CSCI 1100 — Computer Science 1 Homework 5
Even more lists and loops

Overview

This homework is worth **100 points** toward your overall homework grade (30+30+40 points), and is due Friday, October 17, 2014 at 11:59:59 pm. This is a slightly longer homework, so you have one extra day to finish this homework only.

No external code this time. Points to remember:

- Name your submissions as `hw5_part1.py`, `hw5_part2.py` and `hw5_part3.py`, all lower case. Otherwise, we will not be able to locate and execute your submissions properly.
- As in Homework#3, expect different input cases and input files in the submission server. Your program must work for all possible inputs, not just the ones we test for.
- Remember to print any input you read immediately as we have done in previous homewroks.
- We will grade you on correct program as well as good program structure. We will also continue to check for "excessive collaboration".

Part 1: Functions that change a list

In this part, you will use two inputs corresponding to two people walking in a grid of `(x,y)` coordinates. You know their starting point and the path they took. Here is an example input:

```
Wallace,2,2,up,left,stay
```

which means that:

- `Wallace` was initially at point `(2,2)`.
- Then, he walked up at time 1 to position `(2,3)`.
- Then, he walked left at time 2 to position `(1,3)`.
- Then, he stayed at his current position `(1,3)` time 3.

Your job is to read two strings corresponding to the two paths using `raw_input` and then use this information to print the location of each person and the grid distance between them (total number of grid segments in between) for each time point. For example, the distance between `(5,4)` and `(2,3)` is given by `(5-2)+(4-3)=4`.

Finally, print the max and min distance between the two individuals during their walk.

Clearly, you will need to use the `split` function to process the input strings.

Here is a sample output:

```
Enter the first path ==> Wallace,2,2,up,left,stay,up,right,stay,up,right
Wallace,2,2,up,left,stay,up,right,stay,up,right
Enter the first path ==> Gromit,5,4,down,down,left,stay,left,left,down,down
Gromit,5,4,down,down,left,stay,left,left,down,down
Wallace    Action     Gromit     Action     Distance
(2,2)      initial    (5,4)      initial    5
(2,3)      up         (5,3)      down       3
(1,3)      left       (5,2)      down       5
(1,3)      stay       (4,2)      left       4
(1,4)      up         (4,2)      stay       5
(2,4)      right      (3,2)      left       3
(2,4)      stay       (2,2)      left       2
(2,5)      up         (2,1)      down       4
(3,5)      right      (2,0)      down       6
The minimum distance between them is 2
The maximum distance between them is 6
```

Remember. Correctness and good program structure will be a big part of your grade. You must use a list for a person's current location, i.e `[1,2]`. You must use a function to find the next location of a person by changing the input location and the current command (up, down, left, right, stay). Your function must not return a value, but modify the input location list.

You do not have to match the spacing in the output exactly, but your columns must be aligned. The above output uses 11 spaces between all columns.

Once you are sure your program works correctly, turn in only your code named as `hw5_part1.py`.

Part 2 - Bunnies and Foxes, equilibrium

As promised, we will revisit the bunny and fox example from Lab #2. Remember we had formula for computing the population of bunnies and foxes in the next year given the current population. Here is the functions from Lab # 2 combined into a single function that returns a tuple:

```
def next_pop(bpop, fpop):
    bpop_next = max(0,int((10*bpop)/(1+0.1*bpop) - 0.05*bpop*fpop))
    fpop_next = max(0,int(0.4 * fpop + 0.02 * fpop * bpop))
    return (bpop_next, fpop_next)
```

The population of bunnies and foxes change every year as a function of the above formula. What can we expect in the future? Option 1: bunnies or foxes will totally disappear and never appear again (convergence), option 2: the populations will remain unchanged after a certain point (convergence), or option 3: the population will never reach convergence. You will write a program in this part to check which one is expected to occur.

Your program will include the function `next_pop` given above. You must define a second function called:

`check_convergence(bpop, fpop)`

that takes as input the initial population of bunnies and foxes, and then computes the population of the following year continuously using a **WHILE LOOP**. Your loop stops and outputs the remaining population of bunnies and foxes until any one of the following is true:

(a) No bunnies or foxes (or both) are left, i.e. has population zero (convergence is true),

(b) The population of bunnies and foxes in the current year is identical to the previous year, i.e. population equilibrium is reached (convergence is true),

(c) None of the above is true, but you have computed exactly 100 generations (convergence is false).

Your function, when finished computing the while loop must return a 4-tuple:

`(bpop, fpop, iter, converged)`

where `bpop,fpop` are the final populations of bunnies and foxes, `iter` is the total number of iterations the while loop repeated to reach this number, and `converged` is either True or False based on whether convergence was achieved.

Your functions must not have any global variables and you **must use a while loop** in the convergence function.

Your program will use these functions by first asking the user for the current bunny and fox population. Assume the input is a valid integer.

Once a value is given for both populations, you must call function `check_convergence(bpop, fpop)` from your module and print out the returned values. Sample output from two separate runs is given below:

```
Please enter the starting bunny population ==> 100
100
Please enter the starting fox population ==> 20
20
(Bunny, Fox): Start (100, 20)  End: (0,48), Converged: True in 2 generations
```

```
Please enter the starting bunny population ==> 100
100
Please enter the starting fox population ==> 10
10
(Bunny, Fox): Start (100, 10)  End: (31,29), Converged: False in 100 generations
```

When you are sure your program satisfy the requirements stated here, submit it.

Part 3: Team USA!

In this part, you will run a fantasy soccer league, a fantasy like Team USA advancing to the quarter finals in the World Cup. You are using the FIFA rules from homework# 3 to decide which of the two teams are higher ranked.

Write a program to read the current standing of two teams in the league using a comma separated string of the form:

```
win,loss,draw,goals for, goals against
```

Now, assume these two teams are playing a game against each other. The game can have scores like 1-0, 1-1, 0-1, 0-0, 2-0, 2-1, 2-0, 0-2, 1-2, 2-2, etc. You will consider all game scores in which a team scores at most 4 goals.

For any possible game score, compute which team will have the higher standing based on the FIFA rules: (1 for team 1 being higher ranked, 2 for team 2 being higher ranked and 0 for both teams being equal). You will print this information in a 5 by 5 board.

For simplicity, assume that the input is in the correct format.

You must use `for` loops to accomplish this. Small differences between your output format and the submission server are acceptable, but your columns must be aligned. The output below uses 5 spaces for each column (and 6 for the first column).

Here is a possible run of the program:

```
Enter team 1 stats ==> 1,1,1,1,1
1,1,1,1,1
Enter team 2 stats ==> 1,1,1,1,1
1,1,1,1,1
Columns are team 1 goals, rows are team 2 goals
Goals|  0    1    2    3    4
-----|------------------------
  0  |  0    1    1    1    1
  1  |  2    0    1    1    1
  2  |  2    2    0    1    1
  3  |  2    2    2    0    1
  4  |  2    2    2    2    0
```

Column 2 and row 1 means for example that the score was `2-1`, and as a result, team 1 has higher standing after the game.

Here is another run:

```
Enter team 1 stats ==> 1,1,0,4,1
1,1,0,4,1
Enter team 2 stats ==> 1,1,0,2,1
1,1,0,2,1
Columns are team 1 goals, rows are team 2 goals
Goals|  0    1    2    3    4
-----|------------------------
  0  |  1    1    1    1    1
  1  |  2    1    1    1    1
  2  |  2    2    1    1    1
  3  |  2    2    2    1    1
  4  |  2    2    2    2    1
```

In this case, team 1 has to win or tie to do better than team 2 due to its goal difference advantage.

Even though you do not have to, your program will become much easier to write with a function like: `winner(team1stats,team2stats,gamescore)` that takes as input the current team stats and the game score, and returns 1,2 or 0 based on the standing of the team after game with the given score. Beware though, your function must not change the team stats so that you can call it many times. The rest can be easily accomplished with a double for loop.