Università della Svizzera italiana

Faculty of Informatics

Programming Fundamentals I
Fall 2014
Topic 5: Lists

Prof. Mehdi Jazayeri

Assisted by: Anton Fedosov, Yuriy Tymchuk, Rui Xin

Week 5: 20 - 24 October, Due: 27 October

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 5

Topic: This week you will work with lists, files and while–loops.

Introductory questions:

- Write a function that prints the elements of its input list parameter, one element per line.
- Write a function that prints the elements of its input list parameter in reverse order, one element per line.
- Write a function that prints the elements of its input list parameter in reverse order, all on one line.
- Write a function that returns the reverse of its input list.
- Write a function that computes the average of a list of numbers given as a list.
- Write a function that takes a list of integers as input parameter and returns the number of times the most-frequently occurring element appears in the list.
- Write a function that orders a list of integers in ascending order.
- Write a function that removes any duplicates from a list.
- Write a function that finds the minimum or maximum element of a list of integers.
- Write a function that opens a textfile and returns its 10th line as a string.
- Write a function that reads a textile and prints any adjacent lines that are identical.
- Write a while loop to print numbers from 0 to 10.
- Write a while loop to print numbers from 10 to 0.

Program to modify:

Below is a program that creates list a of 20 random integers from 0 until 100. The program sorts the list in ascending order. Later, using a *for* loop, it extracts elements from the interval *[35,65]* from list a and creates a new list b from them. Finally, list b is sorted and printed.

```
import random

# generates a list with 20 random integers from 0 until 100
a = []
for i in range(20):
    n = random.randint(1,100)
    a.append(n)
# sorts a list in ascending order
a.sort()
print("a = ", a)

# creates and populates new list with given condition using for
   loop
b = []
for i in a:
    if 35 < i < 65:
        b.append(i)
b.sort()

print("b = ", b)
```

Your task is to modify the program so that it conforms to the following requirements:

1. Use *while* loop instead of *for* loop to extract the elements from list a and populate list b. Conditions for extraction are the same: list b should consist of the elements of list a from the range *[35,65]*
2. Within the same iteration of the *while* loop, remove those elements from list a that are

moved to list *b*

3. Print the values of the lists *a* and *b*

Program to write: A simple "Database" (Due Friday, 31 October!)

In this section you will implement a simple **database**.

Database is an important technique in Computer Science, which has been used widely in various fields related to **data management**. For example, when you go to the library to find a book, the library management system should be able to know if the book is present, not owned by the library, or already borrowed; when you want to return the book, the library management system should tell the administrator whether the book is returned by the due date. In this case, a database is used to handle such data management.

Basically, a database consist of two parts, *data* and *structure*. *data* represents the content you want to manage, and *structure* represents the way it is organised. A database should provide four basic operations:

- Create. Create database or add new data.
- Read(Retrieve). Read, retrieve, search, or view existing data.
- Update(Modify). Update or edit existing data.
- Delete(Destroy). Delete/remove existing data.

Now you will use Python to write a script *database.py* to implement a **student management system** and add above operations as functions one by one. In this case, you should **use list as the structure to store data**. Later, we will learn a much better way to structure the data for this application.

A. Read From File

The original data is stored in a file *data.txt* which contains several lines like:

Alice,Female,1994
Bob,Male,1995
Carol,Male,1993

For example, in the first line, we have the Student name *Alice*, Gender *Female* and Year of birth *1994*. Write a function *read()* to read the file *data.txt* by line, store the data to a nested list(e.g, [['Alice', 'Female', 1993], ['Bob', 'Male', 1995], ['Carol', 'Male', 1993]]).

B. Formatted printing

After reading and parsing the data, write a function *show()* to print out the data line by line with a header. It should act in Python shell like:

```
>>
>> import database
>> database.read()
>> database.show()
Name            Gender          Year of birth
Alice           Female          1994
Bob             Male            1995
Carol           Male            1993
>>
```

C. Add

Write a function *add()* to add a new item into your database. Pay attention to check

whether the input format is correct. (i.e, first letter of the name should be uppercase, gender should be either 'Male' or 'Female', and Year of birth should be a number.) Here's a sample:

```
>>
>> database.add()
Enter the name: # Here you input 'Dave'
Enter the Gender: # Here you input 'Monkey'
Wrong input, please re-enter.
Enter the Gender: # Here you input 'Male'
Enter the Year of birth: # Here you input '1995'
Add 'Dave'.
>> database.show()
Name            Gender          Year of birth
Alice           Female          1994
Bob             Male            1995
Carol           Male            1993
Dave            Male            1995
>>
```

D. Delete

Write a function *del()* to remove a student from the database by name. Note that you should also take into consideration whether the name exists. Here's a sample:

```
>>
>> database.del()
Enter the name: # Here you input 'Tom'
'Tom' is not in the database. Operation failed.
>> database.del()
Enter the name: # Here you input 'Dave'
'Dave' deleted.
>> database.show()
Name            Gender          Year of birth
Alice           Female          1994
Bob             Male            1995
Carol           Male            1993
>>
```

E. Modify

Write a function *update()* to modify a student's information by name. Also check the existence of the student in the database as well as the input format.

```
>>
>> database.update()
Enter the name: # Here you input 'Tom'
'Tom' is not in the database. Operation failed.
>> database.update()
Enter the name: # Here you input 'Alice'
Enter the new name, empty for no change: # Here you press
   enter
```

```
Enter the new Gender, empty for no change: # Here you press
   enter
Enter the new Year of birth, empty for no change: # Here you
    input 'secret'
Wrong input, please re-enter.
Enter the new Year of birth, empty for no change: # Here you
    input '1993'
'Alice' updated.
>> database.show()
Name            Gender          Year of birth
Alice           Female          1993
Bob             Male            1995
Carol           Male            1993
>>
```

To study further: In this section we invite you to extend the functionality of the program above. Till now you've written a Python script which implements a simple database, which can perform **read**, **create**, **delete** and **update** on a dataset. Try to add more interesting functions as you can think about, such as *sort_by_year()*, merge_two_databases().