.csv, .tsv. *and* .json files

April 4, 2022

Administrative Notes

- Keep working on Project 1. You have one more week to get it finished. Don't wait until the last minute
- Remember that next week's lectures are on-line
  - I'm in Colorado all week
  - Links/invites to the lectures will be provided on Discord and Blackboard
- After I get back, the week of April 18-20 is exam week.
  - Monday, April 18 - review for the exam
  - Wednesday, April 20 - exam 2
    - Same format as last time
    - Sample exam will be made available on the github repo next week
    -

More on File I/O

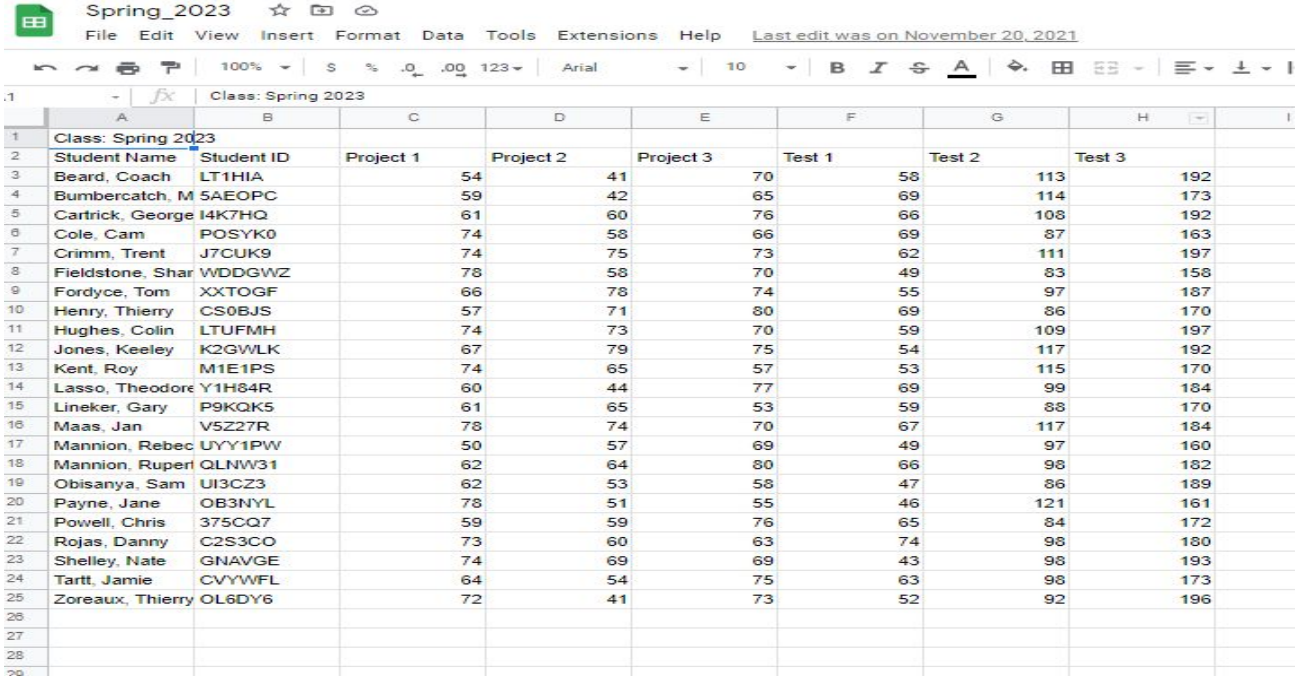
This summarizes and adds to some of the lecture material from right before Spring Break

Three common file formats used in today's computing environment:

- Comma-separated value - .csv
  - Commonly used to exchange data from spreadsheets
    - Each row is on a line; each value separated from other values by a comma
    - Works well if you don't have a a lot of commas in your data
      - Commas in the data can be escaped using quotes - "," - but that gets awkward if you have a lot of commas in your data
- Tab-separated value - .tsv
  - An alternate way to exchange data from spreadsheets
    - Useful when you have a lot of commas in your data - cuts down on escape characters
    - Less common than .csv but useful
- JavaScript Object Notation - .json
  - Commonly used to share data between web applications and web servers

.csv and .tsv files

Here’s a data file from Google sheet (Excel and others work the same way):



Spring_2023

File Edit View Insert Format Data Tools Extensions Help Last edit was on November 20, 2021

Class: Spring 2023

	A	B	C	D	E	F	G	H
1	Class: Spring 2023							
2	Student Name	Student ID	Project 1	Project 2	Project 3	Test 1	Test 2	Test 3
3	Beard, Coach	LT1HIA	54	41	70	58	113	192
4	Bumbercatch, M	5AEOPC	59	42	65	69	114	173
5	Cartrick, George	I4K7HQ	61	60	76	66	108	192
6	Cole, Cam	POSYK0	74	58	66	69	87	163
7	Crimm, Trent	J7CUK9	74	75	73	62	111	197
8	Fieldstone, Shar	WDDGWZ	78	58	70	49	83	158
9	Fordyce, Tom	XXTOGF	66	78	74	55	97	187
10	Henry, Thierry	CS0BJS	57	71	80	69	86	170
11	Hughes, Colin	LTUFMH	74	73	70	59	109	197
12	Jones, Keeley	K2GWLK	67	79	75	54	117	192
13	Kent, Roy	M1E1PS	74	65	57	53	115	170
14	Lasso, Theodore	Y1H84R	60	44	77	69	99	184
15	Lineker, Gary	P9KQK5	61	65	53	59	88	170
16	Maas, Jan	V5Z27R	78	74	70	67	117	184
17	Mannion, Rebec	UYY1PW	50	57	69	49	97	160
18	Mannion, Rupert	QLNW31	62	64	80	66	98	182
19	Obisanya, Sam	UI3CZ3	62	53	58	47	86	189
20	Payne, Jane	OB3NYL	78	51	55	46	121	161
21	Powell, Chris	375CQ7	59	59	76	65	84	172
22	Rojas, Danny	C2S3CO	73	60	63	74	98	180
23	Shelley, Nate	GNAVGE	74	69	69	43	98	193
24	Tartt, Jamie	CVYWFL	64	54	75	63	98	173
25	Zoreaux, Thierry	OL6DY6	72	41	73	52	92	196

Downloading/saving as .csv and .tsv

The .csv version

```
Class: Spring 2023,,,,,,,
Student Name,Student ID,Project 1,Project 2,Project 3,Test 1,Test 2,Test 3
"Beard, Coach",LT1HIA,54,41,70,58,113,192
"Bumbercatch, Moe",5AEOPC,59,42,65,69,114,173
"Cartrick, George",I4K7HQ,61,60,76,66,108,192
"Cole, Cam",POSYK0,74,58,66,69,87,163
"Crimm, Trent",J7CUK9,74,75,73,62,111,197
"Fieldstone, Sharon",WDDGWZ,78,58,70,49,83,158
"Fordyce, Tom",XXTOGF,66,78,74,55,97,187
"Henry, Thierry",CS0BJS,57,71,80,69,86,170
"Hughes, Colin",LTUFMH,74,73,70,59,109,197
"Jones, Keeley",K2GWLK,67,79,75,54,117,192
"Kent, Roy",M1E1PS,74,65,57,53,115,170
"Lasso, Theodore",Y1H84R,60,44,77,69,99,184
"Lineker, Gary",P9KQK5,61,65,53,59,88,170
"Maas, Jan",V5Z27R,78,74,70,67,117,184
"Mannion, Rebecca",UYY1PW,50,57,69,49,97,160
"Mannion, Rupert",QLNW31,62,64,80,66,98,182
"Obisanya, Sam",UI3CZ3,62,53,58,47,86,189
"Payne, Jane",OB3NYL,78,51,55,46,121,161
"Powell, Chris",375CQ7,59,59,76,65,84,172
"Rojas, Danny",C2S3CO,73,60,63,74,98,180
"Shelley, Nate",GNAVGE,74,69,69,43,98,193
"Tartt, Jamie",CVYWFL,64,54,75,63,98,173
"Zoreaux, Thierry",OL6DY6,72,41,73,52,92,196
```

The .tsv version

```
Class: Spring 2023
Student Name	Student ID	Project 1	Project 2	Project 3	Test 1	Test 2	Test 3
Beard, Coach	LT1HIA	54	41	70	58	113	192
Bumbercatch, Moe	5AEOPC	59	42	65	69	114	173
Cartrick, George	I4K7HQ	61	60	76	66	108	192
Cole, Cam	POSYK0	74	58	66	69	87	163
Crimm, Trent	J7CUK9	74	75	73	62	111	197
Fieldstone, Sharon	WDDGWZ	78	58	70	49	83	158
Fordyce, Tom	XXTOGF	66	78	74	55	97	187
Henry, Thierry	CS0BJS	57	71	80	69	86	170
Hughes, Colin	LTUFMH	74	73	70	59	109	197
Jones, Keeley	K2GWLK	67	79	75	54	117	192
Kent, Roy	M1E1PS	74	65	57	53	115	170
Lasso, Theodore	Y1H84R	60	44	77	69	99	184
Lineker, Gary	P9KQK5	61	65	53	59	88	170
Maas, Jan	V5Z27R	78	74	70	67	117	184
Mannion, Rebecca	UYY1PW	50	57	69	49	97	160
Mannion, Rupert	QLNW31	62	64	80	66	98	182
Obisanya, Sam	UI3CZ3	62	53	58	47	86	189
Payne, Jane	OB3NYL	78	51	55	46	121	161
Powell, Chris	375CQ7	59	59	76	65	84	172
Rojas, Danny	C2S3CO	73	60	63	74	98	180
Shelley, Nate	GNAVGE	74	69	69	43	98	193
Tartt, Jamie	CVYWFL	64	54	75	63	98	173
Zoreaux, Thierry	OL6DY6	72	41	73	52	92	196
```

There are Python modules that handle this for you

…But since this is “Computer Science” and not “Coding in Python” we’re going to learn to do it from scratch

- Then you can use the built-in modules later on.

Reading these files

If you know that the file you're reading from is a .csv file, the best approach is:

- Step 1, use readline() or readlines() to read the file in a line at a time. You know that the lines in the file are separated by \n, so let Python do the separating for you
- Step 2, use split(“,”) to split each line into its component elements. Each line is read in as a string. Split that string on the commas. This will throw away the commas, and separate the actual elements into their proper place
- Step 3, all components are still strings. Convert elements to ints, floats, etc. as necessary.

.tsv files are similar

Just split the string using the tab character, “\t”

A warning on commas and tabs in data files

If you're writing a file, embed the comma in the string before you do the ",".join(). Then use something like quotes around the string so that the comma will later be skipped.

If you're reading the file, you might run into problems

- There's no explicit way to mark a comma in the file as "this is a separator" vs. "This is an actual data value." You hope that the person who created the file anticipated this for you.
- You're going to have do some EDA - exploratory data analysis - to see if you are going to have that problem
- If you do run into the problem, you'll have to write code to address it on a case-by-case basis.

Some examples

Writing to a .csv file

Remember that you can only write one string to a file at a time.

So you have to put everything together into a string to write it.

Use the join() command to create the strings.

- Using “,”.join() automatically puts the commas into place between the components of the string - the “values” to be “separated” by “commas”
- But remember how join works - it only works on iterables (lists or dictionaries) and the components of those lists or dictionaries have to themselves be strings

Writing to .csv

An example:

l = [‘a’, ‘b’, ‘c’, ‘d’, ‘e’, ‘f’]

Using

s = “,”.join(l)

.tsv: s = “\t”.join(l)

gives you a string with elements separated by commas, which you can now write to your file

What if instead we had

l = [1,2,3,4,5,6]

What does

s = “,”.join(l)

produce?

write() vs writelines()

“write()” writes a single string to a file. Using this means you write one line at a time

“writelines()” lets you write multiple strings to a file in a single statement. If you have a list of strings, you can write them all to the file with a single use of “writelines”

- Our examples will use “write()” for simplicity

An example

```
with open ("C:\\Users\\Al\\PycharmProjects\\exam_qs\\data.csv", "w") as outfile:
    l = ['1','2','3','4','5','6']
    s = ",".join(l)
    print (s)
    outfile.write(s)
    outfile.write('\n')
    outfile.write(s)
```

This is the full path on a Windows machine. You have to escape the backslash character to make Windows understand what you want. That's why the double "\\" exist.

JSON files