Ruby I

CSc 372: Comparative Programming Languages

Installing Ruby

- http://www. ruby-lang.org/en/downloads/
- Or just use lectura as it’s already installed there. :-)

The “Mary Poppins” of programming languages [1]**...**

- efficiency through fun
- a spoonful of (syntactic) sugar
- focuses on the efficiency of the programmer rather than the efficiency of the language
- Ruby “stays out of your way” [2]
- Follows the “Principle of Least Surprise” [2]
- a “transparent” language [2]

About Ruby

- Creator: Yukihiro Matsumoto (aka Matz) (1993)
- interpreted, purely object-oriented, dynamically typed
- from a family of scripting languages
- OO: supports encapsulation and inheritance
- uses “duck” typing
- became popular in 2006
- https://ruby-doc.org

More about Ruby

- not very efficient in execution speed, but tends to make programmers more productive
- *syntactic sugar:* a language feature that makes code easier to read and write, even though there are other ways to do the same thing within the language

Details of Ruby

- interpreted
- no declared variables
- everything returns a value
- *pure* object oriented (everything really is an object!)

Tutorial: The Basics

puts, print, and p

- `puts`: prints to output with a newline at the end
- `print`: prints to output without a newline
- `p`: prints the result of `<object>.inspect`, which may be especially useful for debugging

Functions

Function Definition:

```
def <name>(<args>)
    <body>
end
```

Functions

Function Call:

- `<name>(<args>)`
- `<name> <args>`

Variables & Constants

- Local variables begin with lower-case letter or _.
  - Scope: inside the block
- Instance variables begin with @.
  - Scope: inside the object
- Class variables begin with @@.
  - Scope: all instances of the class
- Global variables begin with $.
  - Scope: pretty much everywhere
- Constants are capitalized.

Class Definitions

```
class <className>
     <body>
end
```

Class Access

- create new instance with `<className>.new(<args>)`
- call methods with `<var>.<method>(<args>)`

Operators

- +, -, *, /, %, **
- ==, !=, >, <, >=, <=, <=>, ===, .eql?, equal?

Equality

- `==` works like `equals` in Java
- `<=>` works like `compareTo` in Java
- `===` is used in case statements
- `.eql?` tests if items are the same type and the same value
- `equal?` tests if the items are aliases

Comments

```
# Comment

=begin

Comment

Comment

=end
```

Conditionals

- `if…then…`
- `if…elsif…else`
- `<code> if <cond>`
- `unless…`
- `unless…else`
- `case…when`

Loops

- while...end
- while...do...end
- ...while
- begin...end while
- until...end
- until...do...end
- ...until
- begin...end until
- for <var> in...end
- each...do

- `break` breaks out of a loop
- `next` jumps to the next iteration in a loop (like continue in Java)

Miscellaneous

- `class` returns an object's type
- multiple variables can be assigned in one statement
- Strings: “” vs. ‘’ – string evaluation vs. string literal

References

- [1] *Seven Languages in Seven Weeks: A Pragmatic Guide to Learning Programming Languages*, Bruce A. Tate
- [2] http://docs.ruby-doc.com/docs/ProgrammingRuby/