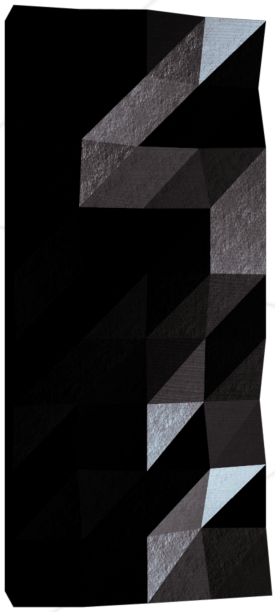

Game Design 1: Command-Line Games

Build your own game logic

Kai kai@42.us.org

Summary: For the first week, practice your mastery of Python by creating a game that runs in the terminal and responds to your input.

Contents

Chapter I

Ideas

For this assignment you should pick any game that you know the rules of, and which doesn't need fancy graphics in order to accomplish the basics of the game. There are more options for text based games than you might guess! Anything based on a grid based board, playing cards, or words works pretty well. Consider the following list of ideas:

- Mastermind
- Tic-Tac-Toe
- 3d Tic-Tac-Toe
- Connect 4
- Sudoku
- Minesweeper
- Hangman
- Word Search
- Game of Life
- Cryptographic Cipher Encoder/Decoder
- Flash cards quiz game
- Base converter
- Metric to Imperial converter
- Matching game (guess two cards on the board until you find pairs)
- Go Fish
- War

- Othello

Spend this week coding a really nice version of one of these in the command line. Work with a pair or a group, so you can divide up the work and help each other debug!

On Friday we will have a showcase of all projects and vote for a winner. :)

Next week we will code some more games with graphics. You may choose to add graphics to this one, or build a different one.

Chapter II

Hints

Here are some hints to get you on the right track.

1. A good place to start is to figure out how to print out the board, or what symbols you will use to represent cards/symbols in your game. Don't forget to put the code for printing the board in a function, so that you can reuse that and print it out many times.

2. An infinite while loop is helpful for allowing players to give input until they win or they lose. Use the keywords 'break', 'continue', or 'pass' to control what happens in the loop.

3. If you clear the screen before every print out of the board (there is a control code for that), you can get it to look like an animation.

4. You can use a two-dimensional list (a list of lists) to represent a grid; or just one long list.

5. You can use a string or a list or a tuple to represent a deck of cards.

6. If you get bored, get fancy with terminal color and extra features.

Chapter III

Guidance

You can look for tutorials online for how to build things like this, but I also recommend that you grab a notebook or a whiteboard, sit away from the computer and brainstorm for yourself how the program would be written.

If you don't know enough about how to use Python, consider switching instead to the Parseltongue Piscine (Intro to Python) class.

As long as you know how to use lists, dictionaries, for/while loops, if statements, printing and input in Python... you have the tools you need! The difficult part is constructing the logic.

I recommend that you imagine how the program will work, and diagram it on paper, before you sit down to actually code much of it. Try this: each member of the group takes some time, say an hour, to imagine and diagram what functions they think should be written to put together the game program. Then, get together and compare notes. Create a group diagram and then split the work from there: each person writes a different function.