CSE 160 Section 4 - Lists

Welcome to section!!! 😃

Lists

Lists are a type of variable that can contain multiple values inside of them. We can perform many operations on a list including:

- Accessing elements and possibly modifying them. **`list[index] = new_value`**
- Slicing the list to get a "sublist". **`sublist = list[0:2]`**
- Extending the list with functions like **`list.append(x)`**, **`list.extend(L)`**, and **`list.insert(i, x)`**.
- Removing items from the list with **`list.remove(x)`** and **`list.pop(i)`**.
- Rearrange the list with **`list.sort()`** and **`list.reverse()`**.

We can take a look at a short example for how we can use these list operations:

```
# create two lists
list1 = [1]
list2 = [2, 160]

list2[1] = 0          # list2 is now [2, 0]
list2.append(3)       # list2 is now [2, 0, 3]
list1.extend(list2)   # list1 is now [1, 2, 0, 3]
list1.sort()          # list1 is now [0, 1, 2, 3]
list1.pop()           # list1 is now [0, 1, 2]
```

<u>Exercise 1</u>:

After the following lines of code are printed, what values are printed?

```
list_1 = [1, 2, 3, 4, 5]
list_2 = list_1
list_3 = list_1[0:5]
# ^ Note: list_1[:] is considered more general,
# better overall and equivalent to list_1[0:5]

list_2[0] = 98
list_1[4] = 99
print("List 1:", list_1)
print("List 2:", list_2)
print("List 3:", list_3)
```

<u>Exercise 2</u>:

Given the following code:

```
list_1 = [1, 2, 3, 4]

list_2 = [10, 12]

list_3 = [21, 22, 23, 24]

list_4 = [13, 14, 15]

list_5 = [16, 17, 18, 19, 20, 25, 26]

list_6 = [9, 8, 7, 6, 5]

list_7 = [11]
```

Use only list accesses `[ ]`, `extend()` ,`insert()` , and `reverse()` – and possibly, a `for` loop. DO NOT add, subtract, or perform other mathematical operations on the numbers in the lists.

-- to modify `list_1` so that it contains numbers 1 through 26 in increasing order:

```
print("List 1:", list_1)
```

The command above should print a list that contains numbers 1 through 26 in increasing order.

<u>Exercise 3</u>:
Using the same initial lists as Question 2, redo the problem, but this time you are allowed to use the `sort()` function.

<u>Exercise 4</u>:
Create a function "`dot_product`" which takes in two lists of integers and returns the dot product of the two lists. You can assume the lists are of equal length, and contain only integer values. To calculate the dot product of two lists, find the product of each value in one list, with the value at the same index in the other list. The dot product of the lists is the sum of these products. For example, the product of the lists `[1,2]` and `[3,4]` is: `1 * 3 + 2 * 4 = 11`.

```
def dot_product(list1, list2):
```

<u>Exercise 5</u>:
Write a function called count_words that takes in a string file_name and returns the number of words in the file.

```
def count_words(file_name):
```

<u>Challenge Problem</u>:

For each of the following questions, implement a function that returns the answer to the question. For each question, follow this procedure:

1. Identify a good name for the function.
2. Identify the return value.
3. Identify any necessary parameters.
4. Write the function definition.
5. Write the function's docstring.
6. Describe, on paper, in words, or in your head, the algorithm you'll use to solve the problem.
7. Implement the function.

Note: Practice effective programming by making sure you understand your approach in step 6 before implementing the code in step 7. It is also a good idea to think of test cases that validate your function input/outputs.

You and your friends have set up "best friends" lists. Each of your lists contains the name of the list owner followed by the names of his/her top friends. Your name in your friends' lists is "me".

```
my_friends = ["me", "Emily", "John", "Ed", "Louise", "Tom"]
emily_friends = ["Emily", "me", "Rob", "Sue", "Alice", "Eric"]
john_friends = ["John", "Ed", "Rob", "Sue", "Eric", "Meg", "Emily"]
ed_friends = ["Ed", "me", "Tom", "John", "Emily","Sue", "Liam"]
louise_friends = ["Louise", "me", "Alice", "Sue", "Emily", "Meg"]
tom_friends = ["Tom", "John", "Alice", "Ed", "Louise", "Emily", "me"]
```

Friends (a list of lists) is constructed as follows. Your list is at position `friends[0]` .

```
friends = [my_friends, emily_friends, john_friends, ed_friends,
louise_friends, tom_friends]
```

1. Write a function to return the index of the first occurrence of name in a friend list, given the name and a friend list. If the name is not in the list, the function will return `-1` .

2. Write a function that given a name and `friends`(the list of friend lists), will return the index of that person's list in the friends list. If that name is not found, your function will return `-1`.

3. Write a function that, when given `friends` will calculate how many of your best friends views you as one of their best friends (if you are their “mutual friend”) and return the value.

4. Modify your function above to calculate the "mutual friend" value for any individual friend list. Hint: Using a function you made in 1-3 may simplify your solution.

5. Write a function to determine, given `friends`, if any of the lists consist entirely of people who "agreed" on their mutual "best friend status. That is, if you view them as a best friend, they also view you as a best friend.