

Assignment 2: Ruby

Assigned: Saturday, 29 January 2011

Due: 11:00 p.m., Thursday, 3 February 2011

Summary: In this assignment, you will further explore some of the key ideas from the reading on Ruby.

Purposes: To give you more experience with Ruby and its underlying concepts, particularly duck typing, mixins, and the use of hash tables and arrays.

Expected Time: Three to four hours.

Collaboration: I would prefer that you work on this assignment in groups of two or three. You may also work on this assignment alone or in groups of up to size four. You may consult with anyone you wish on this assignment (including other classmates), provided you cite your consultants.

Submitting: Upload a tarball to Pioneerweb. See further details below.

Warning: So that this assignment is a learning experience for everyone, I may spend class time publicly critiquing your work.

Assignment

1. Grep

Implement a procedure, `grep pattern file-or-files options`, that searches for *pattern* in the specified file or files using the given options.

You can assume that the pattern is a string.

If the second parameter is a string, you can assume that it's a single file name.

If the second parameter is an array, you can assume that it's an array of file names.

The optional parameters will be an array of constants. You should minimally support `:invert` (print lines not matching the pattern) and `:number` (print line numbers)

2. Cut

Implement a procedure, `cut columns file optional-delimiter`, that extracts columns from a file. (Similar to the Unix `cut` utility.) For example, `cut 3,2 students.txt` should print out columns 3 and 2 of `students.txt`. As is the case of the `cut` utility, the default column delimiter is a tab.

3. Comparators

Implement a module, `SimpleComparator`, that provides a comparison method (`<=>`) which converts the values to strings and uses string comparison.

When demonstrating this module, show how one can use it to sort an array of symbols.

4. Graphs

Consider the following class, intended to represent nodes in a graph.

```
class GraphNode
  def initialize(name, neighbors=[])
    @node_name = name
    @neighbors = neighbors
  end
end
```

Extend the class with the following methods:

- `print_dfs`, which prints all nodes reachable from a node, using a prefix, depth-first search of the graph. You should not print a node twice.
- `print_bfs`, which prints all nodes reachable from a node, using a prefix, breadth-first search of the graph. You should not print a node twice.
- `dump`, which dumps the node and all reachable nodes to files (one node per file).
- `self.load filename` (a class method), which loads a node and all nodes reachable from it.

5. Setters and Getters

Some of you have noted that the `attr_accessor` method seems to violate encapsulation as it gives direct access to the attributes of an object.

Write a mixin (module) that declares two methods, setter *extname, attribute* and getter *extname, attribute*. These methods should build an appropriate setter or getter methods to access the given attribute. For example, setter `:name, :node_name` should add a method something like

```
def set_name new_value
  @node_name = new_value
end
```

Similarly, the method getter `:children :neighbors` should add something like

```
def get_children
  @neighbors
end
```

Important Evaluation Criteria

I will look at the standard “C” criteria: Is your code **correct**? Is your code **concise**? (This second attribute is particularly appropriate in Ruby.) Is your code appropriately **commented**? Have you used any particularly **clever** techniques. How **comely** is your code?

Submitting Your Homework

Create a directory for this assignment of the form `hw02.names`. Within that directory, create subdirectories for each part of the assignment (e.g., `part1`, `part2`). Each subdirectory should contain:

- The code that you have written for that part of the assignment.
- A typescript or typescripts in which you demonstrate the utility of the code you have written.
- (Optional) Unit tests.
- (Optional) A README file with any additional info you want to provide.

Make a gzipped tarball of the assignment directory. (If you don’t know how to do that, let me know.) Upload it via the Assignment 2 link on Pioneerweb.