CLASS 2: INTRO TO RUBY CORE CLASSES

AGENDA

- Using Ruby Core Classes
  - Strings
  - Arrays
- Using Ruby Documentation

RUBY STRINGS

- First Ruby Surprise: Ruby strings are mutable!
  - str="Hello!"
  - str.tr("e", "i")
- Second Surprise: many Array and Enumerable methods work on strings too!
  - They can be thought of as arrays of characters.
  - str.chars
  - str.chars.each{|c| p c}
- Third Surprise: some arithmetic operators do interesting things to strings.
  - * repeats a string
    - "Ruby! " * 3
  - << appends to the end of a string
    - "Ruby" << " Rocks"

USEFUL STRING METHODS

- Useful methods:
  - upcase
  - downcase
  - capitalize
  - split
  - slice
  - capitalize
  - tr
  - reverse
  - chars

- **The list of string methods is here:** http://www.ruby-doc.org/core-2.1.2/String.html

RUBY STRINGS

- **Using the documentation (or just guess!), find at least one way to do the following:**
  - Find the length of a string
  - Concatenate (add) two or more strings together
  - Split a string into pieces
  - Change the case of a string – capitalize it, change it all to upper case or to lower case, swap the cases, etc.
  - Reverse a string
  - Turn the string into an array of characters
  - Stick a prefix or a suffix on a string
  - Report on one other interesting and useful string method.

STRING PRACTICE

Given a string, replace all the vowels in the string with ‘*’

Input: "Hello, World!"

Output: "H*ll*, W*rld!"

STRING INTERPOLATION

- Values of variables can be included in strings using #{}
  - Only works in strings delimited by double-quotes (")
- Example to try:
  - x = 42
  - puts "The answer to the ultimate question of life, the universe and everything is #{x}."

RUBY ARRAYS

- Ordered, integer-index collections of any object
- The first element of an array has an index of zero (0)
- A negative index starts at the end of the array
  - arr[-1] is the last element of the array.

CREATE A NEW ARRAY

- Declare empty array with
  - Array.new
  - []
- Declare an array literal:
  - a = [1,2,3,4]

BASIC METHODS

- Basic methods:
  - join
  - pop
  - push
  - +
  - -
  - empty?,
  - include?
  - first
  - last
  - map or collet
  - reverse
  - sort

MORE ON ARRAYS

- Accessing elements of an array. Given the array [1, 2, 3, 4, 5, 6, 7, 8]
  - Get the first element of the array
  - Get the last element of the array
  - Get the array [2,3,4,5]
- Find the number of elements in an array
- Determine if an array is empty
- See if an element is included in an array
- Add items to the end of an array.
- Add items to the beginning of an array
- Take items from the end of an array.
- Take items from the beginning of an array.
- Remove an item from inside an array.

MORE STUFF TO DO TO ARRAYS

- Remove duplicates from an array.
- Remove nil elements from an array.
- Given the array [2,4,6,8,10], produce the array [4,16,36,64,100]
- What does the flatten method do and when should it be used?
- What is the difference between each, each_index and each_with_index?
- List a couple of other methods you think will be useful in array handling.

PRACTICE #1

Given an array a where each element represents a string, return a new array with each of those elements appended with three dots (“…”).

Input: a = [“Mercury”, “Gemini”, “Apollo”]

Output: [“Mercury…”, “Gemini…”,”Apollo…”]

DESTRUCTIVE OPERATIONS

- Methods that end with ! (bang) usually change the array!
  - Many methods have a non-destructive and a destructive version.
    - reverse, reverse!
    - sort, sort!
    - collect, collect!
    - map, map!
  - The non-destructive version returns an array different from the original array.
  - The destructive version changes the array in place.

DESTRUCTIVE VS NON-DESTRUCTIVE EXAMPLE

- Example:
  a = %w{one two three}
  a.reverse
  puts a
  a.reverse!
  puts a

SUMMARY

- We have looked at 2 Ruby core classes: Strings and Arrays
- Ruby documentation has good examples of methods available for both Strings and Arrays
- Important String methods:
  - Change the case of a string
  - Replace characters in a string
  - Find characters in a string
- Important Array methods:
  - Manipulate elements in an array.
  - Add and remove elements from an array.
  - Provide information about an array

NEXT CLASS

- More Ruby Core Classes
  - Hash
  - Enumerable
- The inject method
- Ruby syntax for making decisions
- Ruby syntax for loops

HOMEWORK - STRINGS

- Given a string, return true if it is a palindrome, false otherwise.
  - Remove blank spaces and punctuation using tr.
  - Use the ternary operator. (condition ? true_value : false_value)

  - Example:
    Given str = "Able was I ere I saw Elba"
  - Expected result: true

HOMEWORK - ARRAYS

- Given an array of strings representing the first names of people in a class. Return an array with two elements: the first element is a new array with class members whose names begin a letter between “A” and “J”. The second element is a new array with those whose names begin with a letter between “K” and “Z” inclusive.

- Example
  a=%w{ Annette Laura Louise Elizabeth Jessica Cheryl Janice Rebecca Kay Shannon AnnMarie Hailey Stephanie }
- Result: [["Annette", "Elizabeth", "Jessica", "Cheryl", "Janice", "AnnMarie", "Hailey"], ["Laura", "Louise", "Rebecca", "Kay", "Shannon", "Stephanie"]]

APPENDIX

Juicy Ruby Tidbits

TIME HOMEWORK PROBLEM ENHANCED

- **Time.now**
  - **current time - 2014-07-24 15:30:55 -0400**
- **Time.now.strftime("%H %M %S")**
  - **Go to http://www.ruby-doc.org/core-2.1.2/Time.html#method-i-strftime for all the different ways of extracting data from Time.now**

```
seconds_in_day = 24 * 60**2
hr, min, sec = Time.now.strftime("%H %M %S").split(" ").map(&:to_i)
sec_since_midnight = hr * 60 ** 2 + min * 60 + sec
sec_until_midnight = seconds_in_day - sec_since_midnight
percent_of_day_gone = sec_since_midnight.to_f / seconds_in_day * 100

puts "There are #{seconds_in_day} seconds in 1 day."
puts "Seconds since midnight: #{sec_since_midnight}."
puts "Seconds left in the day: #{sec_until_midnight}."
puts "This day is #{percent_of_day_gone.round}% done."
```