The Ruby Programming Language

CHAPTER 2

The Structure and Execution of Ruby Programs

2.1.1 Comments

```
# This entire line is a comment
x = "#This is a string"               # And this is a comment
y = /#This is a regular expression/   # Here's another comment

#
# This class represents a Complex number
# Despite its name, it is not complex at all.
#
```

2.1.1.1 Embedded documents

```
=begin Someone needs to fix the broken code below!
    <emphasis>Any code here is commented out</emphasis>
=end
---------------------------
# =begin This used to begin a comment. Now it is itself commented out!
    <emphasis>The code that goes here is no longer commented out</emphasis>
# =end
```

2.1.1.2 Documentation comments

```
   Rdoc comments use a simple markup grammar like those used in wikis.

   Separate paragraphs with a blank line.

   Headings begin with an equals sign

   = Headings
Headings
   == Sub-Headings
Sub-Headings
    The line above produces a subheading.
    === Sub-Sub-Heading
Sub-Sub-Heading
```

```
    And so on.

    = Examples
Examples

   Indented lines are displayed verbatim in code font.
     Be careful not to indent your headings and lists, though.

   = Lists and Fonts
Lists and Fonts
    List items begin with * or -. Indicate fonts with punctuation or HTML:
    * _italic_ or <i>multi-word italic</i>
italic or multi-word italic
    * *bold* or <b>multi-word bold</b>
bold or multi-word bold
    * +code+ or <tt>multi-word code</tt>
code or multi-word code
    1. Numbered lists begin with numbers.
    99. Any number will do; they don't have to be sequential.
    1. There is no way to do nested lists.
Numbered lists begin with numbers.
Any number will do; they don’t have to be sequential.
There is no way to do nested lists.
    The terms of a description list are bracketed:

    [item 1]  This is a description of item 1
    [item 2]  This is a description of item 2
item 1
This is a description of item 1
item 2
This is a description of item 2
```

2.1.2 Literals

```
1                       # An integer literal
1.0                     # A floating-point literal
'one'                   # A string literal
"two"                   # Another string literal
/three/                 # A regular expression literal
```

2.1.4 Identifiers

```
i
x2
old_value
_internal    # Identifiers may begin with underscores
PI           # Constant
```

2.1.4.2 Unicode characters in identifiers

```
def &#xD7;(x,y)  # The name of this method is the Unicode multiplication sign
  x*y        # The body of this method multiplies its arguments
end
```

2.1.4.3 Punctuation in identifiers

```
$files          # A global variable
@data           # An instance variable
@@counter       # A class variable
empty?          # A Boolean-valued method or predicate
sort!           # An in-place alternative to the regular sort method
timeout=        # A method invoked by assignment
```

2.1.5 Keywords

```
__LINE__        case        ensure      not         then

__ENCODING__    class       false       or          true

__FILE__        def         for         redo        undef

BEGIN           defined?    if          rescue      unless

END             do          in          retry       until

alias           else        module      return      when

and             elsif       next        self        while

begin           end         nil         super       yield

break


=begin    =end      __END__


# These are methods that appear to be statements or keywords

at_exit        catch         private        require
```

```
attr           include        proc           throw

attr_accessor  lambda         protected

attr_reader    load           public

attr_writer    loop           raise


# These are commonly used global functions

Array          chomp!         gsub!          select

Float          chop           iterator?      sleep

Integer        chop!          load           split

String         eval           open           sprintf

URI            exec           p              srand

abort          exit           print          sub

autoload       exit!          printf         sub!

autoload?      fail           putc           syscall

binding        fork           puts           system

block_given?   format         rand           test

callcc         getc           readline       trap

caller         gets           readlines      warn

chomp          gsub           scan


# These are commonly used object methods

allocate       freeze         kind_of?       superclass

clone          frozen?        method         taint

display        hash           methods        tainted?

dup            id             new            to_a

enum_for       inherited      nil?           to_enum

eql?           inspect        object_id      to_s
```

```
equal?          instance_of?   respond_to?    untaint

extend          is_a?          send
```

2.1.6.1 Newlines as statement terminators

```
total = x +     # Incomplete expression, parsing continues
  y

total = x  # This is a complete expression
  + y      # A useless but complete expression

var total = first_long_variable_name + second_long_variable_name \
  + third_long_variable_name # Note no statement terminator above

animals = Array.new
  .push("dog")   # Does not work in Ruby 1.8
  .push("cow")
  .push("cat")
  .sort
```

2.1.6.2 Spaces and method invocations

```
f(3+2)+1
f (3+2)+1
```

2.2 Syntactic Structure

```
[1,2,3]                # An Array literal
{1=>"one", 2=>"two"}   # A Hash literal
1..3                   # A Range literal

1          # A primary expression
x          # Another primary expression
x = 1      # An assignment expression
x = x + 1 # An expression with two operators

if x < 10 then   # If this expression is true
  x = x + 1      # Then execute this statement
end              # Marks the end of the conditional

while x < 10 do  # While this expression is true...
  print x        # Execute this statement
  x = x + 1      # Then execute this statement
```

```
end               # Marks the end of the loop
```

2.2.1 Block Structure in Ruby

```
3.times { print "Ruby! " }

1.upto(10) do |x|
  print x
end

module Stats                          # A module
  class Dataset                       # A class in the module
    def initialize(filename)          # A method in the class
      IO.foreach(filename) do |line|  # A block in the method
        if line[0,1] == "#"           # An if statement in the block
          next                        # A simple statement in the if
        end                           # End the if body
      end                             # End the block
    end                               # End the method body
  end                                 # End the class body
end                                   # End the module body
```

2.3 File Structure

```
#!/usr/bin/ruby -w          <lineannotation>shebang comment</lineannotation>
# -*- coding: utf-8 -*-     <lineannotation>coding comment</lineannotation>
require 'socket'            <lineannotation>load networking
library</lineannotation>

  ...                       <lineannotation>program code goes
here</lineannotation>

__END__                     <lineannotation>mark end of code</lineannotation>
  ...                       <lineannotation>program data goes
here</lineannotation>
```

2.4.1 Specifying Program Encoding

```
# coding: utf-8

# -*- coding: utf-8 -*-

# vi: set fileencoding=utf-8 :
```

```
#!/usr/bin/ruby -w
# coding: utf-8
```

2.4.2 Source Encoding and Default External Encoding

```
ruby -E utf-8             # Encoding name follows -E
ruby -Eutf-8              # The space is optional
ruby --encoding utf-8     # Encoding following --encoding with a space
ruby --encoding=utf-8     # Or use an equals sign with --encoding
```