Ruby

A primer

Why is Ruby different from HTML?

- Logic based
- Not used to markup the page
- Purpose: to make decisions

History of Ruby

- Written by Yukihiro “Matz” Matsumoto in the mid 1990s
- Implemented in C
- "I hope to see Ruby help every programmer in the world to be productive, and to enjoy programming, and to be happy. That is the primary purpose of Ruby language."” - Matz

Programming Basics

What is programming?

- Defining commands
- Issuing them
- Ensuring they get executed

Variables - Algebra

- Remember your algebra!
  - $5 + 9 = 14$
  - $5 + x = 14$
  - Solve for x.

Variables: Containers for Values

- x = 5
- y = “Jonathan”
- q = x + r
- Variables can hold many data types.

Data Types

- Float
  - Numbers with decimal points
  - 10.32, 65.323, .32
- Integer
  - Natural numbers
  - 11, 53, 3

Data Types 2

- Strings
  - A line of text
  - “Jon”, “Elephants are awesome”
- Booleans
  - True or False

Data Types 3

- Arrays
  - A collection of values
  - [5,3,4,"omega", 15.3, "cappa"]
- When assigned to variables, can be accessed with brackets[]
  - a = [5,3,4,"omega", 15.3, "cappa"]
  - a[0] == 5, a[4] == 15.3

Hashes

- Another way to store data - similar to an array
- Used to store key => value pairs
- myHash = {“jan” => “January”, “feb” => “February”}
- To access value: myHash[“January”]

Conditions

- A condition is a test for something

```
if (2 == 2)

        puts “2 really does equal 2!”

end
```

Conditions II

- Important key words:
    - if
    - elsif
    - else
    - end

Loops

- Repetitive conditions where one variable inside a block of code changes
- There are a few different kinds:
  - for
  - while
  - each
  - begin...rescue...end

For Loops

```
for i in 0..4
  puts “loop #{i}”
end
```

While Loops

```
u = 5
while u<10
  puts u
  u = u + 1
end
```

Each Loops

- Loop over an array

```
p = [“This”, “is”, “awesome”]
p.each do |item|
   puts item
end
```

Begin...Rescue...End

- Executes a block if the first block fails

```
begin
  “Oranges” + 4
rescue
  “Oranges” + “Apples”
end
```

Methods

- Methods are shortcuts to a block of code
- They take arguments
- They typically return a value

Method Syntax

```
def addTwo(n)
  n + 2
end
```

Objects

- Objects
  - A representation of something in the real world
  - Has properties and methods
  - For instance - a Car has an engine, and a method to start the engine

Object Syntax

```
class Bar
   attr_accessor :brand
   attr_accessor :wheels
   def turn_on_engine
     puts “engine is on”
   end
end
```

Instance vs Class Methods

- Instance methods require a new instance of the class to work (def addTwo)
- Class methods will work without a new instance (def self.addTwo)

Make Yourself a Cheatsheet

- Variables
- Variable types
- Loops
- Functions
- Objects