Computer Programming
Spring 2016

CSI2100-01 Lab 3 (Part 1)

Bernd Burgstaller
Yonsei University

Outline

- Questions
- Programming Problems
- Deliverables, due-date and submission

Questions

- You are kindly asked to submit the answers to the questions on the following page in a file named **README.txt**
- Please refer to the Lab 2 instructions on how to create and edit this file, and how to save it to disk.
- See the slide on "Deliverables" on how to submit your README.txt file.
- Your understanding of the submitted answers will be checked during the individual code counselling session.

- **Question 1:** Exercise 6 from p. 74 of the textbook

- **Question 2:** Exercise 7 from p. 74 of the textbook

- **Question 3**: Exercise 12 from p. 75 of the textbook

Programming Problems

Problem 1: Exercise D3 from p. 77 of the textbook

- **Note:** please do not report partial images. E.g., if the exact number of files that fit on a USB drive would be 5.5 images, you should report 5 images instead. (We assume that there's no point for the user to store 1/2 of an image on a USB drive.)
  Hint: To archive this, you must find a way to compute the floor function $\lfloor x \rfloor$ of a given float value x (the Python 3 math module contains this function). You can assume that the input type is int. For the TIFF format, we assume a 1:1 compression ratio (no compression). Output a right-justified int of fieldwidth 5 in place of ``xxxxx''.

Problem 2: European countries use a 13-digit code, known as the European Article Number (EAN) to identify consumer products (see the example to the right). The EAN is used instead of the 12-digit Universal Product Code (UPC) found in North America. Each EAN ends with a check digit. The purpose of the check digit is to help identify an error in the previous digits. If the check digit does not match the other EAN digits, something is wrong (a scratched digit, a miss-print, ...).

The **procedure to compute the check digit** is as follows:

1. Add the second, fourth, sixth, eighth, tenth, and twelfth digits.
2. Add the first, third, fifth, seventh, ninth, and eleventh digits.
3. Multiply the first sum by 3 and add it to the second sum.
4. Subtract 1 from the total.
5. Compute the remainder when the adjusted total is divided by 10.
6. Subtract the remainder from 9.

Programming Problems (cont.)

For example, consider a product with an EAN of 8691484260008. The first sum is 6 + 1 + 8 + 2 + 0 + 0 = 17, and the second sum is 8 + 9 + 4 + 4 + 6 + 0 = 31. Multiplying the first sum by 3 and adding the second sum yields 82. Subtracting 1 gives 81. The remainder upon dividing by 10 is 1. When the remainder is subtracted from 9, the result is 8 (the last digit of the original EAN code).

Your job is to write a program that asks the user to enter the first 12 digits of an EAN and computes the check digit.

Example:

```
Enter the first 12 digits of an EAN: 869148426000
Check digit: 8
```

Hint: convert the user input to an int, and extract the 12 EAN digits through a sequence of integer division and modulo operations. For example, if **n** is an integer, then **n % 10** is the last digit in **n**, and **n / 10** is **n** with the **last digit removed**. Once you have the 12 individual digits available, it is straight-forward to proceed with the 6-step procedure on the previous page.

Problem 3: Books are identified by an International Standard Book Number (ISBN). ISBNs assigned after January 1, 2007 contain 13 digits, arranged in five groups. For example, 978-0-470-55515-6 is a valid ISBN. The first group of digits (the GS1 prefix), is currently either 978 or 979. The group identifier specifies the language or country of origin.

Programming Problems (cont.)

The publisher code identifies the publisher (470 is the publisher code of Wiley). The item number is assigned by the publisher to identify a specific book (55515 is the code for our textbook). An ISBN ends with a check digit that's used as a checksum for the preceding digits.

Your task is to write a program that breaks down an ISBN entered by the user.

Example:

```
Enter an ISBN: 978-0-470-55515-6
978.................GS1 prefix
0...................Group identifier
470.................Publisher code
55515...............Item number
3...................Check digit
```

Note: The number of digits in each group may vary. You're not allowed to assume that the groups always have the lengths shown in this example. You can test your program with real ISBN numbers found on the back-cover and on the copyright page of a book.

Hint: the main problem to overcome in this case is to split the user-provided input string into the 5 digit groups. To our help, the digit groups are separated by a dash. We can use this with the **split** operation that each string provides (see p. 298 of the textbook). If **isbn** is a string containing a valid ISBN number, then **isbn.split('-')** allows us to retrieve a list of substrings, split across the occurrences of the dash character (try this in the Python shell

Programming Problems (cont.)

to see how it works).

We have already briefly discussed tuples and lists in the previous lecture. The ISBN from the previous example would be split into the list ['978', '0', '470', '55515', '6']. This particular list is a list of strings. Lists of other types are possible as well. Tuple and list literals can be assigned to variables in the same way as integer, float and string literals, for example:

```
mylist = ['978', '0', '470', '55515', '6']
```

See our textbook on p. 130 and following on how you can extract individual elements from a list.

Note 2: the field-width for each digit group plus the dots (...) is 20 characters. This way the digit group descriptions are vertically aligned at their left-hand side (see the example on the previous page). Your program must format the output correctly for arbitrary ISBNs. For example, your program could receive a two-digit publisher code, but still the field-with stays as 20 characters.

Marking Criteria and Plagiarism

- Marking Criteria
  - Score is only given to programs that compile and produce the correct output with Python version 3.5.1.
  - Points are deducted for programs that produce warnings.
  - Points deductions on programming style: please provide comments in your code.
  - Please pay particular attention to the requested output format of your programs. Deviating from the requested output format results in points deductions.

- Plagiarism (Cheating)
  - This is an individual assignment. All submissions are checked for plagiarism.
  - Once detected, measures will be taken for all students involved in the plagiarism incident.

Deliverables

- Please prepare the files for the programming problems and questions. The names of the files, their due-dates and the archive file-name is given in the below table. Please refer to the Lab 2 specification on how to create zip archives.

Problem	File name	Due	Archive name
1	lab3_p1.py	Friday	lab3_part1_<student id>.zip
2	lab3_p2.py		
3	lab3_p3.py		
Questions	README.txt		
4	lab3_p4.py	Monday	lab3_part2_<student id>.zip

Submitting your archives

- You are asked to upload your archives for part 1 and part 2 of this lab on YSCEC.

- Due date for part 1: Friday, 23:00.
- Due date for part 2: Monday, 23:00.

- For instructions on how to upload a file on YSCEC, please see Lab 1.