Trial Lab Test 2 v2 [45 minutes, in-class]

Q1 [*]

Implement a function called `count_high_temperatures()` inside q1.py. This function takes in a list of numbers representing a sequence of body temperature readings. The function returns the number of temperature readings in the input list that are **strictly higher** than 37.5.

- Example #1: `count_high_temperatures([36.9, 37.6, 37.2, 37.1, 38.1])` returns 2
- Example #2: `count_high_temperatures([])` returns 0
- Example #3: `count_high_temperatures( [37.9, 38.2, 38.8, 37.5, 37.3, 37.0])` returns 3

Run q1_test.py to test your code.

Q2: [**]

In the file q2.py, write a program that prompts the user for his/her email address. You can assume that the user enters a single word (a string without any space). Keep prompting the user until **the input string contains a valid email address with the substring "@smu.edu.sg" at the end**.

A sample run of the program look as follows. Text in bold font is user input.

```
Please enter your SMU email address:abc@gmail.com
Invalid!
Please enter a valid SMU email address:www@sis.smu.edu.sg
Invalid!
Please enter a valid SMU email address:ma@jack@smu.edu.sg
Invalid!
Please enter a valid SMU email address:jack@smu.edu.sg
Thanks!
```

There is no testing code for Q2. Try your code with the sample input above.

Q3: [**]

Define a function called `check_age()` that takes in a list of tuples as its parameter. Call this list `student_list`. Each tuple in this list represents a student and consists of three elements: name (type: `str`), gender (type: `str`, either `'M'` or `'F'`) and age (type: `int`). For example, `student_list` may look like the following:

```
[('John', 'M', '21'), ('Kate', 'F', '19'), ('Eric', 'M',
'23')]
```

The function checks if ***every*** male student in the list is at least 20 years old. The function returns `True` if it is the case and `False` otherwise.

If the list doesn't contain any male student, the function returns `True`.

E.g.,

- `check_age([('John', 'M', '21'), ('Kate', 'F', '19'), ('Eric', 'M', '23')])` returns `True`.
- `check_age([('John', 'M', '21'), ('Kate', 'F', '19'), ('Jerry', 'M', '19')])` returns `False`.
- `check_age([('Kate', 'F', '19')])` returns `True`.

Run q3_test.py to test your code.

Q4: [***]

In this question, you need to process a file that contains many documents. Each line of the file contains two columns separated by tabs. The first column is a string, which is the ID of a document. The second column is the content of the corresponding document, shown as a sequence of words, separated by spaces.

You can open "documents.txt" to see an example file of this format.

In q4.py, define a function called `get_document_pair()`. The function takes in the name of a file as its input. The function tries to find out which pair of two documents in the given file shares the largest number of common words. For example, if "D1" and "D2" share 10 words in common, and "D1" and "D3" share 15 words in common, then the pair ("D1", "D3") has more common words than the pair ("D1", "D2").

When counting common words, if a word appears more than once in a document, it should be counted only once. For example, if the word "apple" appears 3 times in "D1" and 4 times in "D2", it should still be counted as 1 common word shared by "D1" and "D2".

The function should return a tuple of three elements: the IDs of the two documents and the number of common words they share.

When you call the function `get_document_pair()` with the argument 'documents.txt' you should get the following output: `('D2', 'D4', 3)`.

Run q4_test.py to test your code.