G. The main topics of Ruby to study are:

(i) basics

•Created in 1993

•Object-oriented

•Dynamically typed

•Scripting Language

(a) Array and Hash

a = [’human’, ‘tiger’, ‘dog’] #array

a[0] # gives us ‘human’

a[2] # gives us ‘dog’

sound = { ‘human’ => ‘talk’, ‘dog’ => ‘bark, ‘tiger’ => ‘roar’} #hash

p sound[‘dog’] # gives us ‘bark’

(b) Code blocks

•A code block is a function without a name

•You can pass it as a parameter to another function

•Code block is basically a chunk of code between braces or between do and end.

(c) Higher order functions like

Map

• collect takes each element from the collection and passes it to the block.

The results returned by the block are used to construct a new array.

For instance:

["H", "A", "L"].collect { |x| x.succ }

•Alternative to collect is map

["a", "b", "c"].map{|x| x.succ}

• my\_array = [4, 2, 7]

my\_array.map!{|x| x\*3}

Select

Inject

my\_array = [2, 3, 5]

result = my\_array.injectdo |r , element|

r \* element

end

p result

Find

(d) regular expression

line = gets

x = /hardware|software/

if line =~ x

puts ”line includes pattern #{x}”

end

(e) Range

•(0..5) is an example of Range. See code below.

for i in 0..5

puts "Value of local variable is #{i}"

end

(0..5).each do |i|

puts "Value of local variable is #{i}"

end

(f) inheritance and polymorphism

class KaraokeSong < Song

def initialize(name, artist, duration, lyrics)

super(name, artist, duration)

@lyrics = lyrics

end

def to\_s

super + " [#{@lyrics}]"

end

end

aSong = KaraokeSong.new("My Way", "Sinatra", 225, "And now, the...")

aSong.to\_s

(g) use of module (mixing)

(h) meta-programming (open class and method\_missing). Basically, make sure you understand items in ppt1, ppt2 and ppt3 (except Slide 12 to 16) and related part in the textbook.

H. The main topics of Prolog to study are:

(i) basics

Prolog is a Declarative Language

* You will throw some facts and inferences and Prolog will do reasoning for you
* Prolog finds answer to the problem based on the facts and rules you provided
* but you do not need to know how!!!

(a) How to write a recursive rule like count that counts number of items in a list, and know how to write the ancestor rule given a family

-To compute count of items in a list

count(0, []).

count(C, [Head|Tail]) :-

count(TailC, Tail), C is TailC+ 1.

-To compute Sum of a List

sum(0, []).

sum(Total, [Head|Tail]) :-

sum(TailTotal, Tail), Total is Head + TailTotal.

-To compute Average

average(Average, List) :-

sum(Sum, List), count(Count, List),

Average is Sum/Count.

-Fibonacci

fib(0,1).

fib(1,1).

fib(N,F) :- N > 1, N1 is N - 1, N2 is N - 2,

fib(N1,F1), fib(N2,F2), F is F1 + F2.

(b) atom, subgoal, facts, rule, query, unification.

•Facts

* Facts are direct observations of our world.

likes(wallace, cheese).

likes(grommit, cheese).

likes(wendolene, sheep).

• Rules

* Rules are logical inferences about our world.

friend(X, Y) :likes(X, Z), likes(Y, Z), \+(X=Y).

–X is a friend of Y if X and Y are not the same and X and Y like the same Z.

• Atom vs. Variable

* In Prolog, the case of the first letter of a word is significant
* If a word begins with a lowercase character (or in quote), it’s an *atom*—a fixed value like a Ruby symbol.
* If it begins with an uppercase letter or an underscore, it’s a *variable*.

•Unification

means “Find the values that make both sides match.”

Example follows.

cat(lion).

cat(tiger).

twin\_cats(X, Y) :- cat(X), cat(Y).

| ?- twin\_cats(One, Two).

One = lion

Two = lion ?

(c) prolog built-in rules like min, append, member, etc.

(d) how to read a file and how to write to a file. Basically, make sure that you fully understand items in lecture ppt1 and ppt2 and related part in the textbook. You can ignore the advanced stuff (Slide 2 to 11) in ppt3.