Important

1. Due Date: **Thursday, October 26th**
2. This homework is graded out of 100 points.
3. This is an Individual Assignment. You may collaborate with other students in this class. Collaboration means talking through problems, assisting with debugging, explaining a concept, etc. Students may only collaborate with fellow students currently taking CS 1301, the TA's and the lecturer. You should not exchange code or write code for others. For individual assignments, each student must turn in a unique program. Your submission must not be substantially similar to another student's submission. Collaboration at a reasonable level will not result in substantially similar code.
4. For Help:
    - TA Helpdesk (Schedule posted on class website.)
    - Email TA's or use Piazza Forums Notes:
    - How to Think Like a Computer Scientists [http://openbookproject.net/thinkcs/python/english3e/]
    - CS 1301 Python Debugging Guide [http://www.cc.gatech.edu/classes/AY2016/cs1301_spring/CS-1301-Debugging-Guide/index.html]
    - Don't forget to include the required collaboration statement (outlined on the syllabus). Failing to include the Collaboration Statement will result in no credit.
    - Do not wait until the last minute to do this assignment in case you run into problems.
    - Comment or delete all your function calls. When your code is run, all it should do is build without any errors. Having function calls or extraneous code outside the scope of functions will result in a lower grade. (import statements and global variables are okay to be outside the scope of a function)
    - **Read the entire specifications document before starting this assignment.**
    - **IF YOUR CODE CANNOT RUN BECAUSE OF AN ERROR, IT IS A 0%.**

Introduction

The goal of this homework is for you to showcase your knowledge about how to properly use dictionaries. For this assignment you will need to implement 5 functions. Refer to the rubric to see how points will be rewarded for each function. You have been given HW7.py to fill out with instructions in the docstrings. However, below you will find more detailed information to complete your assignment. Read it thoroughly before you begin. You have until **Thursday, October 26th** to complete this assignment. Don't forget to include your name and your collaboration statement. Re-download your submission from T-Square after you submit it to make sure that your code runs successfully.

Function name: `scheduler`
Parameters: a dictionary containing members' names as keys, and lists of months that member is available as values
Returns: the best month to schedule the meet-up (str)

Description: You're the event manager of the National Association of Python Coders, and you want to coordinate a big meet-up for your organization. Write a function called `scheduler` that takes in a dictionary with members' names as keys and lists of months during which the member is available to attend a meet-up, and returns whichever month has the most club members available. If there is more than one month with the highest number of available club members, return the month that occurs earlier in the year. NOTE: The list of months for each club member will contain integers 1-12; you should return the corresponding month name as a string ("January", "February", "March", etc.) Hint: It may be useful to use a dictionary for this step.

Test cases:

```
>>> info_dict_1 = {"Andrew Andrews": [1,4,7,8], "Brock Brockson": [1,6,8,12],
"Charles Charleston": [1,2,3], "David Davidson": [1,5,9,10], "Ellen
Ellenson": [1]}
>>> scheduler(info_dict_1)
'January'

>>> info_dict_2 = {"Flora Flores": [4,5,6,3], "George Georges": [4,5,6,3],
"Henry Henries": [4,5,6,3], "Isabella Isbell": [4,5,6,3], "John Johnson":
[4,5,6,3]}
>>> scheduler(info_dict_2)
'March'
```

Function name: `buy_albums`
Parameters: a dictionary named `album_buy_dict` {key is the name of an album (string): value is a the album's artist (string)}, a dictionary named `artist_price_dict` {key is name of an artist (string): value is the price per album of all that artist's albums (float)}
Returns: a dictionary {key is the name of an artist (string): value is how much was paid in total for all of the albums we bought that were theirs (float)}

Description: Write a function that takes in a dictionary in which every key is the name of an album, and every is a string of the name of the album's artist, as well as a dictionary in which every key is an artist's name, and every value is a float of the price per album for that artist for all their albums. The function should return a dictionary in which every key is the name of an artist whose album(s) you bought, and the value is the total amount you paid for the album(s) of that particular artist (i.e. price per that artist's albums * total number of albums bought). The dictionary should also have a key-value pair where the key is "total" and the value is the grand total for all of the albums you bought.

Notes:

- You can assume spelling and punctuation of the artist names in `album_buy_dict` and `artist_price_dict` will be the same for the same artist.
- Each album included in `album_buy_dict` means we bought one of that album.
- You can assume that “total” will never be a key in the original dictionary.
- If you print out your returned dictionary the order of the artists may be different from ours, what’s important is that each artist (key) is correctly matched to their price totals (value).

Test cases:

```
>>> total_buy_dict = buy_albums({"Coloring Book" : “Chance the Rapper”, "Acid
Rap" : “Chance the Rapper”, "The College Dropout" : “Kanye West”, “Yeezus” :
“Kanye West”, “My Beautiful Dark Twisted Fantasy” : “Kanye West”, “Channel
Orange” : “Frank Ocean”, "Blonde" : “Frank Ocean”, "Kauai" : “Childish
Gambino”, “Continuum” : “John Mayer”}, {“Chance the Rapper” : 0.00, “Kanye
West” : 9.75, “Frank Ocean” : 12.15, “Childish Gambino” : 8.45, “John Mayer”
: 4.75})
>>> print(total_buy_dict)
>>> {'Chance the Rapper': 0.00, 'Kanye West': 29.25, 'Frank Ocean': 24.30,
'Childish Gambino': 8.45, 'John Mayer': 4.75 'total': 66.75}
```

Function name: `translator`
Parameters: a filename (str), a string
Returns: a string

Description: Write a function that takes in a filename and a string, then uses the file to create a dictionary with English words as the keys and the corresponding words in another language as the values. In the string passed in, replace every word that is a key in the dictionary with the value that the key maps to and return that new string. NOTE: You can assume the file passed in will always be in the format specified in the test cases below, and that the string will be all lowercase.

Test cases:

french_dictionary.txt

```
hello, bonjour
my, mon
name, nom
is, est
```

```
>>> translator(“french_dictionary.txt”, “hello, my name is christine! my
favorite food is pizza.”)
'bonjour, mon nom est chresttine! mon favorite food est pizza'
```

german_dictionary.txt

```
i, ich
read, lebe
many, viele
books, Bucher
study, studiere
math, Mathematik
and, und
```

```
>>> translator(“german_dictionary.txt”, “i read many books. i study math and
computer science.”)
'ich lebe viele Bucher. ich studiere Mathematik und computer scichence.'
```

Function name: `average_rating`
Parameters: a dictionary named `movies_dict` {keys are the names of movies (string): values are a list of integer ratings from zero to ten (list of ints)}
Returns: a dictionary {keys are names of movies (string): values are that show’s average rating (float)}

Description: Write a function that takes in a dictionary and returns a dictionary. The function will take in a dictionary, `movies_dict`, where the key will be a string that represents the name of some movie and the value will be a list of integers representing ratings on a ten-point scale for that movie. The function should take this dictionary and make a new dictionary where the keys will be strings of the movie name and the values will be the average rating for that movie (rounded to 2 decimal points).

Notes:

- Remember to round the average rating of each movie to two decimal points!
- Every movie included in `movies_dict` will have at least one rating.
- Every rating is out of ten.
- If you print out your returned dictionary the order of the artists may be different from ours, what’s important is that each artist (key) is correctly matched to their price totals (value).

```
>>> movies_dict = {“Back to the Future” : [8, 7, 9, 10], “Independence Day” :
[8, 7, 5, 10, 10], “Men in Black” : [7, 6, 9], “Rush Hour” : [8, 9, 10, 10]}
>>> test1 = average_rating(movies_dict)
>>> print(test1)
{'Back to the Future': 8.5, 'Independence Day': 8.0, 'Men in Black': 7.33,
'Rush Hour': 9.25}
```

Function name: `sentence_stats`
Parameters: a string
Returns: a dictionary

Description: Make a function called `sentence_stats` that takes in a sentence (str) as a parameter and returns a dictionary containing the information detailed in the following table:

key	**value; value type**
uppercase	number of uppercase letters in the string; int
lowercase	number of lowercase letters in the string; int
vowels	number of uppercase and lowercase vowels (A, E, I, O, U) in the string; int
consonants	number of uppercase and lowercase consonants (including Y) in the string; int
numbers	number of numbers (0123456789) in the string; int
special_chars	number of special characters (i.e. non-letters and non-numbers, not including spaces) in the string; int
word_count	number of words in the string; int
most_common	most common character in the string other than the space character; str

Test cases:

```
>>> sentence_stats("Hello! My name is Christine 07/08")
{'uppercase': 3, 'lowercase': 19, 'vowels': 8, 'consonants': 14, 'numbers':
4, 'special_chars': 2, 'word_count': 6, 'most_common': 'e'}

>>> sentence_stats("!!!#$@%^^&$%^%^$@!%$#%$%&@%$#@%$@")
{'uppercase': 0, 'lowercase': 0, 'vowels': 0, 'consonants': 0, 'numbers': 0,
'special_chars': 33, 'word_count': 1, 'most_common': '%'}
```

Grading Rubric

```
- scheduler:            20 points
- buy_albums:           20 points
- translator:           20 points
- average_rating:       20 points
- sentence_stats:       20 points
___________________________________________________
- Total                 100/100 points
```

Provided

The following file(s) have been provided to you. There are several, but you will only edit one of them:

1. `HW7.py`
This is the file you will edit and implement. All instructions for what the methods should do are in the docstrings.

2. `HW7_test.py`
We have provided you with a python file called `HW7_test.py`. In this file we have created a series of tests for your usage. We understand you probably have never been exposed to testing code so you are not expected to do anything to this file or even use this file if you don't want to. However, we encourage you to use it as it will be highly beneficial in testing your code. Feel free to add your own tests to the file to cover any additional cases you would like to test.

If you do desire to test your code, all you have to do is have the `HW7.py` and the `HW7_test.py` files in the same directory. Open and run `HW7_test.py`. After running the test, you should see the results. Check the results and start debugging if needed. If you pass all the tests you should see something like this as your output:

Disclaimer: **The tests found in `HW7_test.py` are not intended to be an exhaustive list of all test cases and does not guarantee any type of grade. Write your own tests if you wish to ensure you cover all edge cases.**

Read more about unittest here: [https://docs.python.org/3/library/unittest.html]

Deliverables

You must submit all of the following file(s). Please make sure the filename matches the filename(s) below. Be sure you receive the confirmation email from T-Square, and then download your uploaded files to a new folder and run them.

1. HW7.py
If this file does not run (if it encounters an error while trying to run), you will get no credit on the assignment.