Università della Svizzera italiana

Faculty of Informatics

Programming Fundamentals I
Fall 2014
Topic 9: Program design and incremental development

Prof. Mehdi Jazayeri

Assisted by: Anton Fedosov, Yuriy Tymchuk, Rui Xin

Week 9: 17 - 23 November, Due: 24 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 9

Topic: This week you will design a complete program using the knowledge that you have gained during the semester.

Introductory questions:

1. Write a function that accepts any number of arguments and prints out the number of arguments it has been passed.
2. Write a function that accepts any number of integer arguments and returns the sum of its arguments.
3. What does this statement do:

   ```
   for last, first in people:
   print(last, first, people[last, first])
   ```

   Be sure you understand the structure of the dictionary and its keys.
4. Write a function, sort_by_length(words), which accepts a list of words and returns a list sorted by the length of the words in the list. That is, the shortest word appears first and the longest word last.
5. Write a function, divrem(num, den), that returns both the result of integer division of num/den and the remainder of the division.
6. Lookup the definition of the zip() operation in function. Given two sequences s1 == (1, 2, 3), and s2 == ('a', 'b, 'c'), what is the result of zip(s1, s2)?
7. Use the zip operation shown in the previous section to check if, given two sequences t1 and t2, there is an index i such that t1[i] == t2[i].

Program to modify:

None this time.

Program to write:

The US Social Security administration has annual data of what names are most popular for babies born each year in the USA. See: `http://www.socialsecurity.gov/OACT/babynames/`
We are going to utilize the data of the year 2013. The data file *baby2013.html* is provided with the current assignment. Take a look at the html and think about how you might scrape the data out of it.
Your task is to implement the *extractnames(filename)* function which takes the filename of a *baby2013.html* file and returns the data from the file as a single list - the year string at the start of the list followed by the name-rank strings in alphabetical order. The output of the function is then recorded into the *baby2013.txt* file

```
2013
Aaliyah 36
Aaron 51
Abagail 8
```

Rather than treat the boy and girl names separately, we'll just lump them all together. In some years, a name appears more than once in the html, but we'll just use one number per name. Optional: make the algorithm smart about this case and choose whichever number is smaller.
Build the program as a series of small increments, getting each step to run/print something before trying the next step. This is the pattern used by experienced programmers — build a series of incremental milestones, each with some output to check, rather than building the whole program in one huge step.

Printing the data you have at the end of one milestone helps you think about how to re-structure that data for the next milestone. Python is well suited to this style of incremental development. For example, first get it to the point where it extracts and prints the year. Here are some suggested milestones:

- Extract all the text from the file and print it
- Find and extract the year and print it
- Extract the names and rank numbers and print them
- Get the names data into a dict and print it
- Build the *[year, 'name rank', ...]* list and print it
- Write a formatted list to the file, one element per line

It's more re-usable to have the function *return* the extracted data, because then the caller has the choice to print it or do something else with it. (You can still print directly from inside your functions for your experiments during development.)
You are free to use any parsing machanism to fetch the relevant data: *year, name and ranking.* We suggest to do parsing of the HTML page using regular expressions. Python has a module called *re* that is useful for this. To use it, you need to read the documentation first and figure out how to apply it. Check out the examples in the documentation. For this assignment, here are some hints. Since the line containing the year looks like this:

```
<h2>Popularity in 2013</h2>
```

you could use regular expressions to look for it:

```
year = re.search(r'Popularity\sin\s(\d\d\d\d)', text)
```

and for names, since the lines look like this:

```
 <td>1</td> <td>Noah</td> <td>Sophia</td>
```

you can pick up the names and rank like this:

```
entries = re.findall(r'<td>(\d+)</td> <td>(\w+)</td> \<td>(\w+)</td>', text)
```

Study further:

As you have seen in this assignment, regular expressions are very handy for parsing text files. Study them some more!