CSCI 1100 — Computer Science 1 Homework 3
If statements, Modules, Lists and While Loops

Overview

This homework is worth **90 points** total toward your overall homework grade (each part is 30 points), and is due Thursday, October 8, 2015 at 11:59:59 pm. There are three parts to the homework, each to be submitted separately. All parts should be submitted by the deadline or your program will be considered late.

Note on grading. As mentioned in past homeworks, correctness and program structure will both be a part of your grade. Use functions, put at least one top comment, name your variables properly and structure your code to put functions definitions before any program code. The homework submission server URL is below for your convenience:

```
https://submit.cs.rpi.edu/index.php?course=csci1100
```

Your programs for each part for this homework should be named:

```
hw3_part1.py
hw3_part2.py
hw3_part3.py
```

respectively. Each should be submitted separately.

Here are some useful functions for this homework. `str.isdigit()` checks if a string contains a positive integer, `str.ljust(x)` formats the input string so that it is left justified and has length `x`. Similarly, you can use `str.rjust(x)` to right justify a string. Justification is needed for parts 1 and 3, and checking for a valid integer is needed for part 3.

```
>>> '8'.isdigit()
True
>>> 'ab'.isdigit()
False
>>> '1'.ljust(3)
'1  '
>>> '1'.rjust(2)
' 1'
>>> '12'.rjust(2)
'12'
```

Fair Warning About Excess Collaboration

For HW 3 and for the remaining homework assignments this semester we will be using software that compares **all** submitted programs, looking for inappropriate similarities. This handles a wide variety of differences between programs, so that if you either (a) took someone else's program, modified it (or not), and submitted it as your own, or (b) wrote a single program with one or more colleagues and submitted modified versions separately as your own work, this software will mark these submissions as very similar. Both (a) and (b) are beyond what is acceptable in this course — they are violations of the academic integrity policy. Furthermore, this type of copying will prevent

you from learning how to solve problems and hurt you in the long run. The more you write your own code, the more you learn.

Here are some strategies for collaborating in the homework and learning from it:

- It is perfectly acceptable to work on a solution with a friend, debug each others' code, etc. However, when you are finished, you must write your own code, from scratch. You must not copy from a solution you worked with someone. This is a test for yourself. If faced with the blank page, can you write a program? Also remember, repetition is how you memorize the basic patterns. If you write lots of code, you will see common patterns and start to abstract from them. That is what programming is all about.
- When working with TAs, mentors, tutors or any other helpful people who seem to materialize when you are writing programs, ask them to give you an example on a piece of paper or their own computer. Then, write it yourself without looking at it. You are cheating yourself if someone stands behind you and tells you what to write. You are not learning to code if you do this, you are just a typist. Challenge yourself continuously.
- Unfortunately, you must also consider that not everyone is well meaning. Do not leave your code around. Do not give your code to people who will not follow the code of conduct that we just described. If they copy your code and turn it in, you will be in trouble too.

To protect those of you who follow these rules of conduct, we will monitor similarities between programs. Students whose programs are quite similar will be given a chance to explain their source of similarities. The standard penalty for submitting work that is not your own, or for providing code that another student submits (perhaps after modification) will be

- 0 on the homework, and
- an additional overall 5% reduction on the semester grade.

Penalized students will also be prevented from dropping the course. More severe violations, such stealing someone else's code, will lead to an automatic F in the course. A student caught in a second academic integrity violation will receive an automatic F.

By submitting your homework you are asserting that you both (a) understand the academic integrity policy and (b) have not violated it.

Finally, please note that this policy is in place for the small percentage of problems that will arise in this course. Students who follow the strategies outlined above and use common sense in doing so will not have any trouble with academic integrity.

Part 1: FIFA scoring rules

In this homework, we will be making use of a module called `hw3_util.py` that provides a function for this part of the homework. To get started with this part, unzip the folder given to you containing the input files and this util file. You will start with reading the file containing information about teams in the championship league. Try the following:

```
import hw3_util
teams = hw3_util.read_fifa()
print teams[0]
```

The function called `read_fifa()` will return you a list of teams. Each item in the list is information for a team given in a list containing the following information:

`[team, games_played, win, draw, lose, goals for, goals against]`

Write a program to compare any two teams from this list and to print which team is superior. Here are the rules:

- The team with the highest points is superior. Points are calculated as follows: each win is 3 points and each draw is 1 points.
- If the points are the same, then the team with the higher goal difference is superior. The goal difference is compuated as follows: subtract the number of goals against from the number of goals scored.
- If the two teams have the same goal difference, then the team with the higher number of goals scored is superior.
- If the two teams are the same with respect to all of the above rules, we will assume that they are the same. In this case, print that the two teams are tied. FIFA has even rules for breaking the ties for that. But, we will stop here.

Your program must ask the index of any two teams from the list. Assume a valid index from the list is given. Then, it will first print the full information corresponding to these teams, as well the points for each team and the goal difference. The format is given below (the team column is 20 characters long, all other columns are 6 characters long).

You will then compare the two teams corresponding to the input indices, and print which team is greater. Assume valid input only for simplicity. Here are some possible runs of the program:

```
Team 1 id => 1
1
Team 2 id => 2
2

Team                Win   Draw  Lose  GF    GA    Pts   GD
Chelsea             18    10    10    65    46    64    19
Stoke City          11    12    15    36    53    45    -17
Chelsea is better
```

```
Team 1 id => 6
6
Team 2 id => 12
12

Team                Win   Draw  Lose  GF    GA    Pts   GD
Manchester United   28    5     5     89    33    89    56
Manchester City (C) 28    5     5     93    29    89    64
Manchester City (C) is better
```

```
Team 1 id => 5
5
Team 2 id => 4
```

```
4

Team                Win   Draw  Lose  GF    GA    Pts   GD
West Bromwich Albion13    8     17    45    52    47    -7
Swansea City        12    11    15    44    51    47    -7
West Bromwich Albion is better
```

When you have tested your code, please submit it as `hw3_part1.py`. Do not submit the input file or the utility module we gave you. Homework server will supply them.

Part 2: Turtle moves!

Suppose you have a Turtle that is standing in the middle of an image, at coordinates (200,200) facing right (along the dimensions). Assume top left corner of the board is (0,0) like in the images.

You are going to read input from the user five times. Though you do not have to use a loop for this part, it will considerably shorten your code to use one. It is highly recommended that you use a loop here.

Each user input will be a command to the Turtle. You are going to execute each command as it is entered, print the Turtle's current location and direction. At the same time, you are going to put all the given commands in a list, sort this resulting list at the end of the program and print it as is (no other formatting).

Here are the allowed commands:

- `move` will move Turtle 20 steps in its current direction.
- `jump` will move Turtle 50 steps in its current direction.
- `turn` will turn the Turtle 90 degrees counterclockwise: right, up, left, down.

If user enters an incorrect command, you will do nothing and skip that command. You should accept commands in a case insensitive way (move, MOVE, mOve should all work).

You must implement two functions for this program:

- `move(x,y,direction,amount)`

  will return the next location of the Turtle as an `(x,y)` tuple if it is currently at `(x,y)`, facing `direction` (direction one of `right, up, left, down`) and moves the given amount.

- `turn(direction)`

  will return the next direction for the Turtle currently facing `direction`, moving counterclockwise.

Now, write some code that will call these functions for each command entered and update the location of the Turtle accordingly. Here is an example run of the program:

```
Turtle: (200,200) facing: right
Command (move,jump,turn) => turrn
turrn
Turtle: (200,200) facing: right
Command (move,jump,turn) => turn
turn
Turtle: (200,200) facing: up
Command (move,jump,turn) => Move
Move
Turtle: (200,180) facing: up
Command (move,jump,turn) => turn
turn
Turtle: (200,180) facing: left
Command (move,jump,turn) => jumP
jumP
Turtle: (150,180) facing: left

All commands entered ['turrn', 'turn', 'Move', 'turn', 'jumP']
Sorted commands ['Move', 'jumP', 'turn', 'turn', 'turrn']
```

In the above example, `turrn` is an invalid command, so it has no effect in the Turtle's state. In case you are wondering, the list is sorted by string ordering, which is called lexicographic (or dictionary) ordering.

When you have tested your code, please submit it as `hw3_part2.py`.

Part 3: Return of the box

In this part of the homework, you will do a variation of the box problem. As you have now become a CS1 master, this part is substantially more difficult. You will read the height of a box and print a diamond of this height (with line numbers on the left side). You will also ask for the foreground and background characters to use in the printing.

You will also do some error checking in this case. You will print an error message for each error. If there are no errors, only then, you will print the box. Here are the errors you should check in this order (assume that if height is a digit it is at least 7 and do not check this one as we have not learnt nested if statements yet):

1. Is the height a number? Check using `str.isdigit()` and if not print `Please enter a number for height`.
2. Is the background character is a single character? If not print `Please enter one character for background`.
3. Is the foreground character a single character? If not print `Please enter one character for foreground`.

Here is a possible run of the program on Wing IDE:

```
Background char => .
.
Foreground char => @
@
Height => 7
7

1.......@@.......
2......@..@......
3.....@....o.....
4....@......@....
5.....@....@.....
6......@..@......
7.......@@.......
```

We have analyzed the technical specifications for the diamond and found a significant flaw: on line 3, there is a single `'o'` at the end of the diamond, instead of the foreground character. This incorrect character appears on the third line of the diamond regardless of its height. Your code must replicate this flaw.

Your output must match ours for full credit, but there will be lots of partial credit. Note a few things:

- The diamond is padded with the background character on the left and right (starting with `height` number of characters on each side of the diamond.)
- When the height is even, we repeat the middle line.
- The line numbers are right justified. When the max line number is double digit, then single digit line numbers are padded with a space on the left (I used `str.rjust(2)` for this, see below for examples.)
- Clearly, you need loops. Think about first writing a loop to do top part of the diamond. Then, write a separate loop for the bottom part. Then, finally add a few if statements to handle the special cases.

Here are some other possible outputs:

```
Background char =>

Foreground char => *
*
Height => 8
8

1        **
2       *  *
3      *    o
4     *      *
5     *      *
6      *    *
7       *  *
8        **
```

```
Background char => .
.
Foreground char => @
@
Height => 12
12

 1............@@............
 2...........@..@...........
 3..........@....o..........
 4.........@......@.........
 5........@........@........
 6.......@..........@.......
 7.......@..........@.......
 8........@........@........
 9.........@......@.........
10..........@....@..........
11...........@..@...........
12............@@............
```

Also, error handling:

```
Background char => ab
ab
Foreground char => cd
cd
Height => ef
ef

Please enter a number for height
Please enter one character for background
Please enter one character for foreground
```

When you have tested your code, please submit it as `hw3_part3.py`.