CS-UY 1114: Lab 9

Dictionaries

You must get checked out by your lab CA **prior to leaving early.** If you leave without being checked out, you will receive 0 credits for the lab.

Restrictions

The Python structures that you use in this lab should be restricted to those you have learned in lecture so far. Please check with your teaching assistants in case you are unsure whether something is or is not allowed!

Create a new python file for each of the following problems.

Your files should be named _lab[num]*q[num].py* **similar to homework naming conventions.**

Problem 1: *Dictionary Warmup*

Given the following code, predict the output of the lines below:

```
def f1(my_dict):
    temp = 0
    for val in my_dict.values():
        temp = temp + val
    return temp
```

```
def f2(my_dict):
    temp = ""
    for key in my_dict:
        if temp < key :
            temp = key
    return temp
```

```
def f3 (my_dict, k, v):
    if k in my_dict:
        my_dict[k] = v
```

```
def main ():
    a_dict = {"Jennifer" : 13, "Asher" : 27, "Jeff" : 19, "Chitra" : 23}
    print(f1(a_dict)) # line 1
    print(f2(a_dict)) # line 2
```

```
    f3(a_dict, "Asher", 30)
    print(a_dict) # line 3
```

(a) What is the output produced by the line `print(f1(a_dict))` (line 1)?

(b) What is the output produced by the line `print(f2(a_dict))` (line 2)?

(c) What is the output produced by the line `print(a_dict)` (line 3)?

Problem 2: *Dictionary Fun*

Given the dictionary:

```
my_dict = {"a": 15 , "c": 35 , "b": 20}
```

Write Python code to do the following:

(a) Print all the keys

(b) Print all the values

(c) Print all the key, value pairs

(d) Print all the key, value pairs in order by key

Hint:

Use the list method `sort()`. If given a list of tuples, the method will sort on the first items in the tuple.

Problem 3: *Frequencies!*

Write a function with the signature count_digits that takes a list of integers, lst, and returns a dictionary containing the total number of times that each digit between 0 and 9 appears in lst.

```
def count_digits(lst):
    """
    parameters:
        lst: a list of integers
    return:
        dictionary with frequency of digits
    """
```

```
count_digits([1,2,3])

{1: 1, 2: 1, 3: 1}
```

```
count_digits([2,0,1,9,0,4,1,9])
```

```
{2: 1, 0: 2, 1: 2, 9: 2, 4: 1}
```

Problem 4: *Grocery List*

Preparing for grocery shopping can be done in two steps: taking inventory of what you currently have, then using that information to determine how many of each item you need to buy. Your program should help complete this task in two functions, each function reflecting one step of this two-step process.

Part 1: *Creating an Inventory of Our Fridge Items*

In a function called `def create_grocery_inventory()`, continuously prompt the user for input of an item name and the current quantity they own of this item. These values should be separated by a comma. You may assume the user will provide a valid input in the format of `item_name,item_amount`.

Then, this function should store this information inside of a dictionary where the keys are the `item_name` and the values are the `item_amount`. Finally, return this dictionary.

The defintion of the function is as follows:

```
# Prompts for user input to populate a grocery list
#
# :return: a dictionary of item keys with their quantities as values
def create_grocery_inventory():
    # Code here
```

Note: The user may input the same `item_name` more than once. There should only be one dictionary entry per item!

Part 2: *Creating a Grocery List*

In a function called `def create_grocery_list(fridge_inventory)`, given a dictionary `fridge_inventory`, which reflects the current quantities of items in a fridge, continuously prompt the user for item of an item name and the quantity desired. These values should be separated by a comma. You may assume the user will provide a valid input in the format of `item_name,item_desired_amount`.

Then, based off the information in `fridge_inventory` and user input, you must create an appropriate entry in a new dictionary where keys are the `item_name` and values are the `amount_to_buy` after your calculations. For example, if we already own 5 apples and want to have at least 3 apples, we do not need to buy any more apples. Finally, return this dictionary.

The definition of the function is as follows:

```
# Prompts for user input to populate a grocery shopping list given the
# current inventory of items
#
# :param: dictionary fridge_inventory
# :return: dictionary of item keys with their quantities to buy as values
```

```
def create_grocery_shopping_list(fridge_inventory):
    # Code here
```

Constraint: You may assume the user will only input items that already exist in the fridge. The user will additionally only input an item once.

You may use the following main definition to test your code:

```
def main():
    fridge_inventory = create_grocery_inventory()
    print()
    grocery_list = create_grocery_shopping_list(fridge_inventory)
    print()

    print("Your shopping list, based off of what you have in your fridge,
should be:")
    print(grocery_list)
```

The following is an example of a possible output:

```
Please enter the item and quantity you own separated by a comma or DONE
when complete: apple,3
Please enter the item and quantity you own separated by a comma or DONE
when complete: banana,2
Please enter the item and quantity you own separated by a comma or DONE
when complete: coconut,1
Please enter the item and quantity you own separated by a comma or DONE
when complete: banana,3
Please enter the item and quantity you own separated by a comma or DONE
when complete: DONE

Please enter the item and quantity you desire separated by a comma or DONE
when complete: apple,4
Please enter the item and quantity you desire separated by a comma or DONE
when complete: banana,6
Please enter the item and quantity you desire separated by a comma or DONE
when complete: coconut,1
Please enter the item and quantity you desire separated by a comma or DONE
when complete: DONE

Your shopping list, based off of what you have in your fridge, should be:
{'apple': 1, 'banana': 1, 'coconut': 0}
```

Problem 5: *Contact List*

In this problem, you will write a program which uses a dictionary to store phone numbers of contacts. You will name your dictionary contacts. It will store the names as keys and phone numbers as values. Your program should have the following functions:

Part A: *Adding Conacts*

Write a function `add_entry(contacts, name, number)` that adds an entry to the dictionary contacts.

```
def add_entry(contacts ,name ,number):
    """
    parameters:
        contacts: a dictionary of contacts
        name: a string representing a name
        number: a string representing a phone number
    return:
        None
    """
```

The function takes a dictionary, contacts, a name as a string and a number as a string. It adds an entry to contacts with the name as the key and the number as the value. The item should only be added to the dictionary if the following is true:

- An entry in contacts with the key name does not exist already.
- Number is a valid phone number. A valid phone number is a string with 10 digits. For example, `2014567890` is a valid phone number. But `201a45b789` or `20145678` are both invalid.
- Your function should display an error message if the entry cannot be added. Test your function with valid and invalid numbers and duplicate names to ensure that it works correctly.

Part B: *Searching for a Contact*

Write a function `lookup(contacts, name)`. This function should return the phone number associated with name in contacts. If the input name is not found in contacts, return an error message that indicates so. Test your function with valid and invalid inputs to ensure that it works correctly.

```
def lookup(contacts, name):
    """
    parameters:
        contacts: a dictionary of contacts
        name: a string representing a name
    return:
        a string representing a phone number
    """
```

Part C: *Deleting a Contact*

Write a function `delete_entry(contacts, name)` that deletes the entry with the key name from contacts. If the input name is not found in contacts, print an error message that indicates so. Test your function with valid and invalid inputs to ensure that it works correctly.

```
def delete_entry(contacts, name):
    """
```

```
    parameters:
        contacts: a dictionary of contacts
        name: a string representing a name
    return:
        None
    """
```

Part D: *Printing all Contacts*

Write a function `print_all(contacts)` that prints all the entries in contacts. Your function should print both the name and the phone number.

```
def print_all(contacts):
    """
    parameters:
        contacts: a dictionary of contacts
    return:
        None
    """
```

For example, the output of contacts could be:

```
Apoorva    1234567890
Kim        2345678901
Selina    3456789012
```

Use the tab character "\t" for spacing between the name and phone number. Test your function to ensure that it works properly.

Part E: *Main Function*

Write a `main()` function to test your program. Your main function should create an empty dictionary, contacts. Then it should prompt the user to input an option. The options are:

- Q: Quit the program
- A: Add an entry to contacts
- L: Look up a phone number
- P: Print all entries in contacts
- D: Delete an entry from contacts

Your program should continue to prompt the user for an option until the user enters `Q`. If the user enters `A`, your program should prompt the user for a name and a phone number and add the contact. If the user enters `L`, your program should prompt the user for a name and lookup the contact. If the user enters `P`, your program should print all the entries in contacts. If the user enters `D`, your program should prompt the user for a name and delete the contact. If the user enters an invalid option you should print an error message.

Here is an example of the output of your program:

```
Please enter an option: A
Please enter a name: Apoorva
Please enter a phone number: 1234567890
Please enter an option: A
Please enter a name: Kim
Please enter a phone number: 2345678901
Please enter an option: A
Please enter a name: Selina
Please enter a phone number: 3456789012
Please enter an option: L
Please enter a name: Apoorva
1234567890
Please enter an option: L
Please enter a name: Billy
Billy is not in the contact list
Please enter an option: P
Apoorva:    1234567890
Kim:        2345678901
Selina:     3456789012
Please enter an option: D
Please enter a name: Apoorva
Please enter an option: P
Kim:        2345678901
Selina:     3456789012
Please enter an option: Q
```