Homework 4: Pick Your Letters

HW deadline as per Canvas.

This homework deals with the following topics:

- lists
- tuples

In this HW, we will be implementing the game *Pick Your Letters*, which is a game that involves re-arranging a group of cards in order to make up a word.

About the Game

This game needs two players and we will just play the user versus the computer. The user's moves are decided by the user playing the game, by asking for input, and the computer's moves are decided by the program.

There is NO right answer for the section that asks you to program a strategy for the computer. All we want you to do is come up with a reasonable enough strategy that ensures a human user does not always beat the computer. So, unlike your previous assignments, this one has a creative component to it.

You must use Python lists and tuples for this assignment.

At the very beginning, the program will prompt the user to enter a number as the Length *L* to be the length of the word they are going to make up. The game starts with a main pile of *n* cards, each labeled with a letter, and $n = 26 * L$. So, there are *L* of each letter. The objective it to be the first player to arrange *L* cards in your hand in an order that forms a word.

The cards in the main pile are shuffled and both the user and the computer are each initially dealt *L* cards from the main pile. As a player receives each card, they must place the card at the far right of their hand in the order they received it. E.g., if a user receives 'A', 'R', 'K', 'I', 'P' consecutively, their hand is in the same order and looks like: 'A', 'R', 'K', 'I', 'P'.

After the hand is dealt to the user and the computer, there will be $((26 * L) - (2 * L))$ cards left in the main pile. The top card of the main pile (the card to the far left of cards list) is turned over to begin the discarded card pile.

On each player's turn, the player chooses to either pick up the top card from the discard pile, or to pick up the top card from the main pile. The top card from the discard pile is known. In other words, the discard pile is 'face up' and everyone knows what the letter on it is. The main pile is 'face down' and the player does not know what the card is.

Once a player chooses a card, either from the discard pile or from the main pile, the player decides where in the hand to put the card. The hand is always of the size *L*, so placing a card means that an existing card in the hand is removed and replaced

with the new card.

If the player takes a card from the main pile (the one that is 'face down'), the player can reject it and place it in the discard pile. This means that nothing in that player's hand changes during that turn. If the player takes a card from the discard pile (the one that is 'face up'), the player MUST place it into their hand.

The first player whose cards in hand form a word wins. How does the program know the cards form a word? We'll discuss that below, soon!

If, at any point, all of the cards have been removed from the main pile, then all of the cards in the discard pile are shuffled and moved to the main pile. Then the top card is turned over to start the new discard pile.

The Actual Program

Below you will find explanations of the functions that need to be written. We are expecting to see these functions with these names and method signatures exactly. Do not change the names of these functions, as we will be running automated tests against each individual function.

You will also have to write some functions by yourself, in addition to the ones listed below. Feel free to name your helper functions whatever you want, as long as they happen to reflect what the function does.

Note that we will make heavy use of lists in this assignment. Since hand cards look like a horizontally oriented structure, we need to have some convention. Our convention is that the hand in the picture below is going to be represented as ['K', 'K', 'A', 'A', 'A'].

If $L = 6$, and the player is dealt the last card 'S', the 'S' is inserted to the right and the list looks like ['K', 'K', 'A', 'A', 'A', 'S']. If you want to replace 'S' with another card, reference index 5, which means the card with index 5 in the list.

The main pile and discard pile are also going to be represented as lists. However, cards in piles looks more like a vertically oriented structure. Our convention is that

the card on the top of a pile is always to the left of the list.

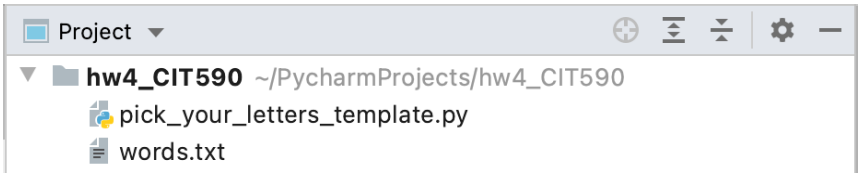
Note that in both the main pile and the discard pile, you should only have access to the top. So, to add/remove card, consider the top of the pile to be the beginning of the list. You will have to use the pop function, the append function, and the insert function, in some manner.

How does the program know hand cards form a word?

We've already written a function for you:

def read_from_file(file_name):

This function reads a file with name 'words.txt', which contains all the words acceptable in the game. All the words have a length of 3 - 10, inclusive, and only contain letters (ignore case). This function returns a list containing all these words. Note that 'words.txt' and 'pick_your_letters_template.py' should be under the same directory as described below.



After calling this function (shown below), all words are stored in the list named *all_words*.

```
all_words = read_from_file("words.txt")
```

You can now use *all_words* directly. The first player whose hand cards forms a word that is included in *all_words*, wins.

Functions to implement

Be sure to add docstrings to all your functions and comments to your code.

ask_for_length():

- Ask the user for the number of hand cards so that the length of words players is going to guess is determined
- Prompt again if the user input is not a valid integer, or if the number is not between 3 to 10, inclusive
- Returns the number of hand cards *L*

filter_word_list(all_words, length):

- Given a list of words, and a number, returns a list of words with the specific

length
- Parameter *all_words* is the list of all words
- Parameter *length* is the given length

set_up(length):
- You'll run this function once after the length of words is determined and the list of words is filtered
- Creates a main pile of 26 * *length* cards, represented as a list of lowercase letters, with *length* of each letter
    - For example, if you called this function with an argument of 2, it will create a main pile of cards represented as a list of lowercase letters, with each letter included twice:
      ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z', 'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z']
- Creates a discard pile of 0 cards, represented as an empty list
- This function returns both lists as a tuple, with the main pile as the first item and the discard pile as the second item
    - The method of returning 2 things from a function is to make a tuple out of the return values. For an example of this, refer to the lecture slides/code where we return both the maximum and minimum in a list.
- Parameter *length* is the given length

shuffle_cards(pile):
- Parameter *pile* is the given list of words
- This function shuffles the given *pile* and doesn't return anything
- You are allowed to import the *random* module and just use *random.shuffle*

deal_initial_cards(main_pile, discard_pile, length):
- Start the game by dealing two sets of *length* cards each, from the given *main_pile*
- Make sure that you follow the normal conventions of dealing. So, you have to deal one card to the computer, one to the user, one to the computer, one to the user, and so on.
- The computer is always the first person that gets dealt to and always plays first
- Remove the card on top of the main pile and put it on the discard pile
- This function returns a tuple containing two lists, the first one representing the human's hand and the second one representing the computer's hand

move_to_discard_pile(discard_pile, card):
- Move the given *card* to the top of the discard pile
- Parameter *discard_pile* is the discard pile
- Parameter *card* is the given letter to be discarded
- This function doesn't return anything

get_first_from_pile_and_remove(pile):
- Return and remove the first item of the given list

- Parameter *pile* is the list from which to remove the first element

check_bricks(main_pile, discard_pile):

- Check whether the main_pile is empty.
- If so, shuffles the discard_pile and moves all the cards to the main_pile. Then turn over the top card of the main_pile to be the start of the new discard_pile.
- Otherwise, do nothing.

computer_play(computer_hand_cards, computer_target_list, main_pile, discard_pile):

- This function is where you can write the computer's strategy. It is also the function where we are giving you very little guidance in terms of actual code.
- You are supposed to be somewhat creative here, but we do want your code to be deterministic. That is, given particular hand cards and a particular letter (either from the discarded pile or as the top of the main pile), you either always take the card or always reject it.
- Here are some potential decisions the computer had to make:
    - Given the computer's current hand, do you want to take the top card from the discard pile or do you want to take a card from the top of the main pile and see if that card is useful to you
    - How you evaluate the usefulness of a card, and the position it should go into
    - There might be some simple rules you can give to the computer. For example, you can compute how many similarities there are between each word in the computer's target list of words and the letters on hand. And you can test if replacing a specific card with another card can increase the similarities.
- You are allowed to do pretty much anything in this function except make a random decision or make a decision that is obviously incorrect. For instance, selecting a fixed target word is not very smart. We want your computer to determine the target word, or target words, dynamically.
- Also, the computer CANNOT CHEAT. What does that mean? The computer cannot peek at the top of the main pile and then make a decision to go to the discard pile. Its decisions should be something that a human could be able to make as well.
- For a given letter, the computer decides whether to take it or not
- Parameter *computer_hand_cards* is the computer's hand cards
- Parameter *main_pile* is the main pile
- Parameter *discard_pile* is the discard pile
- Parameter *computer_target_list* is a list of words. What this list looks like is up to you. It may contain words that are most similar to hand cards or something else, but it should be reasonable.
- This function doesn't return anything

ask_for_the_letter_to_be_replaced(length):

- Ask for the index of the letter that the user wants to replace
- Prompt again if the input index is out of range or invalid
- Parameter *length* is the number of cards in the human's hand
- This function returns the index of the letter to be replaced

ask_yes_or_no(msg):

- Displays *msg* and get user's response
- Prompt again if the input is invalid
- Parameter *msg* is the message to display
- This function returns True if the user answers 'y' or 'yes', and returns False if the user answers 'n' or 'no'

check_game_over(human_hand_cards, compute_hand_cards, words_with_specific_length):

- Check if the game ends
- If there is a tie, the game ends as well
- Parameter *human_hand_cards* is the human's current hand (list)
- Parameter *computer_hand_cards* is the computer's current hand (list)
- Parameter *words_with_specific_length* is a list containing all the words with the specific length
- Returns True if the human or the computer wins the game, otherwise False

User Interface

You're free to create your own user interface for the game, as long as it makes sense for the user playing. The computer goes first. Your program can print the computer's initial hand (in list form) at the beginning of the game. And for the rest of the game, the human player should also see what's in the computer's hand. After the computer's turn, your program should print the results of the turn. For example, your program might print this if the computer chooses the top of the discard pile or chooses the top of the main pile.

```
Welcome to the game!
Enter a number between 3 - 10 to be length of the word you are going to guess:
4
Computers' turn
Computer discarded 's' from MAIN PILE
computer's current hand is
 ['h', 'o', 'k', 'z']
-----------------------------------------------
```

When it's the human user's turn, your program should print the user's hand (in list form), the top card of the discard pile, and, if they choose to pick from the main pile, the top card of the main pile. For example, your program might print this if the human user chooses the top card of the main pile and replaces a card in their hand.

```
------------------------------------------------
Your turn
Your word list is:
['r', 'b', 'c', 'v']
Pick 'g' from DISCARD PILE or reveal the letter from MAIN PILE
Reply 'D' or 'M' to respond:
M
The letter from MAIN PILE is 's'
Do you want to accept this letter? Type 'y/yes' to accept, 'n/no' to discard.
yes
Input the index of the letter to be replaced, e.g. '1':
1
You replaced 'b' with 's'
Your word list is: ['r', 's', 'c', 'v']
------------------------------------------------
```

Note, do not use Python's time.sleep() function to "slow down" (or suspend execution of) your program. This will make playing the game and grading difficult.

Evaluation

The primary goal of this assignment is to get you to feel familiar with lists and tuples and to have some level of fun while creating a game.

While we want you to spend time on coming up with some kind of strategy for the computer, that is NOT the primary part of the assignment. Any simple, but reasonable strategy will have you doing some fun things with lists. If the human player always does absolutely nothing at all, that is, if they reject the top discarded card and they reject the top card from the main pile and move it onto the discard, then we want the computer to win. Your computer player should be smart enough to beat the 'stupid, lazy' user.

You will be evaluated on 5 things:

1. Does your game work - 5 points
   Again, please do not worry about us trying to crash your program. Just ensure that reasonable inputs will allow a user to play the game.
2. Function design – 5 points
   In addition to your game working as it should, we will confirm that the required functions above have been implemented correctly, and do exactly what they are supposed to do. We will test the functions individually by running "unit tests" against them with different inputs and by inspecting the results.

   For example, the *move_to_discard_pile* function should add the given card to the top (or beginning) of the given discard pile. We will call your function with a test card and test discard pile as the arguments, then confirm that the top card of the test discard pile is what it should be.
3. Strategy – 5 points
   Is your computer's strategy something that makes sense? Please comment

your code for that part of the homework very thoroughly, and explain your strategy for the computer.

4. Usability – 3 points
   This is going to be a subjective evaluation. You are making a game. A user who knows the rules of the game should be able to play it without too much trouble. In particular, they should not have to read a single line of your code in order to operate the game.
5. Style – 2 points
   The usual stuff here. Style includes things like appropriate variable names, function names, clear comments, and general readability of your code. Every function must have a docstring defined. All non-trivial lines of code must be commented. Style will always be part of your evaluation.

Data Structures

In this HW you are only allowed to use lists and tuples. Do not use dictionaries, classes, or any built-in python libraries (other than the *random* library for shuffling, *string* library for setting up the alphabet for the main pile, and *copy* library for copying). Remember that the main pile, the discard pile, and the two hand cards (user and computer) are just lists.

Submission

Submit a single file called **pick_your_letters.py**

Remember to put the following lines of code at the end of your file, as the entry point of your program.

```
if __name__ == '__main__':
    main()
```