Università della Svizzera italiana

Faculty of Informatics

Programming Fundamentals I
Fall 2014
Topic 7: Associative data

Prof. Mehdi Jazayeri

Assisted by: Anton Fedosov, Yuriy Tymchuk, Rui Xin

Week 6: 3 - 9 November, Due: 10 November

Instructions

Please read these instructions carefully.

Each week in the first ten weeks of the semester, you are supposed to master a topic and demonstrate your mastery to the teaching staff. To gain mastery, you should attend the lecture, study the topic, and do many exercises on the week's topic. The more exercises you do, the more confidence you will gain in your ability and your understanding. We will post some questions on iCorsi that you can use to test yourself. Also, the textbook has exercises that you can use to test yourself. Make sure you can answer ALL the exercises in the book.

To show your mastery of the topic, you will typically be given a program to study and then modify it to enhance its features. We will also give you a list of topics you could explore on your own if you want to extend your knowledge.

Finally, in the Monday atelier session, we will ask you to write a program and explain how it works.

For any questions, feel free to post on the iCorsi questions forum for the course or ask during the atelier session.

Week 7

Topic: This week you will work with associative data structures such as dictionaries.

Introductory questions:

1. Write a function to print the keys of an input dictionary.

2. Write a function to print the values in a dictionary.
3. Write a function that accepts a dictionary of people as name, age, and returns the average age of the people.
4. Write a function that accepts a dictionary of months as month name, number of days and prints a nicely formatted calendar of all the days of the months of the year.
5. Write a function that given a value and a dictionary, returns the key associated with that value in the dictionary (this is called inverse lookup).
6. Write a function that given a value and a dictionary, returns the number of times that value occurs in the dictionary.
7. Write a function that prints the keys of a dictionary in increasing order of the associated values in the dictionary (Assume key, value for the dictionary where value is integer.)
8. An inventory dictionary countains as key the name of an item as a string (e.g. laptop model), and as value a list in the format *[Number, Price]*. Write a function to update *number* of the inventory items, based on the user input that requests one or more items.
9. Populate dictionary from a file. Each line of the file contains a (key, value) pair separated by space.
10. Use a dictionary to represent a one-week agenda. The dictionary should support events being scheduled for days of the week, starting time, and duration.
11. Given the dictionary in the previous question, write a function to detect conflicting events.
12. Create a dictionary to represent a sample of the iTunes music database.
13. Create a dictionary to represent a people database.

Program to modify: database with dictionary

In this section we provide sample code of the *update()* function in *Assignment 5*. (To reduce the complexity we've removed the 'year of birth' term.) Please modify the code to use a dictionary (i.e., 'Alice': 'Female', 'Bob': 'Male', 'Carol': 'Female') instead of a list for updating the database.

```
# databse.py
# This is a sample solution of the 'database' section.

data = [['Alice', 'Female'],
        ['Bob', 'Male'],
        ['Carol', 'Male']]


def update():
    name = input("Enter the name:")
    for term in data:
        if term[0] == name:
            new_name = input("Enter the new name, empty for no
               change:")
            if new_name == '':
                new_name = name
            while not new_name.istitle():
                new_name = input("Wrong input, please re-enter
                   :")
            term[0] = new_name
```

```
            new_gender = input("Enter the new gender, empty for
               no change:")
            if new_gender == '':
                new_gender = term[1]
            while new_gender not in ['Male', 'Female']:
                new_gender = input("Wrong input, please re-enter
                   :")
            term[1] = new_gender

            print(new_name + ' updated.')
            print(data)
            return

    print(name + ' is not in the data base. Operation failed.')
```

Program to write: Social Network

In this assignment you will face a challenge of modeling one-to-many relationships. We will try to model a social network where one person can have many friends.
In this assignment we provide you with a data file that contains data in the following format:

```
<name of a person> | <name of a person>
<name of a person> | <name of a person>
...
```

Where "<name of a person>" is a string without pipe symbols "|". This notation shows that the person on the right of the pipe is a friend of the person on the left side of the pipe (not the other way around). There can be different amount of whitespace separating "<name of a person>" and pipe symbols, so be sure to trim it.
Your primary goal is to convert this data into a dictionary, where the key is the name of a person and the value corresponds to the person's friends. As you can use only one piece of data for a value, you'll have to use lists that will contain all the friends.
After loading the data from the file to the dictionary, please implement the following functions that work on your dictionary:

1. A function that finds the person with the largest number of friends.
2. A function that find the average number of friends a person has in the network.
3. A function that accepts two names and determines if the second is a friend of the first.

Study further:

Gradually gaining expertise with Python, learning more and more constructs and abstactions of the language, you will often need to convert lists into dictionaries or vice versa. Of course, you can write your own functions to do that, but Python offers several functionalities to achieve such a task. What are they? Especially when it comes to the question of merging lists into the dictionary, you can benefit from built-in function *zip()*. For instance, here you have two lists *dishes* and *countries*, try to merge them into a dictionary *countries/specialities*

```
dishes = ["pizza", "sauerkraut", "paella", "hamburger"]
countries = ["Italy", "Germany", "Spain", "USA"]
```

Once the lists become too big, Python has an *iterator* module with a number of fucntions for efficient looping. We invite you to study the concept of iterators and in particular look at the *itertools.zip_longest*: `https://docs.python.org/3.4/library/itertools.html#itertools.zip_longest`