CSCI 1070: Taming big data

Lab 1: Using files and plotting

Brief description

In this studio you will practise reading information from files, manipulating and plotting it. You will also review loops, lists and dictionaries.

Useful resources

- An excellent introduction on f-strings: click here.
- Joining and splitting string in Python: video here.
- Reading CSV and JSON files in Python.
- Basic plotting using `matplotlib`: click here.
- Also, feel free to consult and use the code seen in class (see Canvas).

Important!

- Please read the section of the syllabus relevant to what structures and constructs you are expected to use in the assignments. For this studios, you are allowed to use arithmetic, conditional statements, loops and `print` statements, as well as line, bar and scatter plots. Dictionaries, including `defaultdict`, tuples and lists are also allowed, as are CSV and JSON files.
- Feel free to use any part of the code seen in class, adapting it as needed.
- Make sure that the name of each person who worked on these exercises is listed in the first answer.
- You can choose whether to read the CSV files as lists or as dictionaries. Consult the corresponding code on Canvas for relevant examples.
- Work through the exercises in the specified order. Also, **check carefully that the output matches the requirements**, you will be evaluated on how precisely this is done. All printouts must be **informative** and **use f-strings**.
- Once you have finished with each problem, save the `Python` code to the specified file. Then compress all your `.py` files into a `zip` file and submit it via Canvas before the assigned deadline.
- The file name for you submission **must** be as follows:
  CSCI1070-F25-Lab1-Lastname1_Firstname1_Lastname2_Firstname2.zip
  Example: CSCI1070-F25-Lab1-Doe_Jane_Citizen_Joe.zip

Exercises

1. ☞ As the answer to the first exercise, list the names of the people who worked together on this studio.

2. Let us explore the structure of the `temps.csv` file. Write the Python code which accomplishes the following:
   a) ☞ open said file and read its contents.
   b) ☞ provide a printout with the file's header.
   c) ☞ calculate and print out the number of columns.
   d) ☞ print out the column names[1].
   e) ☞ calculate and print out the **number of rows of data**[2].
   f) ☞ print out the data type of every column. For this, you can use any data row and the `type()` method.

   ☞ Save your answer to a file called `L1-p2.py`.

3. We shall now explore the structure of the `temps.json` file. Write the Python code which accomplishes the following:
   a) ☞ open said file and read its contents.
   b) ☞ calculate and print out the number of fields.
   c) ☞ print out the field names.
   d) ☞ calculate and print out the **number of data points**.
   e) ☞ print out the data type of every field. For this, you can use any data row and the `type()` method.

   ☞ Save your answer to a file called `L1-p3.py`.

4. ☞ Provide the Python code that plots the measurements found in `temps.csv`. Plot both types of measurement, label the axes appropriately, provide a title and use a different colour for each measurement.

   ☞ Save your answer to a file called `L1-p4.py`.

5. ☞ Provide the Python code that plots the measurements found in `temps.`json. Plot both types of measurement, label the axes appropriately, provide a title and use a different colour for each measurement.

   ☞ Save your answer to a file called `L1-p5.py`.

6. `sales.json` contains the sales for a particular day in a campaign. Write the Python code which accomplishes the following:
   a) ☞ open said file and read its contents.
   b) ☞ create a dictionary which has as its key the category names and whose values are the number of sales per category. Print said dictionary clearly showing each category and each count.

      Example output: `Electronics :  12 sales`
   c) ☞ create a dictionary which has as its key the category names and whose values are the total sales per category. For this, you can either use a `defaultdict` or start with a dictionary whose values are initialised to 0. Print said dictionary clearly showing each category and its corresponding total.

      Example output: `Electronics :  1445.87 euros`
   d) ☞ use the dictionary from the previous question to calculate the sum of all sales. Print said value using an informative string.
   e) ☞ plots the sales per category. For this, use a bar chart. Plot all categories, label the axes appropriately, and provide a title.

   ☞ Save your answer to a file called `L1-p6.py`.

7. `sales.csv` contains the same information as above but in the CSV format. Write the Python code which accomplishes the following:

[1] Joining and splitting strings will help here
[2] Remember, data does not include the header

a) ☞ open said file and read its contents.

b) ☞ create a dictionary which has as its key the category names and whose values are a counter from 0 to the number of categories less 1.

c) ☞ create a list whose values are the total sales per category. Make sure that the indices correspond to the numbers you chose in the previous question. Print the values of said list clearly showing each category and its corresponding count. The answer should coincide with the the value obtained in question 6b.

   Example output: `Electronics :  12 sales`

d) ☞ create a list whose values are the total sales per category. Once again, make sure that the indices correspond to the numbers you chose in question 7b. Print the values of said list clearly showing each category and its corresponding total. The answer should coincide with the the value obtained in question 6c.

   Example output: `Electronics :  1445.87 euros`

e) ☞ use the list from the previous question to calculate the sum of all sales. Print said value using an informative string. The answer should coincide with the the value obtained in question 6d.

f) ☞ plots the sales per category. For this, use a bar chart. Plot all categories, label the axes appropriately, and provide a title.

☞ Save your answer to a file called `L1-p7.py`.