# Language Chosen: Python 3

# Technical Reasons:

1. The list comprehension and the dictionary features help create easy and flexible data structures
2. The code and syntax are human readable
3. Less code can achieve powerful functions
4. The in-built libraries help carry out a lot of tasks very easily

# Experience Description

## Liked Features

1. The powerful list structures and data structures
2. Dynamically typed variables
3. Easy and succinct code syntax
4. Numerous external and inbuilt libraries available to do anything you want to do

## Dislike Features

1. The indentation scoping was a huge pain because one single space would bug the whole program
2. Being a dynamically typed language, hard to track the error that would be relatively very easy in a statically typed language

## Problems Learning the language

1. None

# How to run the program

1. You need to have Python 3 installed on your computer.
2. You can now locate the Longana.py file and run “python3 Longana.py” from your terminal to run the program

# Bug Report

1. After moving to the next round, the hand is not distributed evenly

# Features Implemented

1. The computer uses AI logic, a version of best-first-search, to come up with its move and provides hint with the same logic
2. Very clear and beautiful looking UI

# Description of Data Structures and Classes

## Classes:

1. **layout** class to implement the layout
2. **hand** class to hold the hand of a player
3. **stock** class to hold the stock/boneyard for the round
4. **player** class in which playing strategies must be implemented
5. **human** class which inherits from player class and implements all the validation of input from the human player
6. **computer** class which inherits from player class and implements all the strategies used by the computer
7. **round** class which implements the entire round
8. **tournament** class which implements a tournament
9. **Longana:** class to initialize the players, tournament and load game from file.

## Data Structures:

1. Lists for storing all the tiles
   1. Left, right, top and bottom for layout
   2. Implemented as a stack for boneyard
   3. Hand
2. Layout is stored as a dictionary of lists

# Log:

12/08/17

* Wrote all the classes for the foundation
* Generated a shuffled stock
* Generated handWorked on creating layout
* Working on turn logic
* Working on place where to print and gameplay

4 hrs

12/09/17

* Round begin and engine logic added
* Placement of the tile logic and validation added for Human
* UI strategy added for Human
* Implemented round end logic
* Implemented score logic
* Haven’t tested the last two yet
* Need to implement checkingFor moves and computer play logic

3 hrs

12/11/17

* Added hasMoreMoves() logic on the Player class. This is to check if the player has any move he/she can make before letting user draw from the stock or pass
* Set up basic AI Logic for hint/ computer move
* First get all the possible moves from the hand including the layout.
* If no moves return empty
* Then sort them in an descending order on the sum of tiles
* If there are singles serve the first single
* Else serve the first double

2 hrs

12/12/17

* Implemented computer.play() using the hint logic.
* Working on loading the new gameComputer AI/ Human hint logic works properly. When playing double the AI will suggest to play on your side only when you have matching move on your hand for later. Tournament scores printed properly
* Correctly loaded the game from the file

3 hrs

09/25/17

* Fixed crosswise display
* Printed hand after drawing from stock

2 hrs

9/27/17

* Polishing and finishing up code
* Commenting and documenting the code
* Allowed save right after dealing and loading file in those states

2 hrs

Total: 16 hrs

# Screenshots:

# ../../../../Desktop/Screen%20Shot%202017-12-16%20at%2011.59.03%20

Figure 1: Start of the game

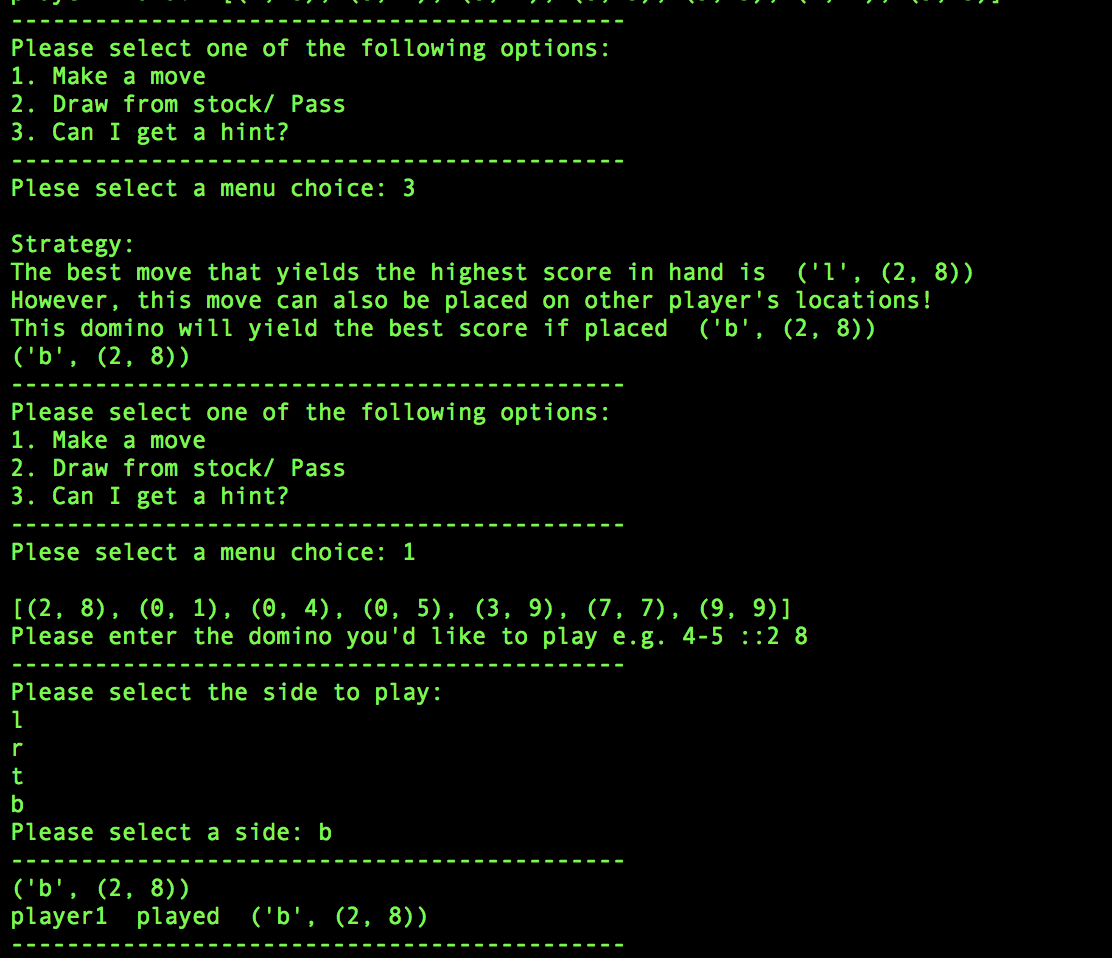


Figure 2: Human player making the move with after asking hint

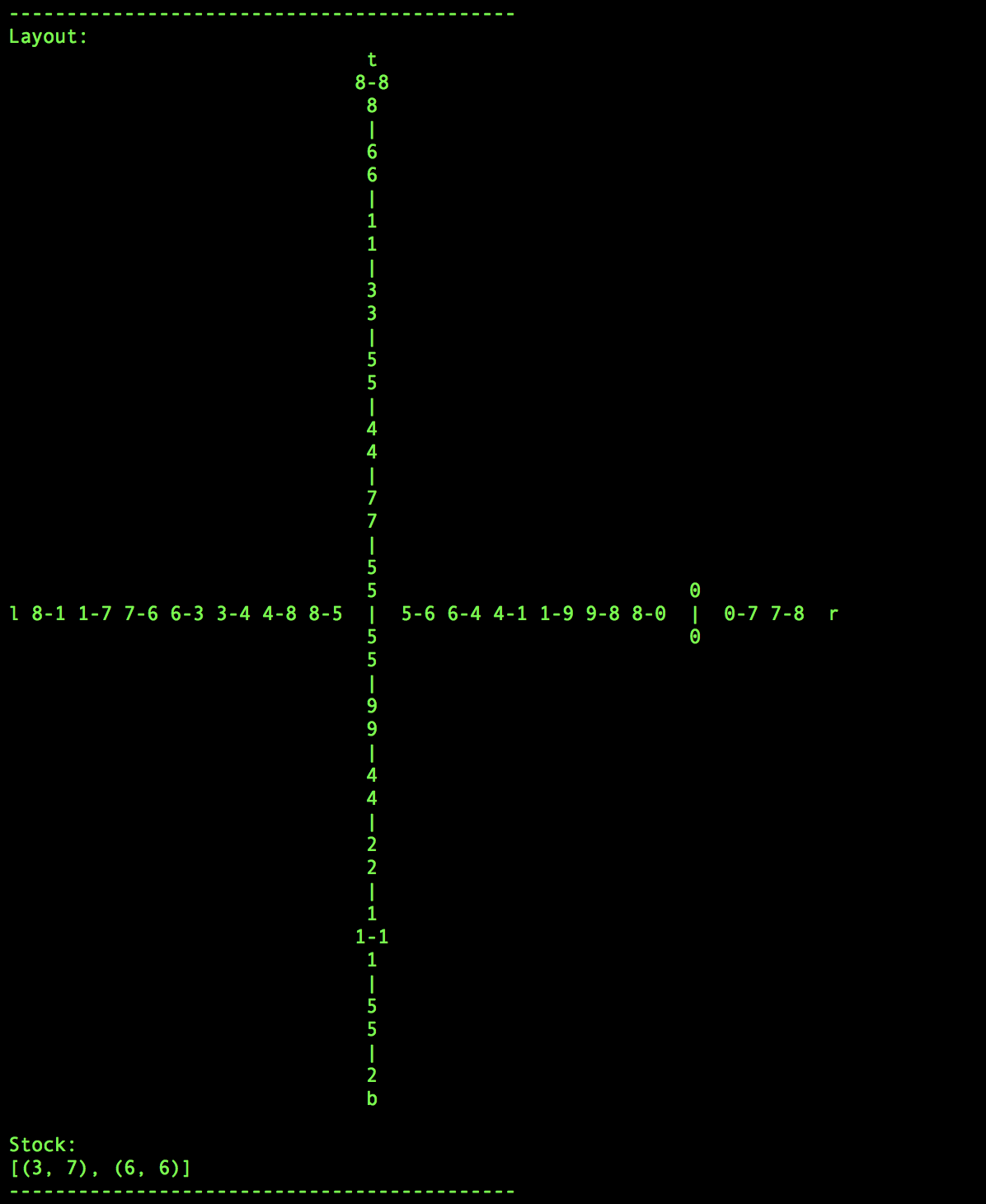


Figure 3: Game components

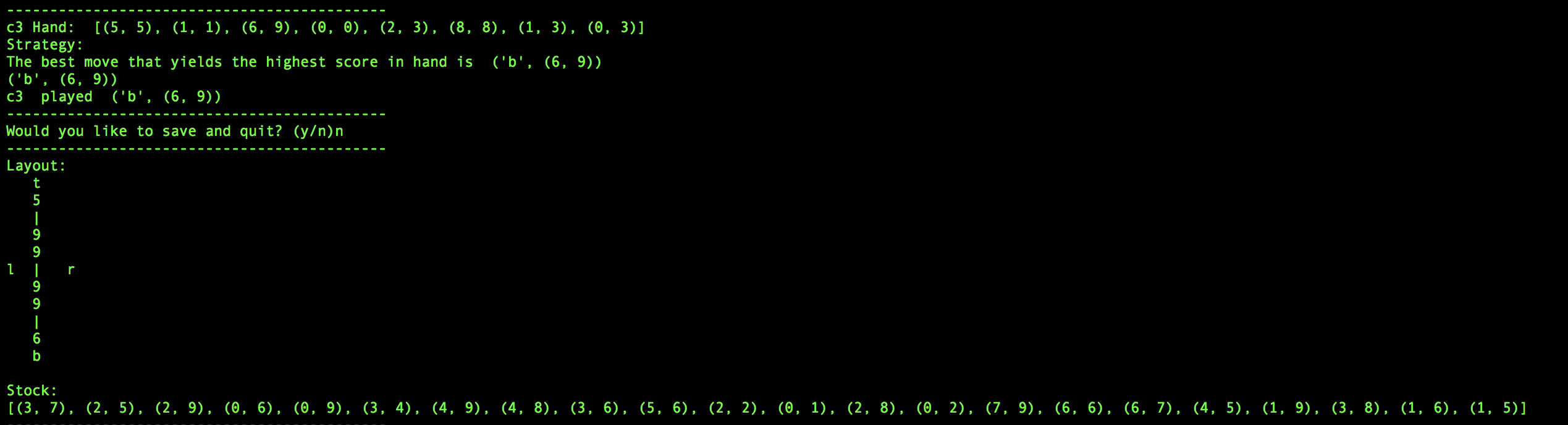


Figure 4: Computer player making the move