CLASS 1: RUBY FUNDAMENTALS

TRY RUBY!

AGENDA

- What is Ruby?
- Interactive Ruby (IRB) is your new best friend!
- Expressions and Operators
  - Arithmetic
  - Variables
  - Assignment
- Printing
- Executing Ruby Programs

RUBY IS A GREAT PROGRAMMING LANGUAGE!

- Created by Yukihiro Matsumoto – “Matz”
  - MINASWAN
- Open Source
- Intuitive
- Duck Typing
- Fun!

RUBY MAKES YOU HAPPY!

Yukihiro Matsumoto: “You want to enjoy life, don't you? If you get your job done quickly and your job is fun, that's good isn't it? That's the purpose of life, partly. Your life is better.

“I want to solve problems I meet in the daily life by using computers, so I need to write programs. By using Ruby, I want to concentrate the things I do, not the magical rules of the language, like starting with public void something something something to say, "print hello world." I just want to say, "print this!" I don't want all the surrounding magic keywords. I just want to concentrate on the task. That's the basic idea. So I have tried to make Ruby code concise and succinct.”

WHICH SHOWS THE INTENT CLEARLY?

```
Calendar cal =
    Calendar.getInstance();
cal.add(Calendar.DATE, -1);
System.out.println(cal.getTime()
    + "");
```

```
yesterday =1.day.ago
```

INTERACTIVE RUBY (IRB)

- irb is a Ruby Shell
- Type any Ruby expression and it will be evaluated
- A good way to try out new language features

```
$ irb --simple-prompt
>> 2**3
=> 8
>> "Ruby! " * 3
=> "Ruby! Ruby! Ruby! "
>> 1.upto(3){|x| puts x}
1
2
3
=> 1
>> quit
```

IRB

- A quick way to evaluate any Ruby expression
- Not just for beginners
  - Helps to understand what various classes and methods do

TRY THESE IN IRB

- 1 + 5
- 8 * 4
- 1 + 3 / 4
  - What???
- 1 + 3.0 / 4
- 2 ** 1024
- “Ruby” + “rocks!”
- “Ruby! “ * 3
  - Try this one: 3 * “Ruby! “
- 42 * 9 / 5 + 32
  - How can you make this more accurate?

EXPRESSIONS AND OPERATORS

- Arithmetic Operators
  - +, -, *, /, %, **
  - Follows Algebraic Order of Operations
    - parentheses, exponents, multiplication, division and mod in order from left to right, addition and subtraction in order from left to right
- Comparison Operators
  - ==, !=, <, > <=, >=, <=>
- Boolean Operators
  - &&, ||, !
- Ternary Operator
  - ? : (condition ? value_if_true : value_if_false)

FLOATS AND ARITHMETIC

- Float arithmetic can be surprising (just like C, Java and other computer languages)
- Try this in irb:
  - x = 9.0 / 10.0
  - y = 9.0 / 100.0 * 10.0
  - x ==y
  - False? Why?
  - Use instead: (x – y).abs < 0.001

VARIABLES

- **Make programs and methods more flexible.**
- **Assign names to storage locations in memory**
- **Ruby variable name requirements:**
  - Consist of letters, numbers and underscore characters
  - May not begin with a number
  - May not include whitespace or non-printing characters
  - A limited number of punctuation character may be included
    - $, @, @@, ?, !, =
    - More on these characters when we talk about objects and classes.
  - Start with a lower case letter, use snake case (Ruby idiom)
    - my_nice_variable
  - Can't use Ruby keywords

Using descriptive names makes your code self-documenting!

ASSIGNMENT

- **= is the Ruby assignment operator**
  - x = 3
  - str = “This is a test.”
- Assignment can be combined with other operators.
  - x += 3
  - y *= 6
- Parallel assignment
  - a,b = 1, 2
  - a, b = b, a #swaps the values
  - a,b,c = [1, 2, 3]

DISPLAYING OUTPUT

- puts or p
  - Displays a string of text to the console (including a newline).
  - Converts non-string objects to strings before printing.
- print
  - Same as puts or p, but without the newline

TRY THESE IN IRB:

```
9.downto(1) {|n| print n}
```

```
9.downto(1) {|n| puts n}
```

CREATING RUBY PROGRAMS

- Ruby programs consist of one or more Ruby statements
- The file extension is .rb
- To execute the file at the command line, type:
  - ruby program_name.rb

```
Place the following code in a text file:

9.downto(1) {|n| print n}
puts " blastoff!"

Save the file as blast.rb
Execute the file at a command prompt by
typing:

ruby blast.rb
```

CREATING AND EDITING A RUBY PROGRAM

- To get started editing a program, open your favorite text editor
  - TextMate
  - TextWrangler
  - Sublime Text 2
  - Notepad

```
9.downto(1) {|n| print n}
puts " blastoff!\n"
```

```
9.downto(1) {|n| puts n}
puts " blastoff!"
```

PROGRAM EXAMPLE WALK-THROUGH

- 1 gallon of Tip-Top paint will cover 400 square feet.
- The room to be painted is 12’x14’.
- The height of the room is 7.5’
- How many gallons of paint are needed?
  - Disregard doors and windows
  - Round up to a whole number of gallons.
  - Don’t paint the ceiling or the floor.
- Extra credit: the room has one door 3’x7’ and one window, 3’x4’.

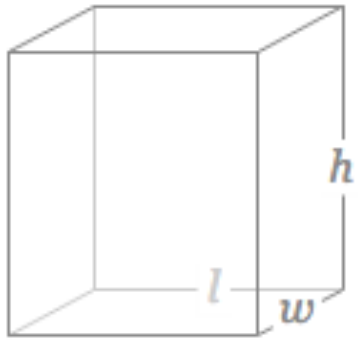
FORMULA

Right rectangular prism

Solve for surface area

$$A = 2(wl + hl + hw)$$

l	Length	Enter value
w	Width	Enter value
h	Height	Enter value



RUBY PROGRAM

- Create a new document using your favorite text editor.
- Write 3 statements to assign values to length, width and height.
- Write an expression which calculates the wall_area.
- Write an expression which calculates how many gallons are needed.
- Create a print or puts statement to print out how many gallons are needed.
- Save the program as paint.rb
- Run your program by typing ‘ruby paint.rb’

ENHANCEMENTS TO THE PAINT PROGRAM

- Go here: http://www.ruby-doc.org/core-2.1.2/Numeric.html and find a way to round up to the next whole number of gallons.
- Print out some text that explains what the number you have found represents. (Remember from 4th grade – always label your answers!)

USE THIS IF YOU ARE STUCK!

- Enter the following code:
  - length=14
  - width=12
  - height=7.5
  - wall_area = 2 * (height * length + height * width)
  - paint = wall_area / 400
  - gallons = paint.ceil
  - puts “You will need #{gallons} gallons of paint.”
- Save your code as paint.rb.
- Run your program by typing at the command line: ruby paint.rb

A BETTER OUTPUT STATEMENT

puts "You will need #{gallons} gallon" + (gallons > 1 ? "s" : "") + " of paint."

What does #{gallons} do?

Explain the expression: (gallons > 1 ? "s" : "")

SUMMARY

- Ruby is concise and succinct.
- irb helps you to learn Ruby.
- Ruby lets you do arithmetic, assignment and printing.
- You can collect your programming statements from irb into a document that can be run with Ruby.

NEXT CLASS

- We will look at Ruby core classes – strings and arrays– to explore what kinds of actions we can add to our Ruby vocabulary.

HOMEWORK

- Create a Ruby program to do the following:
  - Create variables named *hour*, *minute* and *second* and assign them values that are roughly the current time. Use a 24-hour clock, so that at 2 PM the value of *hour* is 14.
  - Calculate and assign the number of seconds since midnight.
  - Calculate and assign the number of seconds remaining in the day.
  - Calculate and assign the percentage of the day that has passed.
  - Print the calculated numbers.
    - Print anything else that will add to the readability of your result.
  - Change the values of hour, minute and second to reflect the current time. Check to make sure that your calculations are correct for the new values.

APPENDIX

Juicy Little Tips

LOG COMMANDS FROM IRB

In your home directory, create a file called .irbrc with the following contents:

```
require 'logger'

if ENV.include?('RAILS_ENV') && !Object.const_defined?
    ('RAILS_DEFAULT_LOGGER')

  Object.const_set('RAILS_DEFAULT_LOGGER', Logger.new(STDOUT))

end


require 'irb/ext/save-history'

IRB.conf[:SAVE_HISTORY] = 200

IRB.conf[:HISTORY_FILE] = "#{ENV['HOME']}/.irb-history"
```

Your home directory will now contain a file called .irb-history that contains your last 200 commands.