Discussion 1 (9/3)

- First 20 minutes
    - Briefly go through **P0** (if most students have not done it / need help)
        - Show class github repo and how to access P0
        - Go through different subsections of P0 README for different OS types
        - Remind students to clone the github repo, can demo if needed

    - Leave time for students to work on it a bit and ask any questions about setup

- Last 30 minutes
    - Walk through some Ruby examples in IRB - before output is shown in shell, you can ask students what the example should output to make it more interactive
        - Mention that Ruby is implicitly and dynamically typed
            - a = 5 => 5
            - a + 4 => 9
            - a = [1, 2, 3] => [1, 2, 3]

        - Show how everything in Ruby is a object/class, and as a result everything in ruby has methods
            - Integer Class
                - 1.class => Integer
                - 1.methods => [:-@, :**, :<=>, :upto, ….] #all integer methods
            - Boolean Classes
                - true.class => TrueClass
                - True.methods => [:===, :inspect, :to_s, ….]
                - false.class => FalseClass
                - false.methods => [:===, :inspect, :to_s, ….]
                - Mention everything except nil and false will evaluate to true, including 0
                - Can give example: if 0 then puts "hello" else puts "world" end => “hello”
            - String Class
                - “Cmsc330”.class => String
                - Mention every class has a .to_s / to string method (from Object class)
            - nil.class => NilClass
                - Mention that nil has no methods other than .to_s and .equals
                    - nil.to_s => “” #empty string
                - Mention that any operation other than .to_s with nil will evaluate to a ***NoMethodError*** for the nil class.
                    - nil + 1 => error

- String examples
  - Length
    - “Cmsc330”.length => 7
    - “Cmsc330”.size => 7
  - Concatenation
    - “Cmsc” + “330” => “Cmsc330”
    - “Cmsc330”.concat(“ rocks”)
    - s = “cmsc” => “cmsc”
      s += “330” => “cmsc330”
      s => “cmsc330”
  - Interpolation (Put expressions to be evaluated into string)
    - foo = 330 => 330
      bar = “cmsc#{foo}” => “cmsc330”
      bar2 = “cmsc#{foo + 21}” => “cmsc351”

- Array examples
  - Declare
    - arr = Array.new => []
    - arr = [] => []
    - arr = [3, 3, 0] => [3, 3, 0]
  - Accessing / Adding
    - arr[0] => 3
    - arr[5] => nil
    - arr[5] = 6 => [3, 3, 0, nil, nil, 6]
    - arr[-1] => 6
    - arr2 = [“hello”, “world”] => [“hello”, “world”]
    - arr + arr2 => [3, 3, 0, nil, nil, 6, “hello”, “world”]
      - Mention that Ruby arrays are not singly typed
  - Iteration
    - arr.each{|x| print x} => 3306
    - Can also use for loop
  - Stack / Queue
    - arr = [] (clean out array)
    - arr.push(“330”) => [“330”]
    - arr.push(“rocks”) => [“330”, “rocks”]
    - arr.pop => “rocks”
    - arr => [“330”]
    - arr.unshift(“I love”) => [“I love”, “330”]
    - arr.shift => “I love”
    - arr => [“330”]

- Hash examples
  - Declaration

        - foo = {} => {}
        - bar = Hash.new => {}
        - foo and bar are treated as the same object for the purposes of keys
        - temp = Hash.new(Array.new) => {} (Default value is an empty array)
            - temp[:foo] << 2 => [2]
            - temp[:bar] => [2] ([2] has become the default because we modified the same array)
    - Insertion and Retrieval
        - foo = {}
            - foo[:s] = “hello” => “hello”
            - foo[:s] => “hello”
            - foo.key(“hello”) => :s
            - foo.keys => [:s]
            - foo.values => [“hello”]
    - Check for existing keys/values
        - foo = {:s => “hello”}
            - foo.has_key?(:s) => true
            - if foo[:s] then true else false => true
            - foo.has_value?(“bye”) => false

- Mention that Ruby documentation has dozens of other methods for all these data structures