RUBY LANGUAGE

Chen Wang
wangc1@email.arizona.edu

Xiaomin Zhao
xiaominzhao@email.arizona.edu

April 30, 2020

ABSTRACT

Ruby is a simple and easy language for coding beginner. It is an object-oriented language that is designed by Matsumoto in the mid-1900s. The language was build based on python and Perl. Ruby has similar control structure as Java or C has, at the same time it also has it's own unique block style, and various data types in Ruby.

1 Introduction

Ruby is an object-oriented language, it treated everything as an object, including classes and other types designated as primitives in other languages(such as integers, booleans, and "nulls"). This language mix the designer Matsumoto's favorite multiple languages (Perl, Smalltalk, Eiffel, Ada and Lisp), created a new language with both functional and imperative programming features. Ruby is a fairly flexible language that allows users to change themselves. The core part of Ruby can be changed or redefined. In addition, new functions can be added on top of existing functions. Ruby does not want to hinder the creativity of programmers.

2 History

Matsumoto was the designer of Ruby, one was finding an good object-oriented language. But She did not like either Python nor Perl. The first version of Ruby was released on December 21, 1995. Since its public release in 1995, Ruby has attracted many loyal programmers worldwide. In 2006, Ruby was widely accepted, and major cities have active user groups and Ruby-related developer conferences. Ruby is still alive and rising.

3 Control Structure

In Ruby language, the control structure is familiar to Java, C, and Perl programmer. Ruby has all common control structures. Such as If-else statement and while loops, as you will see in the examples, "" "" and "(" ")" are not necessary in Ruby. However, Java, C, Perl programmer might used to the syntax of Ruby. Since Ruby does not use parentheses, and Ruby uses the key word "end" to end the control block.

```
puts "If-else statement, grade calculator."
grade = gets.chomp.to_i
if grade < 60
  puts "fail"
else
  puts "grade A"
end
```

Similarly, while loop also uses "end" to stop the control block.

```
puts "please give a number from 1 to 10"
number = gets.chomp.to_i
i = 0
while i < number
  print "count " , i
  i += 1
end
```

The statement modifiers of Ruby is an useful tool.
For example:

```
if radiation > 3000
  puts "Danger, Will Robinson"
end
```

Using statement modifier, we have:
puts "Danger, Will Robinson" if radiation > 3000

4 Data Type

4.1 Number Types

There are different kinds of number used in Ruby like integers and floating points. Ruby can handle both integer type and float type. When floating point numbers evaluate with integers, the output of the evaluation will be floating number by default.
Example:
n = 9.88 / 3

The value of n is 3.29333.

4.2 Boolean

Boolean in Ruby is one bit. It can either be true or false. In addition, nil and 0 can also represent true.
Example:

```
if true
  puts "true "
if nil
  puts "nil is true "
if 0
  puts "0 is true"
end
```

output:
true true true

4.3 String

String is a sequence of characters that made up with letters or numbers. Strings are defined by enclosing a text within double or single quotes. In one string you can use both double and single quotes. In Ruby the '+' operator cannot be

used with strings, however you can use '*' for duplication.
Example:
puts "You are right."
puts 'She said "okay!"'
puts 'It\'s mine.'

output:
You are right.
She said "okay!"
It's mine.

4.4 Array

Array is used to store data, one array can contain different types of data. In the array, data are separated by commas in between each other. The starting index of the array is 0.

Example:

```
array = [1, "two", "three"]

array.each do |i|
  puts i
end
```

output:
1
two
three

5 Subprogram

In ruby, subprograms are call functions and classes. The functions or classes only executed when the subprogram is called or invoked.

5.1 Functions

To define a function:
def function()

end

def means to define or start a function; it is follow by the function name. A subprogram may by called with arguments. Which are the values passed from the calling program. If there are more than one argument, use commas to separate.
Example:
def function(argument1, argument2)

end

Ruby uses "call by value" convention for passing arguments, the copy of value is passed to the subprogram. And the subprogram may return a value to the caller.
Example:

```
first = "old"

def change(first)
  first = "new"
  return first;
```

```
end

puts first;
```

output:
old

Inside a function, we can call another function.
Example:

```
def function1()
  function2()
end

def function2()
  puts "in function1(function 2)"
end
```

We can also make an recursive call by calling the function itself.
Example:

```
n = 6;

def Fib(n)
  if(n <= 1)
    return n
  end
  return Fib(n-1) + Fib(n-2)
end
```

The "end" needed to be place at the end of functions.

5.2 Classes

Another subprogram is classes, in object oriented language, classes are also objects. Inside a class, we can define method.

Example this is a parent class:

```
class Parent

def parent()
  puts "Parent super"
  end

def override()
  puts "parent."
end
```

If we want to build a class inherited from the parent class, we can use "<" to do something like this:

```
class Child < Parent

def child()
  puts "child class"
  end

def override()
  puts "Child."
end
```

end

There is also an interesting top-level method in Ruby, continue using the example code above, we can print the string "Childs" by one line:
p Toplevel.new.send :example

6 Meta-programming

Is meta-programming something you have heard before? Meta-programming might be something only Ruby can do. In a couple word, Ruby meta-programming is use code to write code. It is an approach that you can write codes that can write code by itself dynamically at run-time. This means you can change or define a method or a class at run-time; which might be dangerous at some point of views.

```
class Developer

  def method_missing method, *args, &block
    return super method, *args, &block unless method.to_s =~ /^coding_\w+/
    self.class.send(:define_method, method) do
      p "writing " + method.to_s.gsub(/^coding_/, '').to_s
    end
    self.send method, *args, &block
  end

end

developer = Developer.new

developer.coding_frontend
developer.coding_backend
developer.coding_debug
```

Example by [5] Todorovic,Nikola

This example is taken from Nikola Todorovic's article "Ruby Metaprogramming Is Even Cooler Than It Sounds" [5]. The picture above is an example of meta-class. When coding_frontend is called, method_missing will be activated. Since coding_frontend match the specification of method name starting with "coding_", a method coding_frontend will be created using the method define_method.

7 Summary

The best strengths about Ruby are it's flexibility and elegant syntax. At the same time, Ruby is a fantastic scripting language that can accomplish many tasks brilliantly. Many syntactic sugar in Ruby can greatly improve production efficiency, and various libraries and gems (Ruby packages) can meet most daily needs. Ruby's pure object orientation allows you to handle objects in a consistent manner. The duck type implements a more realistic polymorphic design based on the methods that the object can provide, rather than the inheritance hierarchy of the object. Ruby's modules and open classes enable programmers to closely integrate behavior into syntax, greatly surpassing traditional methods and instance variables defined in classes.

References

[1] Jay Godse. *Ruby Data Processing: Using Map, Reduce, and Select*. Apress, 2018.

[2] Yukihiro Matsumoto. Ruby language reference manual, version 1.4. february 1998. *DOI= http://www. cs. auckland. ac. nz/references/ruby/doc_bundle/Manual/man-1.4/index. html*, 1998.

[3] YUKIHIRO MATSUMOTO, D Thomas, and A Hunt. Programming ruby, the pragmatic programmer's guide, 2001.

[4] Zed A Shaw. *Learn Ruby the hard way: a simple and idiomatic introduction to the imaginative world of computational thinking with code*. Addison-Wesley Professional, 2014.

[5] Nikola Todorovic. Ruby metaprogramming is even cooler than it sounds, Oct 2015.

[6] Tutorialspoint. *Ruby Tutorial Simply Easy Learning*. Tutorialspoint, 2017.

[1] [4] [3] [2] [6] [5]