SMU SINGAPORE MANAGEMENT UNIVERSITY

IS111 – Lab 8
Dictionaries

Instructions

You should attempt the questions using Visual Studio Code. Download and extract the files from the **Lab8_starting_code.zip**. You should obtain several *.py and *.txt files to be used for some of the lab questions.

Q1: Substitution Cipher

"A **substitution cipher** is a method of encoding by which units of plaintext are replaced with ciphertext, according to a fixed system." See https://en.wikipedia.org/wiki/Substitution_cipher.

In this question, we consider a simple substitution cipher that replaces each letter in the original message with another letter.

Part (a) [★]

You are given a dictionary in `q1a_test.py` that stores the mappings from English lowercase letters to their replacements. Note that no two letters will be mapped to the same replacement.

In the given file `q1a.py`, define a function called `encrypt()` that takes in two parameters:

- `my_dict` (type: `dict`): a dictionary as described above.
- `msg` (type: `str`): a message consisting of only lowercase letters and spaces.

The function returns the encrypted message using `my_dict` as the substitution cipher. Note that a space remains untouched in the encrypted message.

Use the test cases provided to test your code.

Part (b) [★★]

In the given file `q1b.py`, define a function called `decrypt()`. This function also takes in two parameters:

- `my_dict` (type: `dict`): a dictionary as described above.
- `encrypted_msg` (type: `str`): an **encrypted** message.

The function returns the original message, i.e., the message that has been encrypted into the string `encrypted_msg`.

Use the test cases provided to test your code.

Q2: Prime Numbers [★★]

In the given file `q2.py`, define a function called `create_prime_dict()` that takes in a list of integers greater than 1 (without any duplicates) as its parameter. The function creates a dictionary that maps each integer in the list to a boolean value that indicates whether or not this integer is a prime number.

IS111 – Lab 8
Dictionaries

Recall that a prime number is a number that is divisible only by 1 and itself.

For example, `create_prime_dict([2, 3, 5, 10, 20, 23])` returns a dictionary that looks like the following: `{2:True, 3:True, 5:True, 10:False, 20:False, 23:True}`.

Use the test cases provided to test your code.

Q3: Facilities

You are given a file called "facilities.txt" that stores the information of some facilities in SMU. Each line of the file contains the following columns of information, separated by tabs.

```
school    room_number    capacity    has_projector
```

Note that different schools may have the same room number. For example, both SIS and SOE may have an MR-5.1. But within each school the room numbers are unique.

Part (a) [★★]

Create `q3a.py` and write a program that allows a user to continuously check the details of a room until the user chooses to quit the program.

A sample run of the program may look like the following:

```
Do you want to search for a facility? [Y|N] :Y
Enter the school [LKCSB|SOE|SOL|SOA|SIS] :SOE
Enter the room number :SR-3.1
SOE SR-3.1 has a capacity of 50 and has a projector.

Do you want to continue the search? [Y|N] :Y
Enter the school [LKCSB|SOE|SOL|SOA|SIS] :SIS
Enter the room number :MR-4.3
SIS MR-4.3 has a capacity of 15 and does not have a projector.

Do you want to continue the search? [Y|N] :Y
Enter the school [LKCSB|SOE|SOL|SOA|SIS] :SIS
Enter the room number :SR-3.1
SIS SR-3.1 has a capacity of 60 and has a projector.

Do you want to continue the search? [Y|N] :N
Thanks for using our system!
```

Part (b) [★★★]

Create `q3b.py` and write a program that allows a user to continuously search for all the facilities within a school until the user chooses to quit the program.

A sample run of the program may look like the following:

```
Do you want to search for the facilities within a school? [Y|N]:Y
Enter the school [LKCSB|SOE|SOL|SOA|SIS]:SIS
The following facilities are in SIS:
    Room #    Cap    Projector?
    ------    ---    ----------
    SR-3.1    60     yes
    SR-3.2    50     yes
    SR-3.3    50     yes
    CR-3.1    20     no
    CR-3.2    20     no
    CR-3.3    15     no
    MR-4.1    20     yes
    MR-4.3    15     no
    MR-4.4    30     yes

Do you want to search for the facilities within a school? [Y|N]:Y
Enter the school [LKCSB|SOE|SOL|SOA|SIS]:SOL
The following facilities are in SOL:
    Room #    Cap    Projector?
    ------    ---    ----------
    SR-2.1    60     yes
    SR-2.2    70     yes
    SR-2.3    65     yes
    CR-3.1    20     yes
    CR-3.2    20     yes
    CR-3.3    15     no
    MR-5.1    25     yes
    MR-5.2    20     no

Do you want to search for the facilities within a school? [Y|N]:N
Good-bye!
```

Hint: You should use a different dictionary than the one used in Part (a) for this question. Note that here the key is only a school name.

Part (c) [★★★]

Create `q3c.py` and write a program that allows a user to continuously search for the facilities with a mininum capacity within a school until the user chooses to quit the program.

A sample run of the program may look like the following:

```
Do you want to search for the facilities within a school? [Y|N]:Y
Enter the school [LKCSB|SOE|SOL|SOA|SIS]:SIS
Enter the minimum capacity you need:20
The following facilities with a capacity of 20 or above are found i
n SIS:
    Room #    Cap    Projector?
    ------    ---    ----------
    SR-3.1    60     yes
    SR-3.2    50     yes
    SR-3.3    50     yes
```

```
    CR-3.1    20    no
    CR-3.2    20    no
    MR-4.1    20    yes
    MR-4.4    30    yes

Do you want to search for the facilities within a school? [Y|N]:Y
Enter the school [LKCSB|SOE|SOL|SOA|SIS]:SOL
Enter the minimum capacity you need:50
The following facilities with a capacity of 50 or above are found i
n SOL:
    Room #    Cap    Projector?
    ------    ---    ----------
    SR-2.1    60    yes
    SR-2.2    70    yes
    SR-2.3    65    yes

Do you want to search for the facilities within a school? [Y|N]:N
Good-bye!
```

Hint: Do NOT include the capacity as part of a key. Here you should first retrieve all the facilities in the given school and then go through them one by one to find those satisfying the capacity requirement.

Q4: Students and Schools [★★★]

In the given file `q4.py`, define a function called `get_students_by_school`. The function takes in a dictionary as its parameter. This dictionary stores mappings from students to their corresponding schools. For example, the dictionary may contain `'Joe Tan':'SIS'` as a (key, value) pair.

The function returns a new dictionary that stores mappings from each school to a list of students in that school. For example, if in the original dictionary there are two students, "Joe Tan" and "Wendy Li", who are from SIS, then in the returned new dictionary, the value associated with the key `'SIS'` should be `['Joe Tan', 'Wendy Li']`.

Use the test cases provided to check your output.

Q5: Word Counts

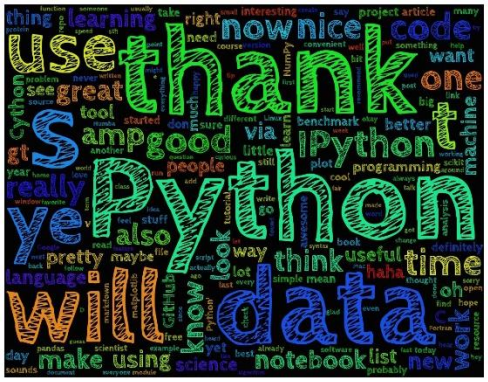
Background

In data analytics, in order to quickly get a gist of what a collection of articles is talking about, **word clouds** have been popularly used.

Below is an example word cloud:

SMU SINGAPORE MANAGEMENT UNIVERSITY

IS111 – Lab 8
Dictionaries



In a word cloud, a word shown in a bigger size has appeared more frequently inside the collection of articles being analyzed whereas a word shown in a smaller size has appeared less frequently.

In order to draw a word cloud, we need to first count the frequencies of each unique word inside the collection of articles that we want to analyze.

The Question

Part (a) [★★★]

In the given file `q5a.py`, define a function called `count_words_in_file()`. The function takes in as its parameter the name of a file, where each line of the file is the content of one article.

The function returns a dictionary that stores the mappings from all the unique words found in the file to their total counts in the file.

For example, if the word "the" has appeared 10 times the file, then the mapping `'the':10` should be stored inside the dictionary.

You need to turn everything into lowercases before counting. For example, "The" and "the" are both counted as `'the'` in the dictionary.

Simply use `line.split()` to break down a line into a list of words. You do not need to handle punctuation marks for this question.

We will now test your function with a file called "spam_sms.txt". This file stores many spam SMS messages. The data is obtained from the following link: https://www.kaggle.com/uciml/sms-spam-collection-dataset . Details about the data can be found here: http://www.dt.fee.unicamp.br/~tiago/smsspamcollection/ .

Use the test file provided to test your code.

Part (b) [★★]

Create `q5b.py` and write a Python program to display all the words in the dictionary whose lengths are at least 4 and whose frequencies are above 50.

Do you think these words can tell you something about a common theme in all these spam SMS messages?

You should get the following output:

```
free : 171
text : 110
have : 135
prize : 58
claim : 105
call : 336
your : 258
mobile : 109
with : 108
from : 127
send : 65
reply : 101
please : 52
nokia : 64
now! : 70
cash : 56
contact : 56
this : 79
stop : 85
only : 66
just : 78
```

Part (c) [★★★]

Create `q5c.py` and write a Python program for this question. Are you able to find the top-10 frequent words inside the dictionary? You're expected to get the following top words and their frequencies:

```
[('to', 684), ('a', 375), ('call', 336), ('your', 258), ('you', 237),
('the', 204), ('for', 196), ('or', 188), ('free', 171), ('2', 169)]
```

Q6: Employees [★★★★]

You are given a file called "employees.txt" that stores the information of some employees inside an organization. Each line of the file is about a single employee. For each employee, there are several pieces of information given. However, different employees may have different pieces of information given in the file. Take a look at the file to understand its structure.

You can assume that each line always contains the ID and name of an employee.

Create `q6.py` and write a prgram that allows a user to check the details of any employee. The user needs to first enter the employee's ID. If the ID is found, the name of the employee should be displayed first. After that, the program asks what information the user wants to know about the employee and displays the information accordingly. If the ID is not found or the requested information is not found, the program should display "Not found!"

See a sample run of the program below:

```
Enter an employee ID: 3
```

```
This employee is Peter Ng.
    Enter a field name or 'S' to stop: title
    The title is manager.
    Enter a field name or 'S' to stop: marriage status
    The marriage status is married.
    Enter a field name or 'S' to stop: phone
    Not found!
    Enter a field name or 'S' to stop: S

Do you want to search for another employee? [Y|N]:Y
Enter an employee ID: 4
Not found!

Do you want to search for another employee? [Y|N]:Y
Enter an employee ID: 10
This employee is George Khoo.
    Enter a field name or 'S' to stop: title
    Not found!
    Enter a field name or 'S' to stop: phone
    The phone is 67893456.
    Enter a field name or 'S' to stop: S

Do you want to search for another employee? [Y|N]:N

Good-bye!
```

Hint: You should store the data in an appropriate dictionary such that when you search for the information of an employee, you only need to do dictionary lookup and no for-loops.