CMPSC 102
Discrete Structures
Fall 2018

Lab 3 Assignment:
Functions Playing Roles of Conversations in Python
Submit deliverables through your assignment GitHib repository bearing your (team) name. Place source code in `src/` and `writing/` directories.

Objectives

To enhance the understanding of Python by completing a program designed with functions which use lists containing strings and values. This lab is designed to give practice in using function calls, using and parsing lists (or sets) of string material using statements, performing conditional checks using `if` statements and other types of code. To practice using functions to determine which strings of information to send in response to an input from another function.

GitHub Starter Link for Groups

STOP! STOP!
Not everyone will be clicking this link!
Only the team leader will be clicking the link to create the repository!!
https://classroom.github.com/g/ro1NCIXt

To use this link, please follow the steps below.

Since this is your first team-based assignment we will be using a group assignment functionality of GitHub Classroom. For group assignments **only one person will be creating the team while the other team members will join that team.**

The selected person of the team should go into the link to the lab in the assignment sheet. Copy this link and paste it into your web browser. Now, you should accept the laboratory assignment and create a new team with a unique and descriptive team name (under "Or Create a new team").

Now the other members of the team can click on the assignment link and select their team from the list under "Join an Existing Team". When other team members join their group in GitHub Classroom, a team is created in our GitHub organization. Teams have amazing functionality, including threaded comments and emoji support. Every team member will be able to push and pull to their teams repository. Your teams project manager should be the one to resolve any conflicts or merge pull requests.

Please work in groups: Unless you provide the instructor with documentation of the extenuating circumstances that you are facing, not working in a team and not accepting the assignment means that you automatically receive a failing grade for it.

To push your changes, you can use the following commands to add a single file, you must be in the directory where the file is located (or add the path to the file in the command):

- `git commit <nameOfFile> -m ‘‘Your notes about commit here’’`
- `git push`

Alternatively, you can use the following commands to add multiple files from your repository:

- `git add -A`
- `git commit -m ‘‘Your notes about commit here’’`
- `git push`

Reading Assignment

Please read Chapters 5 (Stavely) in the course book, consult the week's slides and your class notes. You can also find useful information in the Python community by performing online research. Please take some time to gain experience with using Markdown to complete your writing assessments. See *Mastering Markdown* `https://guides.github.com/features/mastering-markdown/` for more details about Markdown.

Additional Materials

Please locate your `src/` directory for this lab where you will find an empty file into which you will add your code for the completion of this lab.

- An empty Python3 source code; `src/myTalkingHeads.py`
- An empty output file to be written in Markdown; `writing/output.md`
- An empty reflection document to be completed in Markdown; `writing/reflection.md`

About this Lab: Group Work

In this lab, **you will be working in groups no larger than four people** to design a Python program to emulate two (or more) people having a conversation with each other. Your program will actually be made up of functions which exchanging text contained in Python `lists`. These functions will be controlled in some way by a driver function to determine who is speaking and who is listening at a time. To add realism to your two speakers, your code is to have some type of decision making ability to be able match their exchanges so that there is some logical connection between the statements that both speakers say. You are at liberty to create whatever type conversation you desire as long as there is some noticeable connection between what is said between both speakers.

The pieces of conversation stemming from each of your speakers are strings which are to be kept in a "vocabulary" list. You will use this list to pair common words between both speakers to make it look as if a real conversation (in a way) is occurring. Please use randomness when preparing the statements of each speaker. It might be important to remove the *stopwords* from

your conversations (i.e., words which have no real bearing on the actual meaning of the statements such as "and", "have", "is" and etc.) See your included source code for a function to edit that is able to perform this work.

Example of an Conversational Exchange

Below is an conversational exchange between two speakers (Alice and Bob). Each statement has been stored in a `list` or a `set` and has been selected for broadcast due to a common theme (the conversation about the *dog*). All statements have the same word, *dog* in them to connect the speakers. This word can also be checked by your conditional statements in your code to pair a initial statement to its response.

- **SpeakerA** : "I took my *dog* for a walk."
- **SpeakerB** : "I like dogs, what is the name of your *dog*?"
- **SpeakerA** : "The name of my *dog* is Rex."

Figure 1: A conversation between different people; The first speaker begins the conversation and then the second speaker reacts to the discussion and contributes an idea back to the first speaker. One person listens while the other speaks for the duration of the conversation.

Your Output of a Conversational Exchange

Sample output from a sample solution to the *TalkingHeads* program. Your format may look different as long as there is some initial statement and a logical response.

```
# Your conversation strings are elements stored in a list
# (here called "vocabulary lists") and will be used to
# in the flow of exchanges between two speakers.
# Be creative in the design your content as well as to add many strings
```

```
# to use for a conversation between your speakers!

aliceVocab_list = ["I like cats", "I like dogs",
"I like rabbits", "I gave carrots to horses", "I live on a farm"]

bobVocab_list = ["I have two cats", "I have three dogs",
"I know several rabbits","I love carrots", "I love horses",
"I also live on a farm"]
```

```
  This is Alice. I say to Bob : I like dogs
  This is Bob. I reply to Alice : I have three dogs

  This is Alice. I say to Bob : I like cats
  This is Bob. I reply to Alice : I have two cats

  This is Alice. I say to Bob : I gave carrots to horses
  This is Bob. I reply to Alice : I love carrots

  This is Alice. I say to Bob : I live on a farm
  This is Bob. I reply to Alice : I also live on a farm
```

Functions

There will be at least three functions in your code which have been added to the source code to edit. One function (likely `main()` will be the moderator which will decide who is speaking (`speakerAlice()`) and who is listening (`speakerBob()`). The other functions control the speakers themselves. Each function will handle its speaker's own diverse vocabulary which still shares enough overlap with the other speaker's vocabulary to be able to find some response statement for any statement from the previous speaker.

Extra Challenge

It is not necessary for a grade but you might want to try adding this functionality anway: Add a user input line of code and a `for`-loop make the conversation between speakers continue for an arbitrary number of steps.

Required Deliverables

Submit deliverables through your assignment GitHib repository bearing your name, as well as all names in your group for your group work. Place source code in `src/` and the Markdown files `writing/` directories.

1. **Use Group Work with a Team Leader**: You are to work in groups to complete this

open-ended lab. Give your group a name. Each member is to work collaboratively with group members. The group will submit all materials (code and documents mentioned below) to the team repository which will be graded by the instructor. You are to use Markdown to format your written documents. Please be sure to add the names of each member of your group in all your submitted work (i.e., code and markdown documents).

2. **src/myTalkingHeads.py**: Your completed and working python code that you created by editing the file *myTalkingHeads.py* in the `src/` directory. Please be sure to add your coding comments for your developed functions to explain the functionality! Your team should work on this document together; be sure to include the names of all team members. Also, please try the implemented the Extra Challenge as discussed above.

3. **writing/output.md**: A Markdown file (called `writing/output.md`) containing the text of the exact output of your program. Your team should work on this document together and include the names of all members in the document.

4. **writing/reflection.md** A markdown document to contain your reflections from this project. For this document, please prepare a one-half page discussion of the challenges and motivations that you and your group had to work through during the course of this project. Please elaborate on how these challenges were addressed. Your team should work on this document together and include the names of all members in the document.