CSCI-2325
Object-Oriented Paradigm: Ruby
Mohammad T. Irfan

Imperative vs. object-oriented paradigms
Imperative vs. object-oriented

- Imperative
  - Procedural decomposition
  - Procedures are all powerful
  - Data is helpless, at the mercy of procedures

- Object-oriented (OO)
  - Data-centric: data governs the decomposition
  - Classes - templates/patterns, abstracts away data
  - Objects - actual things/instantiations of classes

- Advantages of OO paradigm
  - Collaboration
  - Debugging
  - Reuse

Roadmap

- Learn the basics of Ruby today
- Investigate Ruby’s object-oriented design principles
- Ruby on Rails for web programming
Ruby Installation

- Several ways to install, as described here: https://www.ruby-lang.org/en/downloads/
- Mac/Linux: Use RVM (https://rvm.io/rvm/install)
  - Command line$ curl -sSL https://get.rvm.io | bash -s stable -ruby
  - Then, follow the instruction given in terminal
  - To test, commands are:
    - ruby -v
    - rvm -v
  - If you get errors, run the following commands (assuming 2.1.4 is the latest version—look up rvm website for it)
    - brew update && brew upgrade
    - rvm reinstall 2.1.4 --disable-binary
- Windows: Install Ruby 2.0.0 from RubyInstaller.org http://rubyinstaller.org/downloads/
- Recommended IDE
  - Aptana Studio 3 http://www.aptana.com/

Ruby resources

- Learning
  - English translation of the creator’s user guide (by Mark Slagell)
    - http://www.rubyist.net/~slagell/ruby/index.html
  - Go to reference
    - Documentation: http://ruby-doc.org/core-2.0.0/
    - http://www.tutorialspoint.com/ruby/
  - Interactive tutorial using only your web-browser
    - http://tryruby.org
Origin

- Designed by Yukihiro Matsumoto (Matz) in early 1990s
- Inspired by Perl and Python
  - Less scripting than Perl
  - More object-oriented than Python
  - Happy experience!

Quotes

- **Bruce Stewart (2001):** Did you have a guiding philosophy when designing Ruby?
  - **Matz:** Yes, it’s called the “principle of least surprise.” I believe people want to express themselves when they program. They don’t want to fight with the language. Programming languages must feel natural to programmers. I tried to make people enjoy programming and concentrate on the fun and creative part of programming when they use Ruby. (http://www.linuxdevcenter.com/pub/a/linux/2001/11/29/ruby.html)

- **Bill Venners (2003):** In an introductory article on Ruby, you wrote, “For me the purpose of life is partly to have joy. Programmers often feel joy when they can concentrate on the creative side of programming. So Ruby is designed to make programmers happy.” How can Ruby make programmers happy?
  - **Matz:** 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. (http://www.artima.com/intv/ruby.html)
Interview of Matz

- [http://vimeo.com/52954702](http://vimeo.com/52954702)

Features

- **Purely** object oriented
  - Every data value is an object - no primitive type
  - Every subroutine is a method
  - Inheritance can be applied to any class
- Both classes and objects are **dynamic**!
  - Can add methods to classes and objects dynamically
  - Different objects of the same class can behave differently

- **Dynamically** typed
- **Static** scoping

- 37 reasons to love Ruby!
  - [http://rubyhacker.com/ruby37.html](http://rubyhacker.com/ruby37.html)

You should be able to explain these!
Let’s get our hands dirty

Before we start

- If you want to quickly check something without writing a program
  - Use the `irb` command in Terminal

- Examples
  - ```ruby
  x = 10
  if x % 2 == 0
    puts “Even”
  else
    puts “Odd”
  end
  ```

- What does nil mean in the output? In Ruby, there is no statement. Everything is an expression returning a value, whether you explicitly say return or not.

- ```ruby
  x = [“NFL”, “NBA”, 2014]
  x.class
  x.class.methods
  x.include? “NBA”
  x.include? “2010”
  ```
Variables

- Type is implicit
- Type can be changed dynamically
- Naming:

  | $  | Global variable |
  | @  | Instance variable |
  | [a-z] or _ | Local variable |
  | [A-Z] | Constant |

- Examples (in irb)
  - `x = 10.99`
    - `x.class` #prints Float
    - `x = “Hello Ruby!”`
      - `x.class` #prints String
  - Very rich `String` class
    - Examples: [http://ruby-doc.org/core-2.0.0/String.html](http://ruby-doc.org/core-2.0.0/String.html)

Arrays (mutable)

- Creation, insertion, deletion
  - `myArray = [“NFL”, “NBA”, 2013]`
  - `myString = myArray.join(“ ”) #outputs “NFL NBA 2013”`
  - `left = myArray.shift #left has value “NFL”`
  - `myArray #myArray is now [“NBA”, 2013]`
  - `myArray.push(“MLS”) #myArray is now [“NBA”, 2013, “MLS”]`
  - `myArray.unshift(“NFL”)`
    - `myArray is now [“NFL”, “NBA”, 2013, “ MLS”]`
  - `delete(obj), delete_at(index), delete_if { |item| block }`

- Accessing elements
  - `myArray[0] #“NFL”`
  - `myArray[0..1] #everything in the array`
  - `myArray.each { |item| print item, “-“} #iterate through items`
  - `myArray.each_index { |i| print i, “-“+ myArray[i], “\n”}`
Sample program: factorial

- Save it as source.rb

```ruby
def fact(n)
  if n == 0
    1
  else
    n * fact(n-1)
  end
end
```

- Ways to run
  - 1. Add this line at the end of source.rb and click on run
    - puts fact(10)
  - 2. ruby -l ./ -r source.rb -e “puts fact(10)”
  - Command line arguments are also supported

Problem: Collatz Conjecture

- From Wikipedia
- Take any **natural number** n. The conjecture is that no matter what n is, you will always eventually reach 1.
  - Take n as input from user.
- If n is even, divide it by 2 to get n/2. If n is odd, multiply it by 3 and add 1 to obtain 3n + 1. Repeat the process until you reach n = 1. *(conditional statements and loops)*
  - Print all these numbers to a file.
- The number of numbers is called the cycle length of n.
  - Output the cycle length (to standard output)
# of steps (y) vs. input number (x)

Solution: Collatz.rb

```ruby
begin #beginning multiline comment
  Verify Collatz conjecture for input number
  Also known as the 3n+1 problem
=end #ending comments

while true #Iterate until a valid input is given
  print "Input n > 0: " #semicolon not needed
  n = gets.chomp.to_i #get string, remove \n, convert to int
  break if n > 0 && n.is_a?(Integer) #no primitive type
end
  puts "Wrong input. Try again." #prints with \n
# Really, there's no primitive type
  puts "Your input #{n} belongs to #{n.class} class"

# Open a file for output
outFile = File.new("Collatz_#{n}.txt", "w")
  #Write to file
if outFile #Successfully opened file
  outFile.write("#{n}->") #prints without \nelse #Display a RuntimeError message and exit
  raise "File could not be opened"
end
```
cycle_length = 0
28 while (n != 1) # you may omit parenthesis
29   if n%2 == 0
30     n = n/2
31   else
32     n = 3*n + 1
33   end
34   outFile.write("#{n}"")
35   outFile.write("-> ") if n > 1
36   cycle_length += 1
37 end
38 puts "Cycle length = #{cycle_length}"
41 outFile.close # close the file

Review:
What’s new in Ruby? (vs. Java/C++)

► Purely object oriented
► Classes and objects are dynamic
► Class can be defined later, dynamically
Control structure

- Conditional
  - if - elsif - else - end
  - ---- if condition

- Iteration
  - Usual while loops
  - arrayName.each do |item|
    ...
    end
  - arrayName.each { |item| ... }
  - Other ways: for loop
    for i in 0..4
    ...
    end

Cool stuff:
Reading a website
More fun: Can we “crawl” the web?

1. Extract all links from a web page
2. Do recursion [Assignment—later]
Check out: rubular.com

```
load "open-uri.rb" #Standard library for URI

def getlinks(url, str)
  links = str.scan(<a\shref="(.+)"/>).flatten
  links.each { |oneLink|
    oneLink.insert(0, url) if oneLink.start_with?("/")
  }
end

url = "http://www.bowdoin.edu"
begin
  webPage = open(url).read
  # Error above? Termination model: control goes to rescue
  begin # multiline comment
    puts webPage.class # String
    puts "Here goes the lines from the web page"
    webPage.each_line do |line| # Iterator provided by String
      print line
    end
  end
  links = getlinks(url, webPage)
  puts links
  puts "# of links = #{links.size}"
rescue # Exception handler
  puts "There's an error:"
  puts "\#{$/}"
end
puts "\nEnd of my program"
```

“Gem” for crawling the web

- Uses another gem called nokogiri for parsing web pages
- Command line: $ gem install anemone
- Ruby Code:

```ruby
require 'anemone'
Anemone.crawl("http://www.Bowdoin.edu/") do |anemone|
  anemone.on_every_page do |page|
    puts page.url
  end
end
```
Object-oriented features

Open class

- Can add a method to an existing class

```ruby
class Array
  def summarize
    self.each do |x|
      print x, " 
    end #iterator
    print "\n"
  end #def
end #class
```
Bill Venners: In Ruby, I can add methods and variables to objects at runtime. ... But in Java, for example, once a class is loaded or an object is instantiated, its interface stays the same. Allowing the interface to change at runtime seems a bit scary to me. ... What's the benefit of being able to add methods at runtime?

Yukihiro Matsumoto: First of all, you don't have to use that feature. The most useful application of dynamic features, such as adding methods to objects, is meta-programming. Such features allow you to create a library that adapts to the environment, but they are not for casual uses.
Classes in Ruby - the usual stuff

```ruby
class Website
  @@dummy # Class variable -- visible to subclass
  def initialize(url)
    @url = url # @url: instance var invisible to subclass
    @broken = false
    # Use () to extract the matched part into $1
    if url =~ /(\d+)/ || url =~ /(\d+)/
      @owner = $1
    end
  end
  # Can replace attr_reader and attr_writer
  # for @broken by attr_accessor
  attr_reader :url, :broken, :owner
  attr_writer :broken # don't want @url, @owner to change
end

w = Website.new("http://www.bowdoin.edu/~someone/")
puts w.url
w.broken = true # this will work
print "Is it broken? ", w.broken, "\n"
print "Owner is: ", w.owner, "\n"
# w.url = "it will not work"

http://www.bowdoin.edu/~someone/
Is it broken? true
Owner is: someone
```
Classes in Ruby: surprise!

Yes, classes are objects of Class

What does it mean?

- We can create classes dynamically (just like other objects)

Website .rb
Modify a class dynamically

- Modify the Website class dynamically

Website.rb
(After the previous code that defines the Website class)

```ruby
class Website
  # add an instance var @is_pdf with reader & writer
  attr_accessor :is_pdf # is it a pdf file?

  # also add a new method
  def matchPdf
    if url =~ /\.(pdf)$/
      @is_pdf = true
    else
      @is_pdf = false
    end
  end

  w.matchPdf
  puts w.is_pdf
end
```

Modify a specific object dynamically!
(Not the whole class)

- Singleton method

```ruby
w2 = Website.new("http://www.bowdoin.edu/president");

def w2.president
  @president_name = "Barry Mills"
end

puts w2.president #=> Barry Mills
#puts w.president #ERROR -- president is specific to w2
```
Singleton Design Pattern

- Use predefined Singleton module
  - A module is a collection of methods, constants
  - Unlike a class, modules cannot be instantiated
  - Example: Math

```ruby
class SingletonClass
  include Singleton #include module
  # ...
end

a = SingletonClass.instance
b = SingletonClass.instance
a == b #=> true

c = SingletonClass.new #=> NoMethodError
  # new is private
```

Inheritance: the usual stuff

```ruby
class PersonalWebsite < Website
  def initialize(url)
    @url = url #@url: instance var invisible to subclass
    @broken = false
  end

  attr_reader :url, :broken
  attr_writer :broken
end

#Save the owner's ID
if url =~ /~(.+)\// || url =~ /~(.+)$/
  @owner = $1
end

attr_reader :owner
end
```
No multiple inheritance

- Matz: Single inheritance is good because the whole class inheritance structure forms a single tree with a single root, named Object, and that is very easy to understand. In languages with multiple inheritance, the classes form a network, which is harder to understand.
Inheritance: cool stuff!

- **Mix-in**: multiple inheritance in some sense
  - Share the *behavior, not data*
- Building block: **module**
  - Collection of methods and constants

Mix-in example

```ruby
module A
  PI = 3.14
  E = 2.718
  def printPI
    puts PI
  end
  def printE
    puts E
  end
end

module B
  PI = 3.14159
  def printPI
    puts PI
  end
end

class C #mix-in class
  include A
  include B
  end
end

c = C.new

c.printPI #=> 3.14159

c.printE #=> 2.718
```

Collision!
Matz on Mix-ins

- Matz: “[…] approach of plugging in modules is called Mix-ins. Mix-ins originally started in LISP culture as a usage of multiple inheritance. In fact, a mix-in is actually a strict way of using multiple inheritance. So in LISP, it's a style of using multiple inheritance. In Ruby, we force you to use mix-ins by supporting classes and modules.”

Mix-in: more cool stuff!

- Modules do not have states - why?
- But… it can fake it!
  - Example on the next slide: Personal website subclasses Website and includes a module called PersonalInformation
```ruby
class Website
  def initialize(url)
    @url = url # @url instance var invisible to subclass
    @broken = false
  end
  attr_reader :url, :broken
  attr_writer :broken
end

module PersonalInformation
  @EMERGENCY_PHONE = "207-725-3500"
  # @email: instance var in includer.
  # Initialization will not work- why?
  @email = "no-effect@bowdoin.edu"
  attr_accessor :email
  def sendEmail
    puts "Sending an email to " + @email + "."
    print "Are you sure? (Y/N) "
    if gets.chomp == "Y"
      # now send the email
      puts "Email sent!"
    end
  end
end

class PersonalWebsite < Website
  include PersonalInformation # include module
  def initialize(url)
    super(url) # Call the super class constructor
    # Save the owner's ID
    if url =~ ~(.+)// || url =~ ~(.+)$
      @owner = $1
    # set the email address
    @email = "#{@owner}@bowdoin.edu"
    end
  end
  attr_reader :owner
end

w = PersonalWebsite.new(
  "http://www.bowdoin.edu/~somal"
) puts w.owner #=> somal w.sendEmail #=> Sending an email to somal@bowdoin.edu.
  #=> Are you sure? (Y/N)
```
Class Participation HW

- Create a Twitter developer account and use the Twitter API to search Twitter.
- Submit the print-out of search results (not code—contains personal information).
- Due: Next class (Thursday, Nov 13)

Assignment (Due on Wed, Nov 19)

- Crawl the web beginning at www.bowdoin.edu in an object-oriented way and detect broken links
  - Define necessary classes (and modules if needed)
  - Recursion is the key
    - Example: www.bowdoin.edu has a link to www.bowdoin.edu/computer-science, which has a broken link www.bowdoin.edu/~who. To detect the broken link, you will have to recursively fetch web pages and check links therein.
  - Confine your program to the Bowdoin domain.
  - Caution: it will take a really long time. So, do not re-check the same link.
Ruby Gems
Twitter
SQLite Database

Installation

First, install gems package management system
   - Extract it
   - Go into the extracted folder where you can see the "setup.rb" file
   - Execute this command:
     - ruby setup.rb
2. Update gem after installation
   - Execute this command: gem update --system
Installing new gems

- Command
  - `gem install twitter` [Twitter API]
  - `gem install mail` [Email API]
  - Other gems we will need later:
    - `gem install sqlite3` [SQLite DBMS]
    - `gem install rails` [Ruby on Rails]

- Useful commands:
  - `gem uninstall ...`, `gem list ...`, etc.

Working with the Twitter gem

- Examples
  - [http://sferik.github.io/twitter/](http://sferik.github.io/twitter/)

- Full documentation
  - [http://rdoc.info/gems/twitter/index](http://rdoc.info/gems/twitter/index)
Working with the Twitter gem

- Preparation
  - Sign up for Twitter
  - Sign in with your Twitter account at developer site
    - [https://dev.twitter.com/apps](https://dev.twitter.com/apps)
require 'twitter'
#Following line is for Windows only
OpenSSL::SSL::VERIFY_PEER = OpenSSL::SSL::VERIFY_NONE

client = Twitter::REST::Client.new do |config|
  config.consumer_key = "9jal6o14PuJv...hF5l0vj..."n
  config.consumer_secret = "ZnWxFp...9Aj6014P"n
  config.access_token = "28276124...7p6n"n
  config.access_token_secret = "...7P6N...0vj..."n
end

client.update("Ruby is fun!") #Posts a new tweet
puts "\nMy Friends: ":n
client.friends("mtirfan13").each { |f| puts f.name }n

#tweets is an array of Twitter::Tweet objects
tweets = client.search("#XboxOne #PS4 -r", lang: "en").take(10)
tweets.each do |tweet|
  puts "#{tweet.user.screen_name} -> #{tweet.text}"n
end
Common problem

- Rate limit exceeded!
- Twitter’s rules:
  - [https://dev.twitter.com/rest/public/rate-limiting](https://dev.twitter.com/rest/public/rate-limiting)

Store search results in database

```ruby
require 'sqlite3'

# Create DB instance
twitterDB = SQLite3::Database.new("My Twitter Database")
# Create a new "Table" if it does not already exist
twitterDB.execute("create table if not exists twitterTable
(id INTEGER PRIMARY KEY, name TEXT, status TEXT);")
for tweets.each_index do |i|
  rows = twitterDB.execute("select status from twitterTable where status = ?", tweets[i].text)
  if rows.length == 0 # This is a new status -- add it to the table
    twitterDB.execute("insert into twitterTable
      (name, status) values (?, ?)",
      tweets[i].user.screen_name, tweets[i].text )
  end
end

rows = twitterDB.execute("select name, status from twitterTable")
puts "Found #{rows.length} rows:"
for row in rows.each do |row| # Each row is an array of two strings: name and status
  puts row.join("\n")
end
```
Coming up next

- Ruby on Rails
  - Model
    - DB and constraints on data
  - View
    - What users see
  - Controller
    - Takes user input
    - Consults with model
    - Directs the view