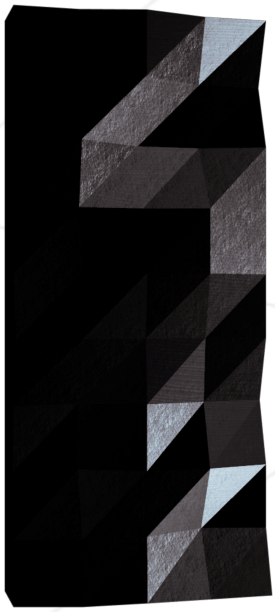

Introduction to Programming - 42

Day 04 - Option 2

Kai kai@42.us.org
Gaetan gaetan@42.us.org

Summary: This document is an alternate subject of the day 04 of the introduction to programming piscine.

Contents

I	**Guidelines**	**2**
II	**Goals**	**3**
III	**General Instructions**	**4**
IV	**Gameboard!**	**5**
V	**Printing**	**6**
VI	**Making the First Move**	**7**
VII	**Looping**	**8**
VIII	**Validity**	**9**
IX	**Did I Win?**	**10**

Chapter I

Guidelines

- Corrections will take place the next morning. Each person will correct another person accoring to the peer-corrections model.
- Questions? Ask the neighbor on your right. Next, ask the neighbor on your left.
- Read the examples carefully. The exercises might require things that are not specified in the subject...
- Your reference manual is called Google / "Read the Manual!" / the Internet / ...

Chapter II

Goals

Take a gentle introduction to the process of building a more complicated program.

Chapter III

General Instructions

If you finish all the exercises of the day, you will have built a working Tic Tac Toe program on the command line.

We will display a game board, and take instructions from two players about where to put their pieces. The game board can be updated to show piece placement and displayed each time. Somewhere in there you can write a function to check and determine when someone has won =)

You will turn in one single .rb file at the end. Go ahead, create a new file: tictac.rb

Chapter IV

Gameboard!

- How do we play tic tac toe? It starts with drawing a # ... two sets of perpendicular lines. A grid.
- Decide for yourself how to design a game board. It could look something like this:

```
 x | o |
-----------
  | o | o
-----------
  |   | x
```

- Be careful and plan ahead! As the game progresses, we want to be able to store and display values of o, x, or nil in each square.
- How will you save the data? It could work multiple ways. For example, you could use nine different variables...

```
square_0 = nil;
square_1 = nil;
square_2 = 'x';
square_3 = nil;
...
```

- Or an array?

```
board = Array.new
board[2] = 'x'
```

- Create the variable(s) that will hold your game board, and then take a look at the next page for discussion of how to print them.

One of these is actually a better choice than the other. As you go through, think about which one and why.

Chapter V

Printing

- When you first start the tictac.rb, it should print a welcome message to the terminal, and the empty game board.

```
?> tictac.rb
Welcome to Intergalactic Tic Tac Toe!
   |   |
-----------
   |   |
-----------
   |   |
```

- Later, you will want to print the same game board with something else in the squares besides empty spaces.

- Create a `method` in your program which takes the state of the board as parameter(s) and prints it out.

- Test it by changing the board state a few times within your program, and printing out each version.

```
def print_board parameters
    #your method defintion here
end

#blank board
print_board board

board[2] = 'x'

#after one move
print_board board
```

String interpolation is a crucial concept to this step. Almost every programming language has a way to insert a variable into a string. Read about how it is done in Ruby and feel free to create a simple side program to get some practice!

Chapter VI

Making the First Move

- OK, that's cool. You can print out different versions of the board if you set them yourself from inside the code.
- Now, have your program change the board based on input from the command line.
- Start with Player 1. Ask them for a move, and read what choice they make. Then update that part of the board to hold an "x".
- Print the board before and after.

```
Welcome to Intergalactic Tic Tac Toe!
   |   |
-----------
   |   |
-----------
   |   |

Player 1, it's your turn to set an x on the board. Will you choose a square?
 0 | 1 | 2
-----------
 3 | 4 | 5
-----------
 6 | 7 | 8

2
Interesting choice, Player 1.
   |   | x
-----------
   |   |
-----------
   |   |
```

Do you need to modify the data type of the input received?

Chapter VII

Looping

Awesome! You know how to print the gameboard and store whether there is an x or an o in each square. You also have been able to receive a move from a player and make it real.

- In the main body of your program, add a loop which repeatedly asks Player 1 and Player 2 for their moves.
- It should alternate between the two players (you could code this several different ways) and print the board after each move.
- For now, the loop can be infinite.
- For now, you don't have to verify that the square is blank in order for someone to play there.

Chapter VIII

Validity

Validity checking is incredibly important in real-world programs.

- To prevent cheating, insert a check to make sure that a players' move is valid.
- A move is not valid if it is outside the game board.
- A move is not valid if someone's x or o is already played there.

Chapter IX

Did I Win?

This is starting to look great, guys. Kudos if you have made it this far.

- Create a method in your program which takes the state of the game board as parameter and determines if anyone has won yet.
- This is the most complex part of the program thus far! Start with just trying to figure out if someone has made a horizontal line.
- Add this to your loops so that it checks after each move.
- Print out a congratulatory message if you find that someone wins!

`You are going to need a lot of if statements and some loops.`

`Some simple math can help you check for vertical lines.`