number as an example.
- All the steps and concepts in this tutorial are from the textbook, so if you encounter problems in this tutorial, please try to read and compare the textbook to solve the problem. If you still can't solve the problem, please feel free to contact your course TA.
- Avoid copy-pasting code from the book or the GitHub repository. Instead, type out the code yourself. Resort to copy-pasting only when you are stuck and find that things are not working as expected.

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

Section 1 - Read and Write

1) Pandas is a very powerful and popular framework for data analysis and manipulation. One of the most useful features of Pandas is the ability to read and write various types of files including CSV.
2) 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
3) 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

4) **Reading and Writing CSV** - Open Jupyter Notebook:
5) Create a new file named `sales_data_sample.csv` under Module folder.
6) 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.

7) Under module folder, create a new file called `reading_writing.ipynb` and simply click on the file to open notebook.

8) Type the following into the file just created. Run selected cell to see each result.

```
import pandas as pd
```

```
sales = pd.read_csv('sales_data_sample.csv', encoding='latin')
```

```
sales
```

	ORDERNUMBER	QUANTITYORDERED	PRICEEACH	ORDERLINENUMBER	SALES	ORDERDATE	STATUS	QTR_ID	MONTH_ID	YEAR_ID	...	ADDRESSLINE1	ADDRESSLINE2	CITY	STATE	POS
0	10107	30	95.70	2	2871.00	2/24/2003 0:00	Shipped	1	2	2003	...	897 Long Airport Avenue	NaN	NYC	NY	
1	10121	34	81.35	5	2765.90	5/7/2003 0:00	Shipped	2	5	2003	...	59 rue de l'Abbaye	NaN	Reims	NaN	
2	10134	41	94.74	2	3884.34	7/1/2003 0:00	Shipped	3	7	2003	...	27 rue du Colonel Pierre Avia	NaN	Paris	NaN	
3	10145	45	83.26	6	3746.70	8/25/2003 0:00	Shipped	3	8	2003	...	78934 Hillside Dr.	NaN	Pasadena	CA	
4	10159	49	100.00	14	5205.27	10/10/2003 0:00	Shipped	4	10	2003	...	7734 Strong St.	NaN	San Francisco	CA	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2818	10350	20	100.00	15	2244.40	12/2/2004 0:00	Shipped	4	12	2004	...	C/ Moralzarzal, 86	NaN	Madrid	NaN	
2819	10373	29	100.00	1	3978.51	1/31/2005 0:00	Shipped	1	1	2005	...	Torikatu 38	NaN	Oulu	NaN	
2820	10386	43	100.00	4	5417.57	3/1/2005 0:00	Resolved	1	3	2005	...	C/ Moralzarzal, 86	NaN	Madrid	NaN	
2821	10397	34	62.24	1	2116.16	3/28/2005 0:00	Shipped	1	3	2005	...	1 rue Alsace-Lorraine	NaN	Toulouse	NaN	
2822	10414	47	65.52	9	3079.44	5/6/2005 0:00	On Hold	2	5	2005	...	8616 Spinnaker Dr.	NaN	Boston	MA	

2823 rows × 25 columns

`'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**

9) 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.

```
sales.columns
```

```
Index(['ORDERNUMBER', 'QUANTITYORDERED', 'PRICEEACH', 'ORDERLINENUMBER',
       'SALES', 'ORDERDATE', 'STATUS', 'QTR_ID', 'MONTH_ID', 'YEAR_ID',
       'PRODUCTLINE', 'MSRP', 'PRODUCTCODE', 'CUSTOMERNAME', 'PHONE',
       'ADDRESSLINE1', 'ADDRESSLINE2', 'CITY', 'STATE', 'POSTALCODE',
       'COUNTRY', 'TERRITORY', 'CONTACTLASTNAME', 'CONTACTFIRSTNAME',
       'DEALSIZE'],
      dtype='object')
```

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

```
data = pd.DataFrame(sales.loc[:50,['ORDERNUMBER', 'CUSTOMERNAME', 'ADDRESSLINE1', 'POSTALCODE', 'STATUS']])

data.to_csv('new_sales.csv')
```

11) 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.

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

   ii. If you misspell a word, you will get a KeyError

   iii. If you forgot to use a list to contain multiple columns, you will also get a KeyError

   iv. You just created a new CSV file named `new_sales.csv` with to_csv method of DataFrame.

   v. See the file under your module folder.

12) Save your Jupyter Notebook with all Output