CS 120: Intro to Computer Programming II

In-Class Activity - 01 Python Review - Day 3

Activity 1 - Turn in this one

Write a file named `ica_01_3_act1.py`

Your code must:

- Read the name of a file, from the keyboard
- Open that file
- Read each line one at a time, either by writing a `for` loop, or calling the `readlines()` method on the file object.
- Print out the lines of the file, along with some metadata about each line.

See the testcases to see what metadata you must print out.

The trick about this ICA is that you **must not add extra blank lines in the output.** Discuss with your table why the extra blank lines tend to appear, and how you can avoid them.

Activity 2 - Turn in this one

Write a file named `ica_01_3_act2.py`

Suppose that you are given a dictionary. It's easy to search through the keys - that's what dictionaries are best at. But how do you search based on **values???**

Write a function `max_key_by_val(data)` that takes a dictionary as its only parameter. Build a loop that iterates through the keys. Look at the value for each one. Keep track of the **key** which has the maximum **value**. Return that key from the function.

EXAMPLE:

```
Key       Value
---       -----
foo       1234
bar       888888
baz       -1
fred      256
```

The key associated with the maximum value, from the example above, would be `"bar"`, because it's associated with the value 888888. The key associated with the second would be `"foo"`, which is associated with 1234.

(activity continues on next page)

Challenge Activity - Do not turn in this one

As a second step, take your code, and add a little bit to it, so that it can also find the key associated with the **second** largest value. Return both keys, like this:

```
return first,second
```

You can call a function like this, which returns two values, and save the results like this:

```
a,b = function_that_returns_two_things(some_param)
```

Challenge Activity - Do not turn in this one

Write a function, named `second_favorite_word()`. It must take exactly one parameter, and that is the name of a file to read. Inside the function, open that file.

Read each of the lines. Break each one into words. Then, for each word, use `lower()` to convert it into lowercase.

Use a dictionary to count how many times each word showed up in the file. When you reach the end of the file, hunt through the dictionary and find the second-most-common word. Return that word from the function.