Introduction to Ruby

Why Ruby?
Ruby in CIS 170
installing Ruby on your computer
documentation
First session

Ruby

Ruby was defined in 1994 by Yukihiro Matsumoto
- studied computer science and programming languages at Tsukuba University
- had used Perl, Python, Java, and other languages but none was “just right”
- His goals:
  - design a scripting language similar to Perl
  - base it on important computer science concepts (functional programming, object oriented programming)
  - implement these ideas in an interactive system
- The main concept: programs operate on sets of objects
  - numbers, strings, files, menus, windows, web pages, ...
  - objects in programs: Simula-67
  - object-oriented programming: Smalltalk (ca. 1970)

Ruby Home Page

Ruby is open source software
- anyone can download it and install it on their system
- The official home page for the Ruby project is:
  (the “en” is for “English”)
- There are links for
  - downloads
  - documentation
  - mailing lists
    - mainly technical discussions, e.g. proposals for new extensions

In Ruby, a piece of data is an object
Ruby in CIS 170

- We will be using Ruby for several small projects this term.
- Documentation on Ruby is available via the class web site.
- For most projects you will not need this extra information.
- Use it as an additional resource if you are curious or if you get stuck.

Where to Run Ruby

- For CIS 170 labs you have a choice:
  - Run Ruby on your own computer.
  - Use one of the UO labs.
- Your own machine:
  - Linux: download from www.ruby-lang.org, follow the installation instructions.
  - Mac OS/X: Ruby is pre-installed on 10.4 and 10.5.
- A UO computer center instructional lab:
  - Klamath B13.
  - McKenzie 101.
  - See http://cc.uoregon.edu/campuslabs.html for locations and hours.

Interactive Programming

- In CIS 170 we will be using Ruby as an interactive programming language.
- An interactive system is basically a fancy “calculator”
  1. Start Ruby.
  2. Ruby prints a prompt to let you know it is ready.
  3. Type an expression.
  4. Ruby evaluates the expression and prints the result.
  5. Go back to step 2.
- This cycle is called a “read eval print” loop.

Running Ruby

- To use Ruby interactively, start a terminal window.
  - In Windows, run a program named “cmd.”
  - In Linux, start a new shell window.
  - In OS/X, run the application named Terminal from the Applications/Utilities folder.
Running Ruby (cont’d)

- The shell command “irb” starts an interactive Ruby session
- A session that uses a Ruby program (evolve.rb) stored in a file

Running Ruby (cont’d)

- When you run Ruby interactively type an expression and hit the return key
  - you can also hit delete and use other keys to edit the expression
  - hint: learn how to use “command history”
- In these slides, prompts and other strings printed by Ruby will be shown in red, and characters I typed will be shown in black:
  >> 2 + 2
  => 4

Get started on “Project 0” today!
Make sure you can start a terminal application, start Ruby, and evaluate an expression

Variables

- To motivate the next topic, suppose you want to compute the surface area of a counter top
  - you know the counter is a square with a triangular piece missing
  - you also know each side of the triangle is half as long as the side of the square
- Using some simple geometry will give you an equation for the area of the counter

\[ a = x^2 - \left( \left( \frac{x}{2} \right)^2 / 2 \right) \]

Variables (cont’d)

- Suppose you measure one edge of the square to get \( x = 109 \text{cm} \)
- Ruby can compute the area for you:
  >> 109*109 - ((109/2)*(109/2) / 2)
  => 10423
- You can simplify this a little bit by using the exponentiation operator
  - to compute \( x^2 \) type \( x**2 \)
  >> 109**2 - ((109/2)**2 / 2)
  => 10423

\[ a = x^2 - \left( \left( \frac{x}{2} \right)^2 / 2 \right) \]
Variables (cont’d)

- You can make the equation even simpler by using a variable
  - create a value named x, and then write the equation using x
    
    >>> x = 109
    => 109
    
    >>> x**2 - ((x/2)**2 / 2)
    => 10423

- The first expression I typed defines a variable named x, and the second uses the value of x to compute the area

- An expression of the form “x = ...” is known as an assignment statement
  - put the name of a variable on the left side of the assignment operator
  - put any expression on the right side
  - Ruby evaluates the expression and saves the result

Variables Can Be Updated

- You can change the value of a variable
  - simply assign it a new value in another assignment statement
    
    >>> x = 109
    => 109
    
    >>> x**2 - ((x/2)**2 / 2)
    => 10423
    
    >>> x = 111
    => 111
    
    >>> x**2 - ((x/2)**2 / 2)
    => 10809

Variable Names

- There are a few rules for defining names for variables:
  - names can be as long as you want
    
    >>> supercalifragilisticexpialadocious = 3.14
    => 3.14
    
    >>> supercalifragilisticexpialadocious * supercalifragilisticexpialadocious
    => 9.8596
  - case is important (a is not the same as A)
  - names must start with a letter, but can have digits or underscores
  - note: variable names in Ruby start with lower case letters

Variables Are Labels for Objects

- Think of the memory of your computer (RAM) as being one big “object storage unit”

- When Ruby evaluates an assignment statement, it creates a new variable if it does not already exist
  
    >>> x = 109
    => 109
    
    >>> y = x**2 - ((x/2)**2 / 2)
    => 10423

- Variables can refer to many different types of objects
  - we’ll see examples of strings and other more complex objects later
Summary

- Ruby is an interactive language
  - `irb` has a read-eval-print loop
  - Type an expression, Ruby evaluates it and prints the result
- Expressions are made up of
  - Constants (e.g. `26` or `3.14159`)
  - Operators (e.g. `+`, `-`, `/`, `**` for expressions involving numbers)
  - Variables (labels on objects)
- When you ask Ruby to evaluate an expression containing a variable, the current value of the variable is plugged into the expression
  - Ruby fetches the value from the "object store"
  - A variable's value can be changed