Students:

This content is controlled by your instructor, and is not zyBooks content. Direct questions or concerns about this content to your instructor. If you have any technical issues with the zyLab submission system, use the **Trouble with lab** button at the bottom of the lab.

10.8 LAB CHECKPOINT: Load from CSV

LAB: Load from CSV

CSW8 Learning Goals

In this lab we will -

- Create a dictionary from its provided description
- Process the dictionary and store it in a CSV (comma separated values) file
- reference Figure 10.7.2: Reading each row of a csv file.

Introduction

Weather statistics are used for a variety of prediction metrics varying from future weather to cyclones. Although the exact amount of data required to predict a weather pattern lies in tens of giga bytes, in this lab, we will only deal with a simple dictionary.

Anita and Chen are meteorologists who want to predict weather in Santa Barbara using the past weather information.

You will need to help Anita read the CSV data from file, extract the necessary fields, and convert the weather data to a Python dictionary, which will be later given to Chen.

Instructions

Anita's information is unfortunately stored as a table that shows the 30-year temperature averages for Santa Barbara, 1991 to 2020. It looks like this:

High °F	Low °F		High °C	Low °C
67	47	January	19	8
67	47	February	19	9
68	49	March	20	10
71	51	April	22	11
72	54	May	22	12

High °F	Low °F		High °C	Low °C
73	57	June	23	14
76	60	July	25	16
78	60	August	25	16
78	60	September	25	15
76	56	October	24	13
71	50	November	22	10
66	46	December	19	8
72	53	Year	22	12

She saved it as a csv file but she needs just the first 3 columns - temperatures in Fahrenheit and the corresponding month name.

As tempting as it was to continue working in Excel and modifying the columns there, Anita knows that she needs a reproducible workflow.

If they need to analyze 30 years worth of data, year by year, having a script to do it automatically for them is the best solution.

She asks you for help: can you convert this information into a Python dictionary whose keys are the full names of the months (e.g.: `"January"`) and values are list objects with 2 elements?

This request makes you happy that you read zyBooks Sections 10.6 and 10.7!

File and dictionary structure

For the two list elements, the first element is the maximum and the second element is the minimum temperature in Fahrenheit - the first two columns of the file. For instance, one item in the dictionary will have the key `"January"` and the value `[67, 47]`. You need to create a dictionary that contains `12` such items, e.g.,

```
temperatures_dict = {
    "January": [67, 47],
    ...
}
```

Steps

1. Create a function `get_temperature_stats(filename)` that accepts a string with the filename from which to load the data (includes the `.csv` extension). The function

returns a dictionary with the month-temperature mappings: the key are the values in the 3rd column, the list values are the first two columns of the file.

2. At the start of the function, add the line `import csv` and define a new empty dictionary, which will hold the processed values.
3. Use the `with` construct to open the file with the required filename. Follow the steps in the **Figure 10.7.2: Reading each row of a csv file**.
4. After defining `csv.reader`, loop over all items in the file using a `for` loop.
5. Assume that the file will be properly formatted with at least 3 columns in it.

   5.1. For each row: The first two elements are the value in the dictionary; the third element is the month name, which is the key. For instance, a row that contains `['67', '47', 'January', '19', '8']` needs you to extract the key as `"January"` and the corresponding dictionary value `[67, 47]`.

   5.2. Extract the elements from the row, remembering to correctly convert the temperatures into integers.
6. Return the resulting dictionary.

In your main program, i.e, in the `if __name__ == "__main__"` block, you should ask for a filename that stores the previous weather information provided to Anita. You should then call the function `get_temperature_stats(...)` to get the dictionary from the csv file.

398092.2571084.qx3zqy7

LAB ACTIVITY | 10.8.1: LAB CHECKPOINT: Load from CSV | 0 / 1

Downloadable files

`temperatures.csv` **Download**

main.py — Load default template...

```
def get_temperature_stats(filename):
    """
    """

if __name__ == "__main__":
    tfile = input()
    result = get_temperature_stats(tfile)
    print(result)
```

Develop mode | Submit mode

Run your program as often as you'd like, before submitting for grading. Below, type any needed input values in the first box, then click **Run program** and observe the program's output in the second box.

Enter program input (optional)

If your code requires input values, provide them here.

Run program

Input (from above) → **main.py** (Your program)

Program output displayed here

Coding trail of your work What is this?

```
History of your effort will appear here once you begin
working on this zyLab.
```

Trouble with lab?