Due Wednesday by 11:59pm **Points** 100 **Available** after Oct 24 at 6am

Home (https://northeastern.instructure.com/courses/156930/pages/landing-page)

Resources (https://northeastern.instructure.com/courses/156930/pages/resources

Homework 6: Dictionaries, Files, Exception Handling

Programming Component (100% of this HW)

Just two programs this week. There is NO written.txt to submit!

Note the due date for this assignment is Wednesday, 08-Nov-2023. You have 2 weeks from the release of this assignment to submit it.
Focus on Files, Exception Handling and Dictionaries

Program 1: Emojini

This assignment should make you :-) LOL!

For this program, you'll be implementing a text transformation solution that substitutes English words into their appropriate text-based emojis. There are a few sets of emojis in use, the most common ones are "Western" style (prevalent in North America and Europe) and "Eastern" Kaomoji (used more in Japan and other APAC areas). Our solution will handle the transformation for both of these text-based emoji sets.

Files:

- emojini.py
- **Starter files** emoji_directives.txt (https://northeastern.instructure.com/courses/156930/files/22372098?wrap=1) (https://northeastern.instructure.com/courses/156930/files/22372098/download?download_frd=1) emojis.txt (https://northeastern.instructure.com/courses/156930/files/22372167?wrap=1) (https://northeastern.instructure.com/courses/156930/files/22372167/download?download_frd=1) recommendation_letter_k_orig.txt (https://northeastern.instructure.com/courses/156930/files/22372179?wrap=1) (https://northeastern.instructure.com/courses/156930/files/22372179/download?download_frd=1) recommendation_letter.txt (https://northeastern.instructure.com/courses/156930/files/22372180?wrap=1) (https://northeastern.instructure.com/courses/156930/files/22372180/download?download_frd=1) supercool.txt (https://northeastern.instructure.com/courses/156930/files/22372009?wrap=1) (https://northeastern.instructure.com/courses/156930/files/22372009/download?download_frd=1) text_to_friend_k.txt (https://northeastern.instructure.com/courses/156930/files/22371961?wrap=1) (https://northeastern.instructure.com/courses/156930/files/22371961/download?download_frd=1)

Task Overview:

You will be creating a solution to transform existing text into their text-based emoji representations, and vice versa. We are providing a text file with the mapping between English and the emoji character set you are to use for this assignment. Our emoji file uses metadata, so you will need to design your solution to handle that style of data representation.

Note: For this assignment, the first line in the data file is metadata. It describes the columns of data provided. Your program should be flexible enough to work EVEN IF WE GIVE AN ALTERNATE FILE with similar data, as long as the metadata is present.

The metadata line look like this: **METADATA ENGLISH WESTERN KAOMOJI**
The word **METADATA** indicates that this line is NOT traditional data. The next 3 (or more, if we add more emoji sets) columns describe the sequence of the emoji information in the rest of the file. Therefore, for each line of this font file, English words are in the first column, Western-style emojis are in the second column and Kaomoji-style emojis are in the third column.

Important Note: As we discussed in lecture, metadata is "data about data". Do NOT rely on the positioning of the columns given here as an example. We will freely move the columns and use different metadata files for this assignment. If you hardcode column positions, your solution will NOT pass our autograder tests!

Design and write the following function:

As part of your solution, you **must** provide the following function:

```
batch_translate(emoji_file_name: str, directives_file_name: str)
```

This function takes two strings:
(1) the name of the emoji mapping file (we've given you one with emojis.txt) and
(2) a directives file which gives instructions for the type of transformation to run, and then orchestrates the process of converting the text as specified. Our auto-test suite will call this function to test your solution, so whatever process needs to be initiated to perform the text transformation should be performed when this function is called.

Do this:

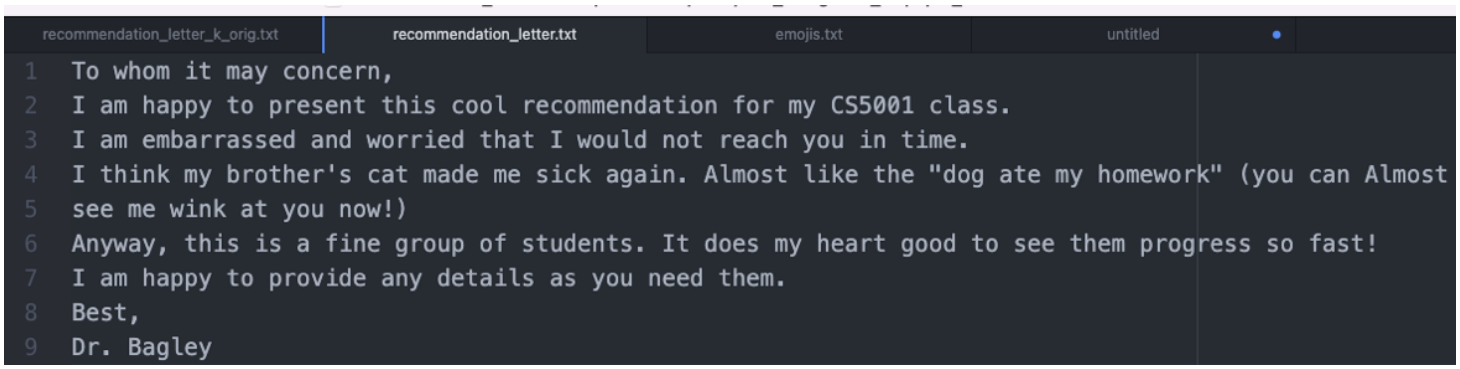
- Write a program that translates English words to their Western or Kaomoji equivalents. Use the given emojis.txt as the basis for your translation. This is whole-word replacement, similar to what you accomplished with Homework 5. Unlike that homework, however, you should NOT be stripping punctuation from your finished transformation. We'll be using your

transformation program as an automated responder to text messages and to create form letters (think mail-merge) and we need to retain the original punctuation in those messages.

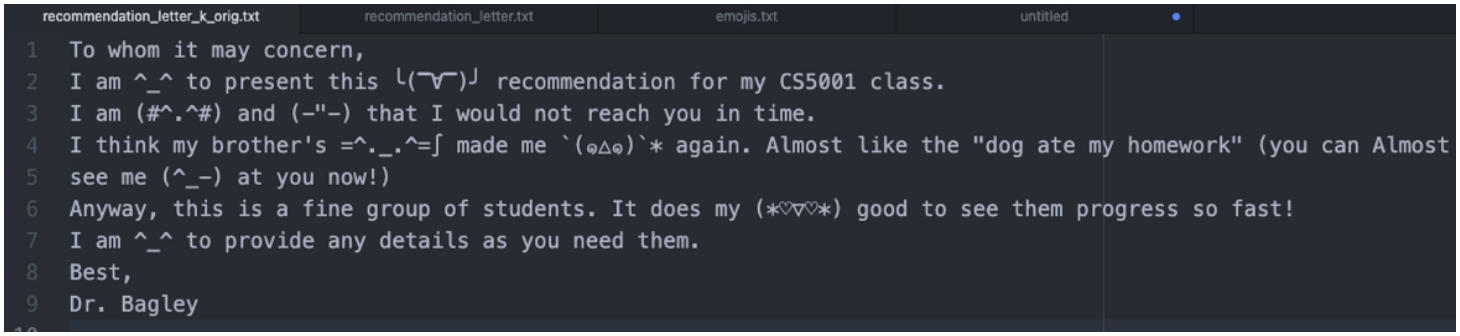
- Your program does not require any user interface. Your output must be written to files. Minimal print-outs to the screen are allowed (and appreciated) to indicate that your file processing has completed, or that you've handled an exception of some type. However, we don't want to see paragraphs of text printed out, and certainly no `input()` from the keyboard.
  For example, here is our output after processing the directives file

```
omework/HW6/emojini/emojini.py
Processing recommendation_letter.txt: english -> western
Processing recommendation_letter.txt: english -> kaomoji
Processing recommendation_letter_k_orig.txt: kaomoji -> english
Processing text_to_friend_k.txt: kaomoji -> english
Processing supercool.txt: english -> kaomoji
Processing supercool.txt: english -> western
done
>>>
```

- Your program must read from a text file that gives instructions on the emoji substitution required, the input file to use as the source information AND the output file you should generate for your output. We've included one called `emoji_directives.txt` but your program should be able to handle ANY file we pass it when we call the appropriate function.
  - For example, the `emoji_directives.txt` starter file given gives you the transformations you must process in sequence. In one case we are asking for a transformation from English to kaomoji for recommendation_letter.txt. The source file: recommendation_letter.txt is given to you as a starter file. Your code must produce the output file recommendation_letter_k.txt



Source file



Output file after transformation is performed

DO NOT do this:

- **Do not hardcode** your solution based on the current file. We will use different directives files and different emoji font files during our testing
- **Do not** forget to handle any file exceptions using Python's exception handling mechanism (try/except) and/or through the use of "defensive programming". We may "accidentally" give you a locked or non-existent file to process in our directives.txt. Your code should be able to handle that situation gracefully

Note: This homework - in addition to giving you practice with files & exception handling (and possibly dictionaries, if you use them) - gives you a taste of what is called "Process Orchestration", batch processing, and data-driven programming. Essentially, the file emoji_directives.txt "drives" how the program behaves, so that the same code you've written may give us different outcomes when we simply change the incoming data files. In larger & more complex programs, a concept similar to our directives.txt may include business rules and other "process flow" directions that are run by complex "process orchestration engines". The approach is also used by smaller teams and practitioners by giving non-technical subject-matter-experts a means of building small situational applications without the assistance of development teams working on larger projects.

Program 2: Hyperspace BnB