Computer Programming
Spring 2020

CSI2100-01 Lab 6

Bernd Burgstaller
Yonsei University

Marking Rules Update

- -20% score for wrong file names
  - Wrong filenames complicate the automated marking process. Please help to keep the grading process smooth!
- -20% for missing or insufficient comments
  - Please put brief comments (#...) with the important parts of your code.

Outline

- Questions
- Programming Problems
- Deliverables, due-date and submission
- Notes:
  1) Please consider using the OnlinePythonTutor when debugging your code.
  2) Some lab problems can be tested with Hyeongjae Python.
     - Please refer to the table of deliverables at the end of this assignment for details.
  3) Because of our coronavirus precautions, you are provided with an excerpt of the relevant sections of the textbook together with this lab assignment.

Questions

You are kindly asked to submit the answers to the questions on the following page in a file named **README.txt**

Question 1: Textbook page 163, Exercise 8.

Note 1: you're asked to provide a single instruction (read: a single line of Python code) for both (a) and (b). There is no need for a loop statement or other lengthy code. Please have a look at the list operations in the table on page 25 of Lecture 4 (on lists) to find out how you can solve these problems with little code.
Note 2: for (a), ”*the fourth character of the string*” is the character at index-position 3. (As we discussed in the lecture, Python sequences are zero-indexed.

Question 2: Textbook page 163, Exercise 9.

List `lst=[ [0,1,2], [5,4,3], [0,0,0], [3,2,1] ]` is one example (out of infinitely many) for the nested list. In your answer, you should **not** include the definition of the `lst` variable. Instead, your code **assumes** that the list `lst` has already been defined for you. You can assume that `lst` contains at least 4 sublists (relevant for **(d)**).

For part (a): please provide the number of list elements not counting the elements of the sub-lists. For the list given above, the asked length would be 4.

For part (b): please count all the integer values in the list. (There are 4*3=12 integers in the above example.)

For part (c): please sum up all the integers in the list. (For the above list, the solution is 1+2+5+4+3+3+2+1=21.)

For part (d): ``third integer'' means the integer at index 2. ``fourth element (sublist)'' denotes the sublist at index 3. (Python is zero-indexed. English is 1-indexed ☺).

Question 3: Textbook page 163, Exercise 11.

Please note: parts (a)—(d) are independent of each other.

For example, for part (a), you start with list1 and list2 and execute the assignments stated with (a) and write down the results.

For part (b), you start again with the **initial** lists list1 and list2 and execute the assignments stated with (b) and write down the results.

Same for (c) and (d).

Programming Problems

Problem 1: Write a program that takes the first name and last name entered by the user, and displays the last name, a comma, and the initial letter of the first name, followed by a period.

Example:
```
Enter a first and last name:  Hwangho Park
Park, H.
```

Note 1: the user's input may contain extra spaces before the first name, between the first and last names, and after the last name. You may assume that the user's input contains characters (A—Z, a—z) and spaces (blanks) only.

Note 2: You may use a loop to process the user's input string character-per-character. Store the **start** and **end positions** of first name and the last name in variables and then **slice** the first and last name from the input string.
Update: as an alternative, you may use the split() method provided by strings. This considerably simplifies this example. Depending on your use of split(), a loop may not be required anymore.

Note 3: you are not required to change the casing in the user's name. Any combination of lower-case and upper-case letters can appear in the user's input, and you can replicate this casing in your output. E.g., "hwanGho parK" will become "parK, h.".

Note 4: you can assume that the user's first name is input without blanks. For example, "Hwang ho" will not occur in the input. It must be "Hwangho".

Problem 2: Write a program that asks the user to enter a fraction, then reduces the fraction to lowest terms. You can assume that both numerator and denominator of the fraction are integers > 0.

```
Enter a fraction:   6 /   12
In lowest terms: 1/2
```

Hint 1: Use of the split() method is NOT allowed. Please process the input string character by character to determine the numerator (6 in the above example) and the denominator (12 in the above example). The user input may contain spaces (arbitrary many), digits, and a single slash between the numerator and the denominator.

Hint 2: To reduce a fraction to lowest terms, first compute the greatest common divisor (GCD) of the numerator and the denominator. Then divide both the numerator and the denominator by the GCD. See Lecture 1 on how to compute the GCD using Euclid's Algorithm.

Hint 3: in case the denominator is 1, only the numerator shall be output. That is, x/1 will be replaced by x for all $x \in N$:

```
Enter a fraction:    16/   2
In lowest terms: 8
```

Problem 3: Modify the Password encryption program from Fig. 4-13 on page 143 of the textbook according to the following specifications. We only care about encryption (leave the decryption part unchanged!). Please find the source-code that you should change on YSCEC in the labs section. (Note that the introductory greetings have been removed.)

1. Include the non-alphabetic characters #,@,% and encrypt them as follows:
   # becomes !
   @ becomes (
   % becomes)
2. **Check** that every password contains at least one letter (A-Z) or (a-z), and at least one non-alphabetic character (from the above list), and at least one digit. The program must **reject** passwords that violate this rule.
3. Restrict the characters that are allowed in a password to letters (A-Z) and (a-z), digits (0-9) and the three characters from the above list. The program must **reject** passwords that violate this rule.

```
Enter password: 1abc%@
Your encrypted password is: 1mht)(
```

```
Enter password: *abc
Invalid password!
```

Hint 1: if you find that the user's password violates conditions (2) or (3) from the previous page, you might want to terminate your program. This can be achieved with the `exit( )` function from the sys module. You must import this module in your program in order to use the `exit( )` function. See the following example.

```
import sys
....
if (...):
  print('Invalid password!')
  sys.exit( ) # terminates program
....
```

Hint 2: do not ask the user for further input if a **password** is **invalid**. Your program should terminate in this case. Likewise, the program shall terminate after it has encrypted a valid password.

Problem 4: Textbook page 164, Exercise P5.

- Please output the list **sorted**! To sort the list, please use the `sort()` method provided by lists (see p. 70 of our Lecture 4). The sort() method uses dictionary order and this is the order required for this problem.
- Lower-case `q' **terminates** the program.

Example 1:

```
Enter a word (q to quit): Baboon
Enter a word (q to quit): apple
Enter a word (q to quit): wow
Enter a word (q to quit): enter
Enter a word (q to quit): q
['Baboon', 'enter', 'wow']
```

Example 2:

```
Enter a word (q to quit): q
[]
```

Problem 5: Textbook page 164, Exercise P6. You can assume valid user input in lower-case letters. We use kilos (kg) instead of pounds (lbs). We use lower-case letters for the output, too. All weights are stated as integers.

Example 1:

```
Enter a fruit type (q to quit): apple
Enter the weight in kg: 200
Enter a fruit type (q to quit): apple
Enter the weight in kg: 300
Enter a fruit type (q to quit): mango
Enter the weight in kg: 200
Enter a fruit type (q to quit): q
apple, 500kg.
mango, 200kg.
```

Example 2:

```
Enter a fruit type (q to quit): q
No data received, exiting.
```

Please note that in the output, the list of fruits must be **sorted alphabetically** (= in lexicographic order).

One elegant solution to this problem is to store sublists of the form `[fruit, weight]` in a list. For example: `fruits=[ ['mango', 200], ['apple', 500]...]`.

The question then becomes: how can you sort this list in the order of the fruit-names?

Python allows you to specify which item of the sublists to use for sorting. This is done using the `itemgetter()` function provided by the `operator` module. For example:

```
>>> from operator import itemgetter
>>> mylist=[ ['mango', 22], ['apple', 33] ]
>>> mylist.sort( key=itemgetter(0) )
>>> mylist
[['apple', 33], ['mango', 22]]
>>>
```

We provide the statement `key=itemgetter(0)` as an argument to the sort() function. `itemgetter(0)` selects index 0 (the fruit names) for sorting. If you want to sort the list on index 1 (the weights), you would have to specify `key=itemgetter(1)`.

As shown in the above example, the sort() function **modifies (mutates) the provided list**. It is wrong to assign the sorted list to a new list, e.g., ~~`newlist = mylist.sort(...)`~~.

wrong!

Problem 6: Consider the following infinite series:

$$\sum_{n=1}^{\infty} \frac{1}{n} = 1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \dots$$

Write a program that evaluates this series up to some limit **L** of number of terms. Have the user enter the limit **L** interactively. Run this program in a loop until the user enters 0. You can assume that only integers >=0 will be entered. Use 6 digits after the floating point for the result. For example:

```
Enter the limit L: 1
Sum of the initial 1 term(s): 1.000000
Enter the limit L: 2
Sum of the initial 2 term(s): 1.500000
Enter the limit L: 5000
Sum of the initial 5000 term(s): 9.094509
Enter the limit L: 0
```

$$\sum_{n=1}^{1} \frac{1}{n} = 1$$

$$\sum_{n=1}^{2} \frac{1}{n} = 1 + \frac{1}{2}$$

$$\sum_{n=1}^{5000} \frac{1}{n} = 1 + \frac{1}{2} + \dots + \frac{1}{5000}$$

Use type integer for the limit **L**, and type float for the sum of the terms.

Marking Criteria and Plagiarism

- Marking Criteria
  - Score is only given to programs that compile and produce the correct output with Python version 3.
  - Points are deducted for programs that are named wrongly. See the list of deliverables for the required file names.
  - 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 (including the ``source'' of the plagiarized code).

Deliverables, Due-date and Submission

- Please prepare the files for the programming problems and questions. The names of the files, their due-date and the archive file-name is given in the below table.

Problem	File name	Due	Archive name	Hyeongjae Python
1	lab6_p1.py	Wednesday April 29 23:00	lab6_<student id>.zip	✓
2	lab6_p2.py			✓
3	lab6_p3.py			✓
4	lab6_p4.py			—
5	lab6_p5.py			—
6	lab6_p6.py			—
Questions	README.txt			—