START UP
INSTITUTE
RAMPUP

INTRO TO RUBY - WEEK 1

WHAT IS RUBY?

“Ruby is a general-purpose, object-oriented, interpreted programming language designed and written by Yukihiro Matsumoto (known widely as “Matz”). It was introduced in 1994 and became popular in Japan during the 1990s. It’s known and admired for its expressiveness—its ability to do a lot with relatively little code—and the elegance and visual smoothness of its syntax and style. Ruby has proven useful and productive in a wide variety of programming contexts, ranging from administrative scripting to device embedding, from web development to PDF document processing.”

- David Black, *author of The Well Grounded Rubyist*

BASIC UNIX COMMANDS

ls - list your files

cd *foldername* - change directory

cp *filename* - copies a file

rm - i *filename* - delete file with prompt

rm -r *foldername* - deletes directory and **everything** in it

mkdir *foldername* - creates a new folder

mv *filename* - moves or rename file

touch *filename* - creates new file

BASIC RUBY DATA TYPES - STRINGS

Strings are instances of the String class. You can create strings like this:

1. String.new(‘hello’)
2. “hello”
3. ‘hello’

Since strings are instances of the String class. They have over 75 standards methods that you can take advantage of. You can refer to the Ruby Documentation to see the complete list provided here: http://ruby-doc.org/core-2.2.0/String.html

Let’s try a couple on the next slide!

BASIC RUBY DATA TYPES - STRINGS

Type **irb** into your console. **IRB** stands for Interactive Ruby shell which is a program that provides an environment where you can execute ruby commands with immediate response and allows experimenting in real time.
Now try this:

‘hello’.capitalize

‘shouting’.upcase

‘interactive ruby’.reverse

‘capitalize’.include?(‘a’)

Try a couple more on your own! Ruby Docs is your new best friend so take the time to get to know her/him better.

INTEGER (FIXNUM/BIGNUM) & FLOAT

Integers or whole numbers in ruby are instances of the Fixnum class. Unlike Strings, which can have many instances, there is exactly one instance of each number. Fixnum Ruby Docs: http://ruby-doc.org/core-2.2.0/Fixnum.html

1. 1
2. 1234
3. ‘3’ is not a number but a string
4. 12345678901234567890 is a Bignum
5. 123_456 underscores are ignored so it is the same as 123456

INTEGER (FIXNUM/BIGNUM) & FLOAT

Floating point numbers are instances of the Float class. They are inexact numbers based on double precision floating point representation. Float Ruby Docs Link: http://ruby-doc.org/core-2.2.0/Float.html

1. 1.0
2. 2.3898438298
3. 1231.481

ARITHMETIC OPERATORS

Ruby arithmetic operators allows you to perform basic math functions with numbers.

1. + : addition 10 + 10 returns 20
2. - : subtraction 20 - 10 returns 10
3. * : Multiplication 10 * 10 returns 100
4. / : Division 10 / 5 returns 2
5. % : Modulus 5 % 4 returns 1
6. ** : Exponent 2 ** 3 returns 8

Some can also manipulate strings. Try this in IRB:

1. ‘hello’ + ‘world’
2. ‘hello’ * 3

<u>Operator Precedence</u>

1. **
2. *
3. /
4. %
5. +
6. -

You can override precedence by using parentheses:

2 + 4 * 5 returns 22
(2+4) * 5 returns 30

BOOLEAN & NIL

In Ruby, the True and False booleans each have their own class. *True* is an instance of TrueClass and *False* is an instance of FalseClass.

Every Ruby expression evaluates to an object. Every Ruby object in turn can be used in a conditional statement which means every object ultimately must evaluate to a boolean value.

Most objects in Ruby will have a boolean value of ***true***. Only two objects have a boolean value of ***false***, they are ***false*** and ***nil***.

Nil is a special non-entity object it derives from NilClass. ***nil*** denotes the absence of a value, result or a state of being that is undetermined. Nil however does **not** mean it is empty.

Like Fixnum, true, false, and nil can only have one instance of itself.

LOCAL VARIABLES

Variables are a way of assigning a “label” that represents a value or a block of code. It provides context and allows you to reference it quick and easily. There are many types of variables. Local variables have limited scope which means it is only visible in a limited part of the program.

Valid local variable formats:

1. **x**
2. **_x**
3. **address**
4. **first_name**
5. **person1**
6. **phone_NUMBER**
7. **_**

When assigning values the local variable must be on the left hand side and the value on the right hand side.

valid:
name = “Johnny”

invalid:
“Johnny” = name

Local variables cannot start with a number and should not start with a capitalized letter. Conventional Ruby style prefers *under_score* names over *camelCase* names for local variables. Examples 1 - 4 are the most commonly used ways to name a local variable. Try it out in IRB. You can assign most things to a local variable.

RUBY ASSIGNMENT OPERATORS

Ruby uses assignment operators to assign the value to a variable.

1. = : Simple assignment. name = “John”
2. += : *a += 2* is a simplified version of *a = a + 2*
3. -= : *a -= 2* is a simplified version of *a = a - 2*
4. *= : *a *= 2* is the same as *a = a * 2*
5. /= : *a /= 2* is the same as a = a / 2
6. %= : *a %= 2* is the same as *a = a % 2*
7. **= : *a **= 2* is the same as *a = a ** 2*

Ruby also supports parallel assignment:

a, b, c = 2, 3 ,4 is the same as a = 2, b = 3, c = 4

IRB VS SUBLIME

As we've mentioned before IRB is an interactive shell that let's you run Ruby code in real time. Once you exit out of it all the code you've written are gone. Alternatively, you can write code in a text editor and save it with the .rb file extension. We will be using the Sublime 2 Text Editor.

In a file, your code is evaluated from top to bottom. Line 1 always runs before Line 2.

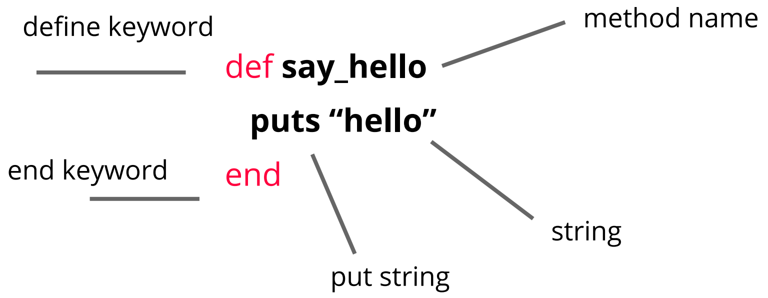
To execute the code in your file:

1. Make sure you save the file with the latest changes.
2. Run your file by typing ***ruby filename.rb***

Notice that executing a file does not output anything in the console. It simply runs, executes the code and exit out of it. It will only output something if there is an error or you explicitly tell it to with "puts" or "print."

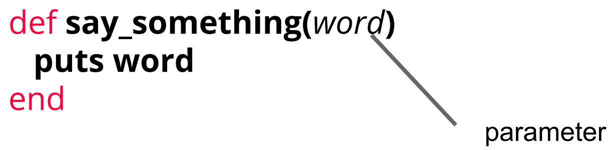
RUBY METHODS

Ruby methods are very similar to functions in any other programming language. Ruby methods are used to bundle one or more repeatable statements into a single unit. Method names should begin with a lowercase.



```
def say_hello
  puts "hello"
end
```

A method can have parameters:



```
def say_something(word)
  puts word
end
```

Methods can have multiple parameters.
The last value in a method is the return value.

PRINT & PUTS

The **puts**, short for "put string", and **print** commands are both used to display the results of evaluating Ruby code. The primary difference between the two is that puts adds a new line after executing while print does not.

```
name1 = “Joe”

name2 = “Dan”


puts name1

puts name2


print name1

print name2
```

You can manually add a new line by adding “*\n*” to your string.

```
name1 = “Joe \n”
name2 = “Dan \n”

print name1
print name2
```

will give you the same results as puts

ASKING FOR USER INPUT

There are times where user input is necessary for a program to run. Ruby has the **gets** method which prompts the user for input in the console. Try the following:

```
puts “What is your name?”
name = gets.chomp
puts name + “is your name”
```

gets automatically adds a newline to your string. We use the **chomp** method to remove that newline. Try the above without the **chomp** method.

gets will always return a string. If a user inputs a number you will need to convert it to a Fixnum. Thankfully Ruby provides a method that turns a string like “2” into a Fixnum by using the ***to_i*** method.

RUBY KEYWORDS

BEGIN	do	next	then
END	else	nil	true
alias	elsif	not	undef
and	end	or	unless
begin	ensure	redo	until
break	false	rescue	when
case	for	retry	while
class	if	return	while
def	in	self	__FILE__
defined?	module	super	__LINE__

This is the list of reserved words in Ruby that you should not use as constants or variables.

LABS

1. Create a method that takes a celsius argument and convert it from celsius to fahrenheit. Or vice versa

2. Create a method that asks for a user input and converts it to fahrenheit. Or vice versa.

3. Write a program that asks for a name and an age. Have it output something like “Bob was born in 1985”

4. Write a mad lib program. Users are prompted for verbs, nouns, and adjectives and then it should output a paragraph using those words.

Be prepared to present one of the above labs in class next week!

START UP INSTITUTE
RAMPUP

INTRO TO RUBY - WEEK 1