Overview of ESP's Ruby Scripting Language

5/7/22 Brent Roman brent@mbari.org

Why Choose Ruby?

- English-like, conversational syntax
  - Built-in REPL [Read-Evaluate-Print Loop]
- Minimal required punctuation
  - `SE.up 6` vs. `SE.up(6);`
- Everything is an “object”
  - Each object can react differently to the same command verb.
  - But, does not preclude top-level procedures
- Supports introspection and dynamic programming
- Easy to interface with ‘C’ code and Unix tools

It's Ruby all the way down

- Commands, Missions, Scripts, Protocols, Configuration Files
  - All are written in version 1.8 of the Ruby scripting language
- *Learn a little Ruby*
  - Rote memorization fails when something goes wrong
  - Standard on Mac OS, easily installed everywhere else.
- A gentle tutorial:
  - https://pine.fm/LearnToProgram/
- The bible:
  - https://ruby-doc.com/docs/ProgrammingRuby
- More (free) choices to suit your learning style:
  - http://ruby.about.com/od/tutorialsontheweb/tp/10waysfree.htm

Ruby Literals

- `1.is_a?  Integer`
- `1.0.is_a?  Float`
- `“Ruby”.is_a? String`
- `‘Ruby’.is_a? String`
- `:Ruby.is_a? Symbol`
- `[1, 1.0].is_a?  Array`
- `(1.0..3.14).is_a?  Range`
- `{red: 0, blue:-19, green:42}.is_a? Hash`
- `true.is_a? TrueClass`
- `false.is_a? FalseClass`
- `nil.is_a?  NilClass`

Ruby Symbols vs Strings

- Symbols literals start with colon (:symbol)
  - `:Brent` or `:"Brent Roman"`
- Strings are mutable (changeable)

```
'dog' + "s" #"dogs"
```

- Symbols are immutable (unchangeable)

```
:dog + :s
```

```
NoMethodError in simfast -- undefined method `+'
for :dog:Symbol
```

- Comparing Symbols is faster than Strings

Ruby Hash Objects

- Ruby hashes merely map arbitrary keys to corresponding values
  - Hash Examples:
    - { :a => 4, :b => 3.14, :foo => “foobar” }
    - {0 => 1, 1 => 0}
  - Note that keys need not be Symbols, but they usually are.
  - Curly braces must be omitted when passing Hash literal to methods
    - Otherwise, it would look like passing in a block of code!!
      - p {:a => 4, :b => 3.14, :foo => “foobar” } #INVALID

        vs.
      - *p :a => 4, :b => 3.14, :foo => “foobar”*

        alternatively
      - *p a: 4, b: 3.14, foo: “foobar”*
  - Undefined hash keys *usually* return nil
    - { :a => 4}[:b] == nil

Ruby Constants

- Constant Identifiers (aka Symbols)
  - begin with an UPPER case letter

```
MyPI = 3.14159
```

  - may be nested inside of a “module” or “class”

```
module Math; MyPI = 3.14159; end
Math::MyPI    # :: is like / in file paths
```

- Modules and Classes are themselves constants
  - declared by prefixing with ‘module’ or ‘class’ as above
- Special syntax needed to change once defined

```
Math.adjust :MyPI, 3.1415927
```

- Have global visibility (unlike local variables)

Ruby Global Variables

- Global Variable Identifiers
  - begin with '$' character

```
$myPi = 3.14159
```

- May be changed without special syntax
- Are globally visible
  - are ***not*** defined in the context of a module or class
- Should be avoided
  - better to use local variables whenever practical

Ruby Local Variables

- Local Variable Identifiers
  - begin with lower case character

```
myPi = 3.14159
```

- May be changed without special syntax

```
myPi = 3.1415927
```

- Visible within the method in which they are defined
  - methods are defined in modules or classes
- Syntax for calling a method with no args is same
  - as that for reading a local variable!
  - Local variables can shadow local methods
    - if the method has no arguments

Ruby Methods

- Define function that returns a single object
  - That returned object may be arbitrarily complex
- Associated with a module, class, or object
- Methods followed by an optional argument list
  - each argument is separated by a comma
  - required arguments first in argument list
    - followed by optional ones
    - followed (optionally) by a code block
- When proceeded with module, class, or object
  - calls method defined in that module, class, or object

`ESP.configure` **vs** `SE.configure`

MBARI's Custom Ruby Interpreter

- Forked from Ruby 1.87 in 2011
  - Due to numerous bugs crash bugs in the official version
  - https://sites.google.com/site/brentsrubypatches/brents-patches-to-the-ruby-intepreter
  - MBARI Ruby was offered by third parties as commercial product
    - until about 2014
  - Official Ruby is now at version 3.1.2
    - Bugs fixed, much faster
    - But, also needs much more memory
  - I designated MBARI Ruby as version 1.89
- The ESP application **requires** MBARI Ruby
  - Ruby 1.9, 2.x and 3.x are not compatible
  - There's no effort being made to convert ESP to run on Ruby 3.x
- MBARI Ruby can (and does) run on Macs, Unix and Linux