Program Arcade GamesWith Python And Pygame

Lab 7: Adventure

7.1 Description of the Adventure Game

One of the first games I ever played was a text adventure called Colossal Cave Adventure

(labs/adventure/castle_01.png)

(http://en.wikipedia.org/wiki/Colossal_Cave_Adventure). You can play the game on-line here (http://www.web-adventures.org/cgi-bin/webfrotz?s=Adventure) to get an idea what text adventure games are like. Arguably the most famous of this genre of game is the Zork (http://en.wikipedia.org/wiki/Zork) series.

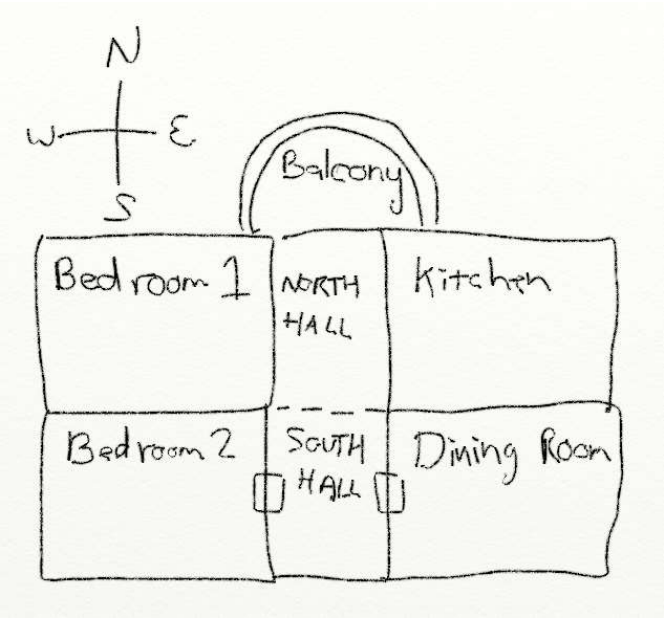
The first "large" program I created myself was a text adventure. It is easy to start an adventure like this. It is also a great way to practice using lists. Our game for this lab will involve a list of rooms that can be navigated by going north, east, south, or west. Each room will be a list with the room description, and then what rooms are in each of the directions. See the section below for a sample run:

7.2 Sample Run

```
You are in a dusty castle room.
Passages lead to the north and south.
What direction? n

You are in the armory.
There is a room off to the south.
What direction? s

You are in a dusty castle room.
Passages lead to the north and south.
What direction? s

You are in a torch-lit hallway.
There are rooms to the east and west.
What direction? e

You are in a bedroom. A window overlooks the castle courtyard.
A hallway is to the west.
What direction? w

You are in a torch-lit hallway.
There are rooms to the east and west.
What direction? w

You are in the kitchen. It looks like a roast is being made for supper.
A hallway is to the east.
What direction? w

Can't go that way.
You are in the kitchen. It looks like a roast is being made for supper.
A hallway is to the east.
What direction?
```

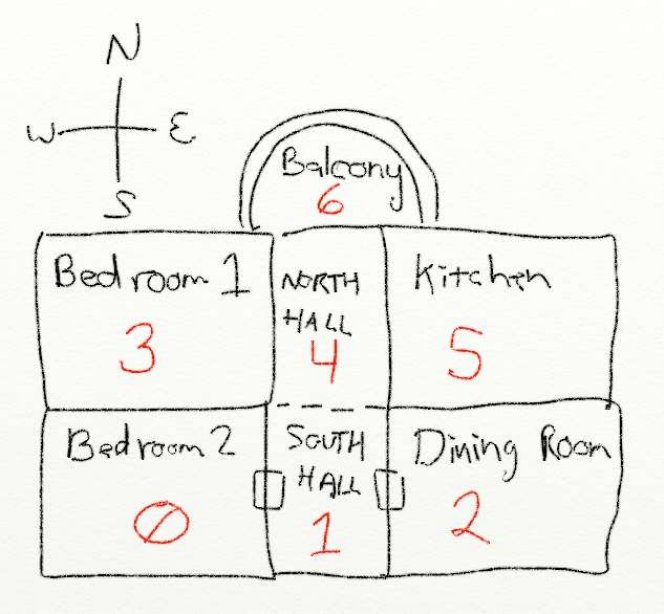
7.3 Creating Your Dungeon

Before you start, sketch out the dungeon that you want to create. It might look something like this:



(labs/adventure/castle_map_01.png)

Next, number all of the rooms starting at zero.



(labs/adventure/castle_map_02.png)

Use this sketch to figure out how all the rooms connect. For example, room 0 connects to room 3 to the north, room 1 to the east, and no room to the south and west.

7.4 Step-by-step Instructions

1. Create an empty array called **`room_list`**.
2. Create a variable called **`room`**. Set it equal to an array with five elements. For the first element, create a string with a description of your first room. The last four elements will be the number of the next room if the user goes north, east, south, or west. Look at your sketch to see what numbers to use. Use **`None`** if no room hooks up in that direction. (Do not put **`None`** in quotes. Also, remember that Python is case sensitive so **`none`** won't work either. The keyword **`None`** is a special value that represents "nothing." Because sometimes you need a value, other than zero, that represents)
3. Append this room to the room list.
4. Repeat the prior two steps for each room you want to create. Just re-use the **`room`** variable.
5. Create a variable called **`current_room`**. Set it to zero.
6. Print the **`room_list`** variable. Run the program. You should see a really long list of every room in your adventure. (If you are using an IDE like Wing, don't leave it scrolled way off to the right.)
7. Adjust your print statement to only print the first room (element zero) in the list. Run the program and confirm you get output similar to:

```
['You are in a room. There is a passage to the north.', 1, None, None, None
```

8. Using **`current_room`** and **`room_list`**, print the current room the user is in. Since your first room is zero, the output should be the same as before.
9. Change the print statement so that you only print the description of the room, and not the rooms that hook up to it. Remember if you are printing a list in a list the index goes after the first index. Don't do this: **`[current_room[0]]`**, do **`[current_room][0]`**

```
You are in a room. There is a passage to the north.
```

10. Create a variable called **done** and set it to False. Then put the printing of the room description in a **while** loop that repeats until **done** is set to **True**.
11. Before printing the description, add a code to print a blank line. This will make it visually separate each turn when playing the game.
12. After printing the room description, add a line of code that asks the user what they want to do.
13. Add an **if** statement to see if the user wants to go north.
14. If the user wants to go north, create a variable called **next_room** and get it equal to **room_list[current_room][1]**, which should be the number for what room is to the north.
15. Add another **if** statement to see if the next room is equal to **None**. If it is, print "You can't go that way." Otherwise set **current_room** equal to **next_room**.
16. Test your program. Can you go north to a new room?
17. Add **elif** statements to handle east, south, and west. Add an **else** statement to let the user know the program doesn't understand what she typed.
18. It is a great idea to put blank lines between the code that handles each direction. I don't mean to print a blank line, but actually have blank lines in the code. That way you visually group the code into sections.
19. It is a great idea to add comments too, to each section.
20. Test your program. Make sure you have enough of a description that someone running the program will no what direction to go. Don't say "You are in the kitchen." Instead say "You are in the kitchen. There is a door to the north."
21. Optional: Add a quit command. Make sure that the program works for upper and lower case directions. Have the program work if the user types in "north" or "n".

Spend a little time to make this game interesting. Don't simply create an "East room" and a "West room." That's boring.

Also spend a little time to double check spelling and grammar. Without a word processor checking your writing, it is important to be careful.

Use \n to add carriage returns in your descriptions so they don't print all on one line. Don't put spaces around the \n, or the spaces will print.

What I like about this program is how easy it is to expand into a full game. Using all eight cardinal directions (including "NorthWest"), along with "up" and "down" is rather easy. Managing an inventory of objects that can exist in rooms, be picked up, and dropped is also a matter of keeping lists.

Expanding this program into a full game is one of the two options for the final lab in this course.

English version (http://programarcadegames.com/index.php?lang=en) by Paul Vincent Craven (http://simpson.edu/author/pcraven-2/)

Spanish version (http://programarcadegames.com/index.php?lang=es) by Antonio Rodríguez Verdugo

Russian version (http://programarcadegames.com/index.php?lang=ru) by Vladimir Slav

Turkish version (http://programarcadegames.com/index.php?lang=tr) by Güray Yildirim

Portuguese version (http://programarcadegames.com/index.php?lang=pt-br) by Armando Marques Sobrinho (mailto:marquessbr@gmail.com) and Tati Carvalho

Dutch version (http://programarcadegames.com/index.php?lang=nl) by Frank Waegeman

Hungarian version (http://programarcadegames.com/index.php?lang=hu) by Nagy Attila

Finnish version (http://programarcadegames.com/index.php?lang=fi) by Jouko Järvenpää

French version (http://programarcadegames.com/index.php?lang=fr) by Franco Rossi

Korean version (http://programarcadegames.com/index.php?lang=ko) by Kim Zeung-Il

Chinese version (http://programarcadegames.com/index.php?lang=cn) by Kai Lin