CSCI 1100 — Computer Science 1 Homework 4
Lists, Loops and If statements

Overview

This homework is worth **100 points** toward your overall homework grade, and is due Thursday, October 17, 2019 at 11:59:59 pm. It has two parts. Part 1 is worth 45 points and Part 2 is worth 55 points.

The goal of this assignment is to work further with loops, lists and if statements. As your programs get longer, you will need to develop some strategies for testing. Here are a few simple ones: solve a simpler version of the problem first without all the special cases, or even solve a simpler and different problem. Work from that solution to the current one.

Start testing early, and test small parts of your program by writing a little bit and testing. Ask yourself what is a good input and what is the expected output first, instead of waiting for Submitty inputs. This will allow you to understand the problem better.

Remember to use office hours, especially earlier in the week, for conceptual questions.

As always, make sure you follow the program structure guidelines. You will be graded on program correctness as well as good program structure.

Remember, we will be continuing to test homeworks for similarity. So, follow our guidelines for the acceptable levels of collaboration that we posted in the Submitty Course Materials.

Part 1: Is this crazy password valid? (45 Points)

A good password (in this strange system) should have the following properties:

Rule 1: The password should be between 10 and 25 characters inclusive, starting with a letter.

Rule 2. The password should have at least one of the following characters `@ $` but no `%`.

Rule 3: The password should have at least one upper and one lower case character, or it should contain at least one of 1, 2, 3 or 4.

If it contains upper and lower case characters, we don't care about numbers 1,2,3,4. If it contains 1,2,3 or 4, we don't care about upper or lower case characters.

Rule 4: Each upper case character (if there are any) must be immediately followed by at least one underscore symbol (`_`).

Rule 5: Each numerical digit, if there are any, must be less than 4. (So, `a35` is not valid because of 5). Note that in Rule 3 you checked the existence of one at least number in 1-4 range, now you want to check that all numbers are on this range.

Write a program that asks the user for a potential password, then checks and prints whether it satisfies each of the above rules.

Then, if the password is a valid one (satisfying all the above rules), tell the user. If the password is not valid, but satisfies rule 1, suggest a valid password by taking the first 8 and the last 8 characters and appending `42` between them (note the password constructed this way may not be valid with respect to the above rules, but randomness is always good for security). Otherwise, you print nothing.

Hint: A very good way to solve this problem is to use booleans to store whether each rule is satisfied or not. Start slowly, write and test each part separately. Rules 1,2 and 3 do not require looping at

all. For rule 4, the simplest solution involves simply looping through the string once. The same is true for rule 5. You can construct the suggested password, in one line of code using slicing.

There are some really useful string functions you can use: `str.islower`, `str.isupper` and of course `str.count` and `str.find`. Try help to learn more about them.

Here is an example on how to get started slowly. It uses Booleans to store the result of some simple steps and provides a platform to build up your solution.

```
is_long_enough = len(word) <= 25 and len(word) >= 10
is_lowercase = word.islower()
if is_long_enough and is_lowercase:
   print ("It is long enough and lower case")
```

Here are some s runs of the full program:

```
Enter a password => this_is_#lowercase
this_is_#lowercase
Rule 1 is satisfied
Rule 2 is not satisfied
Rule 3 is not satisfied
Rule 4 is satisfied
Rule 5 is satisfied
A suggested password is: this_is_42owercase
```

```
Enter a password => M_ixed@casinG
M_ixed@casinG
Rule 1 is satisfied
Rule 2 is satisfied
Rule 3 is satisfied
Rule 4 is not satisfied
Rule 5 is satisfied
A suggested password is: M_ixed@c42d@casinG
```

```
Enter a password => short1@%pass
short1@%pass
Rule 1 is satisfied
Rule 2 is not satisfied
Rule 3 is satisfied
Rule 4 is satisfied
Rule 5 is satisfied
A suggested password is: short1@%42t1@%pass
```

```
Enter a password => with2$valid3numbers
with2$valid3numbers
Rule 1 is satisfied
Rule 2 is satisfied
Rule 3 is satisfied
Rule 4 is satisfied
Rule 5 is satisfied
The password is valid
```

When you have tested your code, please submit it as `hw4_part1.py`. You must use this filename, or Submitty will not be able to run and grade your code.

Part 2: World-wide University Rankings (55 Points)

In this part, we will use data from `university_data.csv`, a file of world-wide university rankings with data for 1000 universities. The data comes from `https://www.kaggle.com/mylesoneill/world-university-rankings`.

To complete this part, you will import a module named `hw4_util` that works very similar to the `lab05_util` module we used in Lab 5. You will use this module to read in the `university_data.csv` file into a list of lists. Within this list of lists, the first item is a list of data field names (header), and the remaining entries are lists containing information about different universities (the specific university name is at index zero). Here is a short example of how this module works using data for the first five fields from the header and two random rows:

```
import hw4_util

data = hw4_util.read_university_file("university_data.csv")

print (data[0][:5])  ##this is the header item in the list
print (data[400][:5])
print (data[800][:5])
```

Produces the following output:

```
['Institution', 'World rank', 'Country', 'National rank', 'Quality of education']
['Huazhong University of Science and Technology', 400, 'China', 16, 367]
['Showa University', 800, 'Japan', 59, 367]
```

Problem specification

Write a program that first reads the file as shown above into a list of lists using the utility we provided you. The program then asks the user for the name of a university and two indices from the list (between 1 and 1000). You can assume the entered indices are valid.

Search for the entered university name in the data. If the university is not found, provide an error message. If the university is found, print a table comparing it to the other two universities as shown below. Use the format function to pad the fields with spaces – we pad the first field to 25 characters and the remaining fields to 12 characters. You can easily accomplish this with the format function:

```
>>> "{0:12}".format("abc")   ## left justified
'abc         '
>>> "{0:>12}".format("abc")  ## right justified, this is what we use
'         abc'
```

Your program must include a function `find_university(data, uname)` that takes as input the list of lists you have read from the file and the university name you have read from the user. Your function should return the index of the university if it is in the data list or -1 if it is not found.

Here are sample outputs of this program:

```
University name => Rensselaer Polytechnic Institute
Rensselaer Polytechnic Institute
Line number for first university to compare (1-1000) => 120
120
Line number for second university to compare (1-1000) => 680
680
First university: Rice University
Second university: University of L'Aquila

                          First      Second       Yours
            World rank      120         680         282
               Country      USA       Italy         USA
         National rank       61          37         110
   Quality of education      52         367         137
      Alumni employment     161         567         142
    Quality of faculty       95         218         218
          Publications      255         655         402
             Influence      150         678         346
             Citations       59         645         182
          Broad impact      130         590         292
               Patents       76         871         210
                 Score    49.73       44.38       46.19
                  Year     2015        2015        2015
```

A second run:

```
University name => Rensselear Polytechnic Institute
Rensselear Polytechnic Institute
Line number for first university to compare (1-1000) => 1
1
Line number for second university to compare (1-1000) => 10
10
University not found
```

When you have tested your code, please submit it as `hw4_part2.py`. Only submit your code, not the other files we already provided you. You must use this filename, or Submitty will not be able to run and grade your code.