Assignment

January 22, 2016

1 Week 2 Assignment

1.1 Readings

Note that the output of your programs must match the example output exactly. This is needed for automatic grading to work.

1.2 1. The command line and bash scripting

Shell scripts are just files with lines of commands. They are designed to be run at the command line terminal and it's actually a full featured language (although no one programs completely in shell scripts). Shell scripts are an automated way of programming tedious tasks, repeated sets of commands that need to be performed.

For example, you might write one to run a series of python programs every morning or when you deploy a website. You might write one to automatically create a file structure for a project (like a project template).

We'd like you to write a bash script that generates the below file tree and save it as `make_tree.sh`. You do not need to execute this bash file as we will read it manually in order to grade it!

There are several ways of executing bash files. Firstly you can execute them via a command line call like `sh make_tree.sh`.

```
s1
|---s3
|   |---conf.txt
|---s2
    |---text_analyzer1.py
    |---Advanced
         |---text_analyzer2.py
```

`conf.txt` should contain the sentence `I love bash scripting.` on the first line.

`text_analyzer1.py` should prompt the user for a text string. It should then output (1) the number of vowels (aeiou), (2) the first vowel, (3) the character immediately after the first vowel.

For example:

input:

`"Boogie Woogie"`

output:

```
vowels: 8
first vowel: o
character immediately after first vowel: o
```

Your program should print something helpful (not throw an exception) if the input has no vowels.

`text_analyzer2.py` should also prompt the use for a text string. It should then create a dictionary in which the keys are the five vowels, and the values are the number of times each vowel occurs in the text. Print the dictionary.

For example:

input: `Boogie Woogie`
output:

```
{'i': 2, 'a': 0, 'e': 2, 'u': 0, 'o': 4}
```

You may write the python files and copy or move them to the given location or you may write them from within the bash file.

1.3 2. Matrix Inverter

One place tuples are convenient is in the representation of matrices. Label the values in a 2x2 matrix as follows:

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

Write a script, matrix_fun.py, that asks the user for a text string including the four values, a, b, c, and d, separated by spaces. You can use the `split()` method on the string to create a list of the four values in order.

Create a tuple that represents each row, then create a tuple that contains those two tuples. It should have the form `((a, b), (c, d))`. Print this representation.

The inverse of the matrix above is given by the formula,

$$\frac{1}{ad - bc}\begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$$

Compute the inverse of the given matrix, again represented as nested tuples, and print this representation. For example:

input: 1 2 3 4
output:

```
matrix: ((1.0, 2.0), (3.0, 4.0))
inverse: ((-2.0, 1.0), (1.5, -0.5))
```

1.4 3. Playground

Please make at least two commits to the github playground. They must be seperated by at least two days and can be small but they must edit a single file called `edit.txt`. This is to show you how to deal with merge conflicts and multiple editors[ie people editing the repository]. If this file does not exist, please go ahead and be the first to create it.

Feedback is essential for us to improve. Please submit feedback if you have it at this link.