Computer Science 1 (1021) -- Spring 2013
Lab & Homework 10
Text Games!

Topics covered: Top-Down Design, Helper Functions

Demonstration due end of lab (show progress) March 27, 2013
HW due on Blackboard by midnight ***Monday April 1***

For this lab you'll be making a text adventure game, similar to Zork, or the game outlined in the book. You should submit a 'L10.py' file completely implementing your game, no external files allowed. After loading your 'L10.py' file, the user can start the game by running the 'runGame()' function.

The game runs by displaying instructions on how to play the game, states the users location, any extra information, and presenting them with possible directions/actions to go. After the user has chosen the direction/action (either with a pop-up box or on the command-line), the game presents the new location and the process continues. A user may exit at any time by replying with the 'Exit' direction/action.

Below are your required features:

1. When starting the game, or when the user types 'Help' as a direction/ action, the game should show a description of the game, with general actions allowed.

2. At least 5 areas. You can have rooms, locations, or whatever you call them, but there must be at least 5 areas a user can navigate between.

3. Description of your location. Whenever the user moves in your game it should print out the new location and a short description, similar to the text on P222-P225 of our book.

4. At least 6 directions/actions. The game must allow at least 6 actions in each location {North, South, East, West, Exit, Help}. You may use N,S,E,W, (or lower case) just be consistent and make sure Help describes this. Exit quits the game. Some directions may be invalid and will keep the user in the same spot. During the description of the location, the game MUST tell the user the possible actions allowed.

5. Special direction. In at least one area, allow a special direction such as 'Down', 'Up', 'Jump'. This brings you to a new area. Be sure to tell the user these possible directions.

6. Include at least one action in at least one room, such as 'GrabBomb' or 'DropBomb'. This can be chosen when the user enters a location (think of it as another direction). Doing an action does not move the user. This also

means a user can have an item (or items). Make sure you inform the user as to what items they currently have.

7. Win/Lose. Include the ability for a player to win or lose the game. This could be finding some spot, or being in some location while possessing some item. Similarly allow the user to lose.

Your game can be built in any way, but the architecture in Chapter 9 is a good starting point. Be sure to name your functions to describe what they do, include a docstring, and put at least a comment in each function.

In class Monday and Wednesday we'll cover some alternative ways of building these games using lists and dictionaries.

Grading:

1. 10% - File headers, function headers and comments. Be sure to insert comments (at least one) describing any complex operation into each function. Include your UC login in the file header next to your name. Limit your line width to 80 characters.
2. 10% - File name as specified, paper copy, and runGame() function.
3. 80% - Operation. 10% per feature above.

Extra Credit: 5% extra credit for each of the following extra functionality.

1. Area visit counter. Keep track, and display to the user, how many areas they have visited throughout their game.

2. Location map. After the user picks a new direction, draw (programmatically) a map of the world and where the user is within it. Use show() to show the generated image to the user.

3. Random behavior. Incorporate some random behavior (import random, then random.randint(min,max)) into your game. Examples are trying to push down a door or doing battle with an adversary.