**Week 12 In-Class Exercises**

**Note:** Please download the file “Week 12 In-Class Exercises Starting Code (v1.0H).ipynb”.

## Q1: Count Strings [ \*\* ]

Write a program that prompts the user continuously for strings until the user enters an empty string. The program then prints out the number of strings entered for each starting character that has been observed. Note that for starting characters that are letters, uppercase and lowercase letters are merged. For the final output, it doesn’t matter in what order the numbers for different starting characters are shown. A sample run of the program is shown below:

Enter a string > Singapore

Enter a string > management

Enter a string > school

Enter a string > SIS

Enter a string > $500

Enter a string > 3 apples

Enter a string > Python

Enter a string > programming

Enter a string >

You've entered

3 strings starting with 'S' or 's'

1 string starting with 'M' or 'm'

1 string starting with '$'

1 string starting with '3'

2 strings starting with 'P' or 'p'

Note the following:

* In the output, if the count is 1, you should display '1 **string** starting with...' instead of '1 **strings** starting with...'.
* If the first character is a letter, you should display "…starting with 'X' or 'x'" (where X is a one of the 26 English letters).

## Q2: Count Emails [ \*\* ]

You’re given a file that contains the information of email exchanges among a group of people on a particular day. Each line of the file represents a single email and contains the following four columns:

|  |
| --- |
| time\_stamp sender recipient has\_attachment |

The different columns are separated by a single '\t' character. The format of the four columns are as follows:

* time\_stamp: This is shown in either one of the following two formats: (1) hh:mm:ss , where hh is between 00 and 23, mm and ss are between 00 and 59. (2) hh:mm:ssAM or hh:mm:ssPM, where hh is between 01 and 12, and mm and ss are still between 00 and 59.

For example, '13:50:00' and '01:50:00PM' both represent 1:50 in the afternoon. '00:30:00' and '12:30:00AM' both represent 12:30 past midnight.

* sender: This is the sender’s email address.
* recipient: This is the recipient’s email address. We assume that each email has only a single recipient.
* has\_attachment: This column is either 'Yes' or 'No', indicating whether the email has an attachment or not.

The rows of the file are NOT sorted in any order.

Define a function called count\_emails(). This function takes in the following parameters:

* file\_name (type: str): The name of a file of the format described above.
* start\_time (type: tuple): This tuple contains two integers and represent a time stamp, where the first integer is between 0 and 23 (both inclusive) and represents the hour value, and the second integer is between 0 and 59 (both inclusive) and represents the minute value. It is assumed that the number of seconds is always 0. E.g., the tuple (23, 59) represents '23:59:00' or '11:59:00PM'.
* end\_time (type: tuple): This tuple also contains two integers in the same format as start\_time. The tuple represents an end time.
* sender (type: str): A string representing a sender’s email address.
* has\_attachment (type: bool): A bool value (True or False) representing having attachment or not.

The function should return the number of emails that (1) are sent by the specified sender, (2) are sent between the start\_time and the end\_time (both inclusive), and (3) have attachment status according to the specified has\_attachment parameter.

You can assume that the end time is always after the start time.

For example, if the file 'emails-1.txt' contains the following lines:

|  |
| --- |
| 12:30:00 [joe@abc.com](mailto:joe@abc.com) [jerry@abc.com](mailto:jerry@abc.com) Yes 13:20:00 [joe@abc.com](mailto:joe@abc.com) [john@abc.com](mailto:john@abc.com) No 12:45:30PM [joe@abc.com](mailto:joe@abc.com) [bob@abc.com](mailto:bob@abc.com) Yes 01:59:59PM [jerry@abc.com](mailto:jerry@abc.com) [cindy@gmail.com](mailto:cindy@gmail.com) Yes 02:00:01PM [joe@abc.com](mailto:joe@abc.com) [cindy@gmail.com](mailto:cindy@gmail.com) Yes 12:35:00AM [joe@abc.com](mailto:joe@abc.com) [jerry@abc.com](mailto:jerry@abc.com) Yes 02:00:00PM [joe@abc.com](mailto:joe@abc.com) [adam@xyz.edu](mailto:adam@xyz.edu) Yes |

Then count\_emails('emails-1.txt',(12, 30),(14, 0),'joe@abc.com',True) will return the number of emails with attachment sent by [joe@abc.com](mailto:joe@abc.com) between 12:30:00 and 14:00:00 (both inclusive), or in other words, between 12:30:00PM and 02:00:00PM (both inclusive). This will give us 3, corresponding to the 1st row, the 3rd row and the last row of the file.

## Q3: HTML [ \*\*\* ]

Webpages are in HTML format. In an HTML file, texts surrounded by tag pairs <b> and </b> will be shown in bold font on a Web browser.

For example, given the following text in HTML format:

|  |
| --- |
| This is an <b>example</b> of some <b>text</b> in HTML. <b>Some of the segments</b> will be <b>shown in bold</b>. |

The text segments 'example', 'text', 'Some of the segments' and 'shown in bold' will be bold.

Define a function called extract\_bold\_texts(). The function takes in a single string called html\_text. It returns a list of strings that are segments inside html\_text which are surrounded by <b> and </b>.

E.g.,

* extract\_bold\_texts('<b>ABC</b> abc <b>def 123 </b><b></b>0000') returns ['ABC', 'def 123 ', '']. (Note that the last string is an empty string.)
* extract\_bold\_texts('A piece of text without tags.') returns [].