CSE 231 Spring 2022

Programming Project #9

Assignment Overview
In this assignment, you will practice with dictionaries.

This assignment is worth 55 points (5.5% of the course grade) and must be completed and turned in **before 11:59 on Monday, 04/11/2022**
A 3pts extra credit will be awarded if submitted by 11:59PM EST on Sunday April 10, 2022.

Background
Computer Vision is an important area of computer science. Autonomous vehicles process images to determine the surrounding environment for making decisions on where to go and when to stop. Recent years have shown a spike of *deep learning* in computer vision applications, namely for self-driving cars, robotics, medical imaging, and many others. Data and computing power play an important role in this process.

MSCOCO (Common Objects in Context) in one of the most popular image datasets out there, with applications like object detection, segmentation, and captioning. We will use a subset of the dataset. The images have bounding boxes around objects in the images as shown in Figure 1 and items are identified, e.g. "2 dogs and one suitcase". Finally, one or more captions exist. Therefore, each image will have an ID number and three pieces of data: a number of bounding boxes around features, identification of features, and captions.

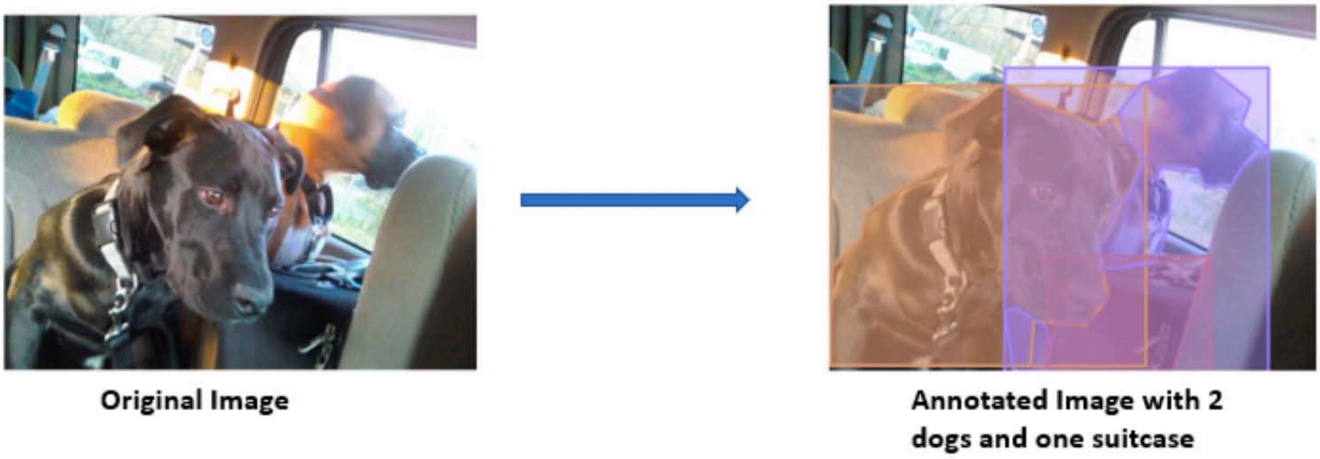

Figure 1MSCOCO Example (img_id: 580255)

A popular way of organizing and sharing data is to use JSON objects. As with CSV files, we will use the json package to read these files. One nice feature of the JSON organization is that it maps to Python dictionaries.

When the MSCOCO file is read in, it will be a dictionary with the image ID (`str`) as the key and the value as a dictionary with three keys (`str`): `'bbox_list'`, `'bbox_category_label'`, and `'cap_list'`. Each has a list as its value. For example:

```
'161141': {'bbox_list': [[35.06, 263.95, 88.84, 291.13],
```

```
   [125.81, 162.37, 279.57, 463.48],
   [176.4, 0.0, 334.0, 379.78],
   [10.89, 191.89, 79.52, 252.94]],
  'bbox_category_label': [18, 2, 7, 2],
  'cap_list': ['a red bicycle with a basket on front next to a train.',
   'a red bike is leaning against a blue wall',
   'the bike is parked next to the broken down bus.',
   'a bicycle leans against a standing train car.',
   'a bike that is parked next to a building']}
```

The values in the `'bbox_category_label'` list can be identified from a mapping in a file named `category.txt` with each line an int and a string. For example, the `'bbox_category_label'` in image `'161141'` above has the list `[18, 2, 7, 2]` which has these entries in the `category.txt` file:

```
18    dog
 2    bicycle
 7    train
```

So you know that the image contains a dog, a bicycle, a train, and another bicycle.
Note that the image key `'161141'` is a string of digits which we need to deal with when sorting.

Our focus in this project will be the values in the `'bbox_category_label'` list and the `'cap_list'` list. We will not be doing anything with the `'bbox_list'`.

Requirements:

1. You will write a program that reads a JSON file, puts the data into a master dictionary of dictionaries, and has some functions that will manipulate that `master_dictionary`. There will be a main program that loops, prompting the user for choices.
2. You have to use the eight functions plus `main()` in the provided `proj09.py` skeleton. We also provide a `strings.txt` file which contains all the strings that your need for the input and print statements to match Mimir tests. You have to implement and use them as specified:
    a. `get_option():→ str`
        i. Prompt the user for a valid option (use the `MENU` variable defined in the starter code) and return it as a string. If an invalid option is input, an error message is displayed, and the user is re-prompted until a valid one is input. The user input can be input as upper or lowercase, but it should be converted to lowercase before being returned. See the provided `proj09.py` skeleton and the `strings.txt` file for prompts and error message strings. A valid input should be one of these characters: `OPTIONS = "cfimwq"`
        ii. Parameters: none
        iii. Returns: `str` (lower case)
        iv. Displays: prompts & error messages

    b. `open_file(s)→ file pointer`
        i. Prompts for a file name and continues to prompt until a file is correctly opened. The prompt will be customized using the parameter (string). See the provided `strings.txt` file for prompts and error message strings. Hint: use `try-except`. The input s (a string) will input between {} in the formatted string:
        `"Enter a {} file name: "`
        ii. Parameters: `string`
        iii. Returns: `file pointer`
        iv. Displays: prompts & error messages

c. `read_annot_file(fp1)→ dictionary of dictionaries`
   i. Read the JSON file referenced by the `fp1` parameter. Use the `json` module's `load` method: `json.load(fp1).` The `load()` method does *all* the work for you so this is a one-line function. Make sure that you have the import statements like in the starter code. The obtained dictionary will be a dictionary with the image ID (`str`) as the key and the value as a dictionary with three keys (`str`): `'bbox_list'`, `'bbox_category_label'`, and `'cap_list'`. Each has a list as its value.
   ii. Parameters: `file pointer`
   iii. Returns: `dictionary of dictionaries`
   iv. Displays: nothing

d. `read_category_file(fp2) → dictionary`
   i. The category file is a text file where each line is space separated:
   `int  string`
   Create a dictionary whose key is the `int` and whose value is the `string`. Assume that the file is well-formed so that no key appears twice (Hint: this makes building the dictionary easy.)
   ii. Parameters: `file pointer`
   iii. Returns: `dictionary`
   iv. Displays: nothing

e. `collect_catogory_set(D_annot,D_cat)→ set of strings`
   i. `D_annot` dictionary is the dictionary returned from the `read_annot_file()` function) and The `D_cat` dictionary contains *all* the categories (as obtained from the `read_category_file()` function). This function creates a set of the categories actually used in the `D_annot` dictionary. The categories used are in the list under the `'bbox_category_label'` key for each image. Because the `D_annot` has dictionaries within dictionaries you need nested loops to get into them.
   Strategy: begin with separate code that will simply print all the `'bbox_category_label'` lists for all the images, i.e. iterate through all the images and for each image print its `'bbox_category_label'` list. Then add a loop to print each category number in the `'bbox_category_label'` list. The test function named `test_read_annot_small.py` that we provide has a small dictionary with only three images which is useful for testing. Once you can print each category number add them all to a set. Unfortunately, you are not done yet because the list needs to be the category names (strings) so you need to use the `D_cat` dictionary to get the names using the category integers.
   ii. Parameters: `dictionary of dictionaries, dictionary`
   iii. Returns: `set of strings`
   iv. Displays: nothing

f. `collect_img_list_for_categories(D_annot,D_cat,cat_set)→ dictionary of sorted lists`
   i. This function creates a mapping of each category to the list of images that has an instance of that category. For example, which images have a truck in them?
   `The category truck appears in the following images:`
   `47263, 70815, 187349, 351840, 395230, 569729`
   The key will be the category, e.g. `truck` (string), and the value will be a list of images (strings). The list of images will be sorted in increasing value, i.e. default sorting (note that the images are strings (of digits) so you are actually sorting strings).
   Strategy: as in the previous function, `collect_catogory_set`, you iterate

through the images and then the `'bbox_category_label'` lists. As you iterate through that list of categories, for each category you know the image you are in so append the image ID to the appropriate list of the dictionary that you are creating. As before, begin by writing separate code to print the `'bbox_category_label'` lists (which you already have from the previous function). Hint: we include `cat_set` which was returned from the previous function, `collect_catogory_set`; you can use `cat_set` to initialize your dictionary to be an empty list for each category making your code easier for adding to the dictionary. There is a lot of shared code between these two functions. Once you have your dictionary iterate through it to sort each list (sorting creates consistency for testing). Use the list `sort()` method rather than the `sorted()` function because the latter creates a new list and you want to sort the list that is in the dictionary.

ii. Parameters: `dictionary of dictionaries, dictionary, set of strings`
iii. Returns: `dictionary of sorted lists`
iv. Displays: nothing

g. `max_instances_for_item(D_image)→ tuple`
Find the most occurrences of an object (category) across all images. That is, if there are two dogs in an image, that counts as *two* occurrences. The parameter is the dictionary returned from the `collect_img_list_for_categories` function. Return a tuple `(int, string)`, e.g. `(7, 'dog').` (Those looking for a challenge: this can be done in one line using list comprehension.)
i. Parameters: `dictionary of sorted lists`
ii. Returns: `tuple`
iii. Displays: nothing

h. `max_images_for_item(D_image)→ tuple`
This function is almost identical to `max_instances_for_item`. In this function, find the most images that an object (category) appears in. That is, if there are two dogs in an image, that counts as *one* image that a dog appears in. Note that when a category shows up twice in an image, the image ID will show up twice in ID list of images that corresponds to that category in the `D_image` dictionary. The parameter is the dictionary returned from the `collect_img_list_for_categories` function. Return a tuple `(int, string)`, e.g. `(7, 'dog').`
(Those looking for a challenge: this can be done in one line using list comprehension.)
i. Parameters: `dictionary of sorted lists`
ii. Returns: `tuple`
iii. Displays: nothing

i. `count_words(D_annot)→ list of tuples`
i. Count the occurrences of words in captions. Returns a list of tuples of words and their count of the number of occurrences across all captions. The first element of the tuple is the count, and the second element is the word. The list should be sorted by count in reverse order (highest to lowest). If there is a tie, sort by word in reverse order also (from Z-A). Do not count uninteresting words, provided in the STOP_WORDS list. You can count numbers as words, but not as digits, e.g. count the string `'twenty-five'`, but not the number `'25'`. Also, strip words of punctuation so `'cat.'` becomes `'cat'`. (Hint: `import string` and use the `string.punctuation` value.) .
ii. This function is similar to the function in Section 9.2 Word Count Example in the text.

Strategy: similar to the suggested strategy above, write separate code that prints all the captions (`D_annot` have a key `'cap_list'` where the image caption is). Then print all the words in captions. Then incorporate a dictionary to count words (where the key is the word (string) and the value is the count of the number of occurrences across all captions): when you first encounter a word, add a dictionary entry with a count of 1. When you encounter a word that you already have in your dictionary, add 1 to its count. Then handle the rules about stripping punctuation and not counting certain words. Once you have the dictionary, you can create the sorted list of tuples. In Python Dictionary, `.items()` method is used to return the list of tuples with all dictionary keys with values.

iii. An example of returned list:
```
[(7, 'dog'), (5, 'man'), (5, 'kitchen'), (4, 'woman'), (3, 'next'), (3, 'lap'), (3, 'bike')]
```

iv. Parameters: `dictionary of dictionaries`
v. Returns: `list of tuples`
vi. Displays: nothing

j. `main()`:
   i. It's the main driver of the program. It doesn't have any parameters. It opens and reads a file and then it calls the other functions in a loop. Using `get_option` it gets an option that indicates which function to call.
   ii. Here is the basic structure.
      1. Display the heading `"Images\n".` The correct string format is in the `string.txt` file.
      2. Open and read a JSON file into a master dictionary of annotated images `D_annot`, a dictionary of dictionaries. Use the string `"JSON image"` as the argument to the `open_file()` function.
      3. Open and read a TXT file into a dictionary that maps category numbers to names `D_cat`, a dictionary of strings. Use the string `"category"` as the argument to the `open_file()` function.
      4. Find the categories used in the images file read in above.
      5. Build the dictionary that maps each category to the list of images that has an instance of that category by calling `collect_img_list_for_categories`
      6. Use `get_option` to prompt for an input character. Remember that the `get_option` function returns lower case version of the option.
      7. while not `'Q'` or `'q'` loop:
         call each appropriate function and display the result. Note that the messages are given in the `strings.txt`:
         a. if the option is `'c'` then display the category names in alphabetical order, separated by commas and a space. Note that you are displaying the categories actually used in the image file read in above; not all possible categories.
            Hint: build a list, sort it, then build a string with commas
         b. If the option is `'f'` to find images for a specified category, first print the sorted categories in alphabetic order (separated by commas and a space; categories used in the image file) so the user knows which categories they can choose. Then prompt the user for a category (loop until a valid category is entered). Finally, display the sentence with the choice embedded followed by a list of the unique images, in numerical order, separated by a space. Note that sorting the numbers as strings

will not put them in numerical order, e.g. the string `'74'` is greater than the string `'100'` because 7 > 1. So you must convert them to numbers, then sort, then print.

c. If the option is `'i'` , print the maximum instances of a category in the image file. See `strings.txt` for formatting. (hint: call function `max_instances_for_item`)

d. If the option is `'m'` , print the category that appears in the maximum number of images. See `strings.txt` for formatting. (hint: call function `max_images_for_item`)

e. If the option is `'w'` for word count, prompt for the number of words to display (loop until an integer greater than zero is entered), then display that many of the top words with the highest word counts.

Prompt for another input character.

8. Print the closing message:
   `'\nThank you for running my code.'`

3. Use Coding Standard 1-8

Deliverables

The deliverable for this assignment is the following file:

`proj09.py` -- your source code solution

Be sure to use the specified file name and to submit it for grading via Mimir before the project deadline.

Getting Started

- *Solve the problem using pencil and paper first.* You cannot write a program until you have figured out how to solve the problem. This first step can be done collaboratively with another student. Talk through each function and what it does. However, once the discussion turns to Python specifics and the subsequent writing of Python, you must work on your own.
- Use Anaconda Spyder to open the provided file (`proj09.py`).
- Write a simple version of the program, e.g. open, read the file, and print it so you can test on your own Spyder.
- Use the **Mimir** system to turn in the first version of your program. The tests are written so that if you start with the code we provide, functions will be tested individually.

- Next work on another function. The `get_option` is the easiest; `count_words` is the hardest.

- Cycle through the steps to incrementally develop your program:

  - Edit your program to add new capabilities.
  - Run the program and fix any errors.

- Use the **Mimir** system to submit your final version.

Example Interaction

Test 1

```
Images

Enter a JSON image file name: small.json
Enter a category file name: category_very_small.txt

    Select from the menu:
        c: display categories
        f: find images by category
        i: find max instances of categories
        m: find max number of images of categories
        w: display the top ten words in captions
        q: quit

    Choice: c

Categories:
bicycle, bottle, bowl, couch, cup, dog, microwave, oven, person, train, vase

    Select from the menu:
        c: display categories
        f: find images by category
        i: find max instances of categories
        m: find max number of images of categories
        w: display the top ten words in captions
        q: quit

    Choice: f

Categories:
bicycle, bottle, bowl, couch, cup, dog, microwave, oven, person, train, vase
Choose a category from the list above: cup

The category cup appears in the following images:
210012

    Select from the menu:
        c: display categories
        f: find images by category
        i: find max instances of categories
        m: find max number of images of categories
        w: display the top ten words in captions
        q: quit

    Choice: f

Categories:
bicycle, bottle, bowl, couch, cup, dog, microwave, oven, person, train, vase
Choose a category from the list above: dog

The category dog appears in the following images:
161141, 210012, 504900

    Select from the menu:
        c: display categories
        f: find images by category
```

```
        i: find max instances of categories
        m: find max number of images of categories
        w: display the top ten words in captions
        q: quit

    Choice: i

Max instances: the category bottle appears 6 times in images.

    Select from the menu:
        c: display categories
        f: find images by category
        i: find max instances of categories
        m: find max number of images of categories
        w: display the top ten words in captions
        q: quit

    Choice: m

Max images: the category dog appears in 3 images.

    Select from the menu:
        c: display categories
        f: find images by category
        i: find max instances of categories
        m: find max number of images of categories
        w: display the top ten words in captions
        q: quit

    Choice: w

Enter number of desired words: 4

Top 4 words in captions.
word           count
dog                7
man                5
kitchen            5
woman              4

    Select from the menu:
        c: display categories
        f: find images by category
        i: find max instances of categories
        m: find max number of images of categories
        w: display the top ten words in captions
        q: quit

    Choice: q

Thank you for running my code.
```

Test 2

```
Images

Enter a JSON image file name: small_img2bbox.json
Enter a category file name: category.txt

    Select from the menu:
        c: display categories
        f: find images by category
        i: find max instances of categories
        m: find max number of images of categories
        w: display the top ten words in captions
        q: quit

    Choice: c

Categories:
apple, backpack, bench, bicycle, bird, boat, book, bottle, bowl, bus, car, cat, cell,
chair, clock, couch, cup, dining, dog, fire, frisbee, handbag, hot, knife, laptop,
microwave, motorcycle, orange, oven, parking, person, potted, refrigerator, sink, spoon,
sports, stop, suitcase, toilet, traffic, train, truck, umbrella, vase, wine

    Select from the menu:
        c: display categories
        f: find images by category
        i: find max instances of categories
        m: find max number of images of categories
        w: display the top ten words in captions
        q: quit

    Choice: f

Categories:
apple, backpack, bench, bicycle, bird, boat, book, bottle, bowl, bus, car, cat, cell,
chair, clock, couch, cup, dining, dog, fire, frisbee, handbag, hot, knife, laptop,
microwave, motorcycle, orange, oven, parking, person, potted, refrigerator, sink, spoon,
sports, stop, suitcase, toilet, traffic, train, truck, umbrella, vase, wine
Choose a category from the list above: cup

The category cup appears in the following images:
57936, 158497, 200365, 209468, 210012, 427639, 558840, 560830

    Select from the menu:
        c: display categories
        f: find images by category
        i: find max instances of categories
        m: find max number of images of categories
        w: display the top ten words in captions
        q: quit

    Choice: f

Categories:
apple, backpack, bench, bicycle, bird, boat, book, bottle, bowl, bus, car, cat, cell,
chair, clock, couch, cup, dining, dog, fire, frisbee, handbag, hot, knife, laptop,
microwave, motorcycle, orange, oven, parking, person, potted, refrigerator, sink, spoon,
sports, stop, suitcase, toilet, traffic, train, truck, umbrella, vase, wine
Choose a category from the list above: dog
```

```
The category dog appears in the following images:
74, 4678, 7125, 13882, 16164, 20671, 26654, 32054, 33405, 37017, 47263, 49097, 57703,
57936, 70815, 75283, 79966, 98194, 116061, 145544, 151988, 153692, 154589, 155997,
158497, 161141, 173350, 174871, 176474, 176697, 178939, 184771, 185156, 187349, 191280,
194306, 205350, 205573, 209468, 210012, 212545, 225919, 233660, 235126, 236954, 239985,
251716, 252203, 263146, 270744, 278435, 278550, 282134, 283119, 307993, 325969, 329011,
346071, 346965, 351840, 354537, 357542, 361022, 364010, 369190, 370210, 373193, 376608,
378163, 378561, 382406, 383780, 386165, 390348, 392659, 395230, 400728, 408307, 412764,
424378, 426342, 427639, 458424, 459566, 471175, 495357, 504900, 511463, 514915, 519838,
535483, 537955, 540449, 555669, 559950, 560830, 569729, 580255

    Select from the menu:
        c: display categories
        f: find images by category
        i: find max instances of categories
        m: find max number of images of categories
        w: display the top ten words in captions
        q: quit

    Choice: i

Max instances: the category person appears 175 times in images.

    Select from the menu:
        c: display categories
        f: find images by category
        i: find max instances of categories
        m: find max number of images of categories
        w: display the top ten words in captions
        q: quit

    Choice: m

Max images: the category dog appears in 98 images.

    Select from the menu:
        c: display categories
        f: find images by category
        i: find max instances of categories
        m: find max number of images of categories
        w: display the top ten words in captions
        q: quit

    Choice: w

Enter number of desired words: 10

Top 10 words in captions.
word           count
dog              336
car               92
man               91
sitting           83
dogs              83
two               68
woman             56
bike              52
black             45
```

```
next              43

    Select from the menu:
        c: display categories
        f: find images by category
        i: find max instances of categories
        m: find max number of images of categories
        w: display the top ten words in captions
        q: quit

    Choice: q

Thank you for running my code.
```

Test 3: error checking

```
Images

Enter a JSON image file name: abc
File not found.  Try again.
Enter a JSON image file name: small.json
Enter a category file name: xyz.t
File not found.  Try again.
Enter a category file name: category.txt

    Select from the menu:
        c: display categories
        f: find images by category
        i: find max instances of categories
        m: find max number of images of categories
        w: display the top ten words in captions
        q: quit

    Choice: z
Incorrect choice.  Please try again.

    Select from the menu:
        c: display categories
        f: find images by category
        i: find max instances of categories
        m: find max number of images of categories
        w: display the top ten words in captions
        q: quit

    Choice: C

Categories:
bicycle, bottle, bowl, couch, cup, dog, microwave, oven, person, train, vase

    Select from the menu:
        c: display categories
        f: find images by category
        i: find max instances of categories
        m: find max number of images of categories
        w: display the top ten words in captions
        q: quit

    Choice: F
```

```
Categories:
bicycle, bottle, bowl, couch, cup, dog, microwave, oven, person, train, vase
Choose a category from the list above: tiger
Incorrect category choice.
Choose a category from the list above: car
Incorrect category choice.
Choose a category from the list above: cup

The category cup appears in the following images:
210012

    Select from the menu:
        c: display categories
        f: find images by category
        i: find max instances of categories
        m: find max number of images of categories
        w: display the top ten words in captions
        q: quit

    Choice: f

Categories:
bicycle, bottle, bowl, couch, cup, dog, microwave, oven, person, train, vase
Choose a category from the list above: dog

The category dog appears in the following images:
161141, 210012, 504900

    Select from the menu:
        c: display categories
        f: find images by category
        i: find max instances of categories
        m: find max number of images of categories
        w: display the top ten words in captions
        q: quit

    Choice: i

Max instances: the category bottle appears 6 times in images.

    Select from the menu:
        c: display categories
        f: find images by category
        i: find max instances of categories
        m: find max number of images of categories
        w: display the top ten words in captions
        q: quit

    Choice: m

Max images: the category dog appears in 3 images.

    Select from the menu:
        c: display categories
        f: find images by category
        i: find max instances of categories
        m: find max number of images of categories
        w: display the top ten words in captions
        q: quit
```

```
    Choice: w

Enter number of desired words: 4

Top 4 words in captions.
word           count
dog                7
man                5
kitchen            5
woman              4

    Select from the menu:
        c: display categories
        f: find images by category
        i: find max instances of categories
        m: find max number of images of categories
        w: display the top ten words in captions
        q: quit

    Choice: q

Thank you for running my code.
```

Scoring Rubric

Computer Project #4 Scoring Summary

General Requirements

__0__ (4 pts) Coding standard 1-9
(descriptive comments, function headers, mnemonic identifiers, format, etc...)

Program Implementation

__0__ (4 pts) Function Test get_option (no Mimir test)

__0__ (4 pts) Function Test open_file (no Mimir test)

__0__ (3 pts) Function Test read_annot_file

__0__ (4 pts) Function Test read_category_file

__0__ (5 pts) Function Test collect_category_set

__0__ (5 pts) Function Test collect_img_list_for_categories

__0__ (4 pts) Function Test max_instances_for_item

__0__ (4 pts) Function Test max_images_for item

__0__ (6 pts) Function Test count_words

__0__ (4 pts) Test case 1

__0__ (4 pts) Test case 2

__0__ (4 pts) Test case 3

Note:
1. hard coding an answer earns zero points for the whole project .
2. (-10) points for not using main()