CS 112, Lab 7 – Exercise **(due: Tues. 6/29, 11:59pm)**
Functions, Parameters

Files:

- **download** the tester file from blackboard for testing
- Grading: entirely test-case based.
  ```
  python3 testerL7E.py gmason76_230_L7.py
  ```
- You can also test individual functions. Name any you want tested at the end of the command line:
  ```
  python3 testerL7E.py gmason76_230_L7.py rank3 remove
  ```

As this is an **Exercise**, you can read any and all materials, ask us questions, talk with other students, and learn however you best learn in order to solve the task. Just create your own solution from those experiences, and turn in your work.

We can define functions to name a block of code, and then feed it inputs and get an output. Functions also allow us to give default values for parameters, which is useful but needs special attention. We will write functions that are more than just side-effect-free input-output factories.

Notes

- Don't call `input()` or `print()` anywhere.
- We need you to figure out the "signature" (top line) of some functions, so we don't explicitly show you the parameters list those times. Part of the goal is learning how to choose when to use default parameters, and what default values to use.
- You're also learning when to modify the original or not; test cases should target this fact.

Turning It In

Add a comment at the top of the file that indicates your name, userID, G#, lab section, a description of your collaboration partners, as well as any other details you feel like sharing. Once you are done, run the testing script once more to make sure you didn't break things while adding these comments. If all is well, go ahead and turn in just your one .py file you've been working on, named with our usual convention (see above), over on BlackBoard to the correct lab assignment. We have our own copy of the testing file that we'll use, so please don't turn that in (or any other extra files), as it will just slow us down.

Tasks

Implement each of these four functions.

Returning multiple values: tupling up the results.

- `def rank3(x,y,z, ascending=True):` Given three integers, return them in order in a `tuple` of length three. An optional fourth argument (`ascending`) states whether output should be sorted ascending (argument is `True`) or descending (argument is `False`).
    - Examples:
        - `rank3(5, 3, 4)` → `(3,4,5)`
        - `rank3(5, 3, 4, False)` → `(5,4,3)`
        - `rank3(6, 8, 6)` → `(6,6,8)`

Allowing defaults, modifying a list in-place:

- `def remove(val, xs, limit=None):` Remove multiple copies of `val` from `xs` (directly modify the list value that `xs` refers to). You may only remove up to the first `limit` occurrences of `val`. If `limit==3`, and `xs` had ten copies of `val` in it, then you'd only remove the first three and leave the last seven in place. When `limit==None`, there's truly no limit (and we remove all occurrences of val). Return `None`, because the work happened in-place. Negative or zero `limit`: no removals.
    - hint: if you need to traverse a list from front to back, but you don't always want to go to the next index location, while loops can be very useful – we don't have to step forward each iteration.
    - Note – some test cases are multiple lines (which we've avoided so far). You might need to notice the line number of a failed test, and go look at the code of that particular testing function, to see what's being attempted. This is a good habit to get into for projects.
    - Examples:
        - ```
          >>> xs = [1,2,1,3,1,4,1,5,1]
          >>> remove(1,xs)
          >>> xs
          [2, 3, 4, 5]
          ```
        - ```
          >>> xs = [1,2,1,3,1,4,1,5,1]
          >>> remove(1,xs,3)
          >>> xs
          [2, 3, 4, 1, 5, 1]
          ```

(see more on the next page)

Guess the Signature

Designing a function's signature is an important skill. We haven't gotten a lot of practice yet, as we absolutely need your functions to work with our testers, and that has so far meant that we tell you exactly how to interface with the testing scripts. But now, we will describe the signature, and you have to get it correct based upon the needs of the function (how it's supposed to behave).

- **`filter_chars:`** Write the definition for the **`filter_chars`** function. It accepts a message as the first argument (call it **`msg`**), and an optional argument called **`keeps`** as the second argument. Dig through **`msg`**, and any character in **`msg`** that is also present **`in keeps`** gets kept; all others are ignored/skipped. Return this possibly-shorter string.

    - Strings can't be modified, so we can't modify the message. But we can create a new string that only contains the parts we wanted to keep.
    - When **`keeps`** isn't provided, all upper and lowercase letters should be kept.
    - Hint: the **`in`** keyword is your friend.

    - Examples:
        - `filter_chars("hi there! :)")` → `'hithere'`
        - `filter_chars("hi there! :)", "ether")` → `'hthere'`
        - `filter_chars("hi there! :)", "ie:")` → `'iee:'`
        - `filter_chars("hi there! :)","1234567890")` → `''`

Using a default value that is complex: Watch out!

- **`relocate_evens:`** The first argument, **`data`**, contains integers. We will be removing any evens we find in **`data`**, and appending them to the second argument, **`new_home`**. A reference to the list which received any evens must be returned. When no second argument is given, a list of all the relocated evens should be created and returned.

    - Careful! What default value do you want to use for **`new_home`**? What happens when we call the function multiple times in the same coding session? *(try it out)*

    - Go ahead and use methods like **`.insert()`**, **`.pop()`**, **`.append()`**

    - Examples:
        - ```
          >>> xs = [1,2,3,4,5]
          >>> relocate_evens(xs)
          [2,4]
          >>> xs
          [1,3,5]
          ```

        - ```
          >>> vals = [6,7,8,9,10]
          >>> sanctuary = [2,4]
          >>> relocate_evens(vals,sanctuary)
          [2,4,6,8,10]
          >>> sanctuary     # same as what relocate_evens returned
          [2,4,6,8,10]
          >>> vals          # all evens have been relocated out of here.
          [7,9]
          ```