Ruby Programming Course Notes

Complete Beginner's Guide (Mac Edition)

Based on FreeCodeCamp Course

June 11, 2025

Contents

1 Introduction

Ruby is an extremely popular programming language and the foundation behind Ruby on Rails, one of the most popular web development frameworks. This course covers everything from basic installation to advanced object-oriented programming concepts.

2 Installation on Mac

2.1 Checking Existing Installation

Mac computers come with Ruby pre-installed. To check your current Ruby version:

1. Open Terminal (search for "Terminal" in Spotlight)
2. Type: `ruby -v`
3. Press Enter

You should see output showing your Ruby version (e.g., "ruby 2.7.0").

2.2 Updating Ruby (Optional)

If you need a newer version of Ruby, use the Ruby Version Manager (RVM):

```
# Install RVM
\curl -sSL https://get.rvm.io | bash -s stable

# Reload your terminal or run:
source ~/.rvm/scripts/rvm

# Install latest Ruby
rvm install ruby

# Use the new version
rvm use ruby --default
```

3 Development Environment Setup

3.1 Text Editor: Atom

1. Visit `atom.io`
2. Download for Mac
3. Move to Applications folder
4. Install the `atom-runner` package:
   - Go to Atom → Preferences → Install
   - Search for "atom-runner"
   - Click Install

3.2 Your First Ruby Program

Create a new file with `.rb` extension:

```ruby
# hello.rb
print "Hello World"
```

Run with Ctrl+R in Atom (using atom-runner) or in Terminal:

```
ruby hello.rb
```

4 Basic Ruby Concepts

4.1 Printing Output

```ruby
# print - no newline after output
print "Hello"
print "World"  # Output: HelloWorld

# puts - adds newline after output
puts "Hello"
puts "World"   # Output:
               # Hello
               # World
```

4.2 Drawing with Text

```ruby
puts "    /|"
puts "   / |"
puts "  /  |"
puts " /   |"
puts "/____|"
# Creates a simple triangle
```

5 Variables

Variables are containers for storing data values:

```ruby
# Variable assignment
character_name = "John"
character_age = 35

# Using variables in strings
puts "There once was a man named " + character_name
puts "He was " + character_age.to_s + " years old"

# Modifying variables
character_name = "Mike"
puts "But everybody called him " + character_name
```

Key Points:

- Use lowercase with underscores for variable names

- Convert numbers to strings with `.to_s` when concatenating
- Variables can be reassigned throughout the program

6 Data Types

6.1 Strings

Text data enclosed in quotes:

```
name = "Mike"
occupation = "programmer"
```

6.2 Numbers

Integers (whole numbers):

```
age = 75
negative_age = -75
```

Floats (decimal numbers):

```
gpa = 3.2
temperature = -4.5
```

6.3 Booleans

True or false values:

```
is_male = true
is_tall = false
```

6.4 Nil

Represents "no value":

```
flaws = nil
```

7 Working with Strings

7.1 String Methods

```
phrase = "Giraffe Academy"

# Convert case
puts phrase.upcase      # "GIRAFFE ACADEMY"
puts phrase.downcase    # "giraffe academy"

# Remove whitespace
phrase = "   Giraffe Academy   "
puts phrase.strip       # "Giraffe Academy"
```

```ruby

# String information
puts phrase.length      # Returns number of characters
puts phrase.include? "Academy"  # Returns true/false

# Accessing characters
puts phrase[0]          # First character (G)
puts phrase[1]          # Second character (i)
puts phrase[0, 3]       # Range: first 3 characters

# Finding text
puts phrase.index("A")  # Returns position of "A"
```

7.2 Special Characters

```ruby
# Quotation marks in strings
puts "He said \"Hello\""

# New line
puts "Line 1\nLine 2"
```

8 Math and Numbers

8.1 Basic Arithmetic

```ruby
puts 5 + 9      # Addition: 14
puts 5 - 2      # Subtraction: 3
puts 5 * 3      # Multiplication: 15
puts 10 / 3     # Division: 3 (integer division)
puts 10.0 / 3   # Division: 3.333\ldots{} (float division)

# Exponents
puts 2 ** 3     # 2 to the power of 3: 8

# Modulus (remainder)
puts 10 % 3     # Remainder of 10 / 3: 1
```

8.2 Number Methods

```ruby
num = -20.487

puts num.abs        # Absolute value: 20.487
puts num.round      # Round: -20
puts num.ceil       # Ceiling: -20
puts num.floor      # Floor: -21
```

8.3 Math Class

```ruby
puts Math.sqrt(36)  # Square root: 6.0
puts Math.log(1)    # Natural logarithm: 0.0
```

9 Getting User Input

Important: For user input, use Terminal instead of atom-runner.

```ruby
# Basic input
puts "Enter your name: "
name = gets.chomp

puts "Enter your age: "
age = gets.chomp

puts "Hello " + name + ", you are " + age
```

Key Points:

- `gets` gets user input
- `chomp` removes the newline character
- All input comes as strings - convert with `.to_i` or `.to_f`

10 Building a Calculator

```ruby
puts "Enter first number: "
num1 = gets.chomp.to_f

puts "Enter operator (+, -, *, /): "
op = gets.chomp

puts "Enter second number: "
num2 = gets.chomp.to_f

if op == "+"
    puts num1 + num2
elsif op == "-"
    puts num1 - num2
elsif op == "*"
    puts num1 * num2
elsif op == "/"
    puts num1 / num2
else
    puts "Invalid operator"
end
```

11 Mad Libs Game

```ruby
puts "Enter a color: "
color = gets.chomp

puts "Enter a plural noun: "
plural_noun = gets.chomp

puts "Enter a celebrity: "
```

```ruby
celebrity = gets.chomp

puts "Roses are " + color
puts plural_noun + " are blue"
puts "I love " + celebrity
```

12 Arrays

Arrays store multiple values in a single variable:

```ruby
# Creating arrays
friends = Array["Kevin", "Karen", "Oscar"]
# or
friends = ["Kevin", "Karen", "Oscar"]

# Accessing elements
puts friends[0]      # "Kevin" (first element)
puts friends[-1]     # "Oscar" (last element)
puts friends[0, 2]   # First 2 elements

# Modifying arrays
friends[0] = "Dwight"
friends[5] = "Holly"  # Creates nil elements in between

# Array methods
puts friends.length     # Number of elements
puts friends.include? "Karen"  # true/false
puts friends.reverse    # Reversed array
puts friends.sort       # Sorted array
```

13 Hashes

Hashes store key-value pairs:

```ruby
# Creating hashes
states = {
    "Pennsylvania" => "PA",
    "New York" => "NY",
    "Oregon" => "OR"
}

# Alternative syntax
states = {
    :Pennsylvania => "PA",
    :New_York => "NY",
    :Oregon => "OR"
}

# Accessing values
puts states["Pennsylvania"]  # "PA"
puts states[:Pennsylvania]   # "PA" (symbol key)
```

14 Methods (Functions)

```ruby
# Basic method
def say_hi
    puts "Hello User"
end

# Call the method
say_hi

# Method with parameters
def say_hi(name, age)
    puts "Hello " + name + ", you are " + age.to_s
end

say_hi("Mike", 25)

# Method with default parameters
def say_hi(name="No name", age=-1)
    puts "Hello " + name + ", you are " + age.to_s
end

say_hi          # Uses defaults
say_hi("Mike")  # Uses default age
```

15 Return Statements

```ruby
# Method that returns a value
def cube(num)
    return num * num * num
    puts "This won't execute"  # Code after return is ignored
end

puts cube(3)  # Prints 27

# Ruby automatically returns the last expression
def cube(num)
    num * num * num  # Implicit return
end

# Returning multiple values
def get_name_and_age
    return "Mike", 25
end

name, age = get_name_and_age
```

16 If Statements

16.1 Basic If Statements

```ruby
is_male = true
is_tall = false

if is_male
    puts "You are male"
else
    puts "You are not male"
end

# Multiple conditions
if is_male and is_tall
    puts "You are a tall male"
elsif is_male and !is_tall
    puts "You are a short male"
elsif !is_male and is_tall
    puts "You are not male but are tall"
else
    puts "You are not male and not tall"
end
```

16.2 Comparison Operators

```ruby
# Comparisons
def max(num1, num2, num3)
    if num1 >= num2 and num1 >= num3
        return num1
    elsif num2 >= num1 and num2 >= num3
        return num2
    else
        return num3
    end
end

puts max(1, 2, 3)  # Returns 3
```

Comparison operators:

- == equal to
- != not equal to
- > greater than
- >= greater than or equal to
- < less than
- <= less than or equal to

17 Case Expressions

For checking multiple conditions against the same value:

```ruby
def get_day_name(day)
    day_name = ""

    case day
    when "mon"
        day_name = "Monday"
    when "tue"
        day_name = "Tuesday"
    when "wed"
        day_name = "Wednesday"
    when "thu"
        day_name = "Thursday"
    when "fri"
        day_name = "Friday"
    when "sat"
        day_name = "Saturday"
    when "sun"
        day_name = "Sunday"
    else
        day_name = "Invalid abbreviation"
    end

    return day_name
end

puts get_day_name("mon")  # "Monday"
```

18 While Loops

```ruby
# Basic while loop
index = 1
while index <= 5
    puts index
    index += 1  # Same as: index = index + 1
end
# Prints: 1, 2, 3, 4, 5

# Be careful of infinite loops!
# Make sure the condition eventually becomes false
```

19 Building a Guessing Game

```ruby
secret_word = "giraffe"
guess = ""
guess_count = 0
guess_limit = 3
out_of_guesses = false

while guess != secret_word and !out_of_guesses
    if guess_count < guess_limit
        puts "Enter guess: "
        guess = gets.chomp
```

```
        guess_count += 1
    else
        out_of_guesses = true
    end
end

if out_of_guesses
    puts "You Lose!"
else
    puts "You Win!"
end
```

20 For Loops

```
# Loop through array
friends = ["Kevin", "Karen", "Oscar"]
for friend in friends
    puts friend
end

# Loop through range
for index in 0..5
    puts index
end

# Using times method
6.times do |index|
    puts index
end

# Using each method
friends.each do |friend|
    puts friend
end
```

21 Building an Exponent Method

```
def pow(base_num, pow_num)
    result = 1
    pow_num.times do
        result = result * base_num
    end
    return result
end

puts pow(2, 3)  # 2^3 = 8
puts pow(5, 2)  # 5^2 = 25
```

22 Comments

```ruby
# This is a single line comment

puts "Hello World"  # Comment after code

# Multiple line comments
# Line 1 of comment
# Line 2 of comment

=begin
Multi-line comment block
Everything here is ignored
=end
```

23 Reading Files

```ruby
# Reading entire file
File.open("employees.txt", "r") do |file|
    puts file.read
end

# Reading line by line
File.open("employees.txt", "r") do |file|
    puts file.readline  # First line
    puts file.readline  # Second line
end

# Reading all lines into array
File.open("employees.txt", "r") do |file|
    for line in file.readlines
        puts line
    end
end

# Alternative way to open files
file = File.open("employees.txt", "r")
puts file.read
file.close  # Always close files opened this way
```

24 Writing Files

```ruby
# Append to file
File.open("employees.txt", "a") do |file|
    file.write("\nOscar, Accounting")
end

# Overwrite file
File.open("employees.txt", "w") do |file|
    file.write("Angela, Accounting")
end

# Create new file
File.open("index.html", "w") do |file|
```

```ruby
    file.write("<h1>Hello World</h1>")
end

# Read and write
File.open("employees.txt", "r+") do |file|
    file.readline  # Move cursor to next line
    file.write("Overwritten")
end
```

25 Handling Errors

```ruby
# Basic error handling
begin
    num = 10 / 0
rescue
    puts "Error occurred"
end

# Handling specific errors
begin
    num = 10 / 0
    puts nums[5]
rescue ZeroDivisionError
    puts "Division by zero error"
rescue TypeError => e
    puts "Type error: " + e.to_s
end
```

26 Classes and Objects

26.1 Creating a Class

```ruby
class Book
    attr_accessor :title, :author, :pages

    def initialize(title, author, pages)
        @title = title
        @author = author
        @pages = pages
    end

    def is_long?
        return @pages > 300
    end
end

# Creating objects
book1 = Book.new("Harry Potter", "JK Rowling", 400)
book2 = Book.new("Lord of the Rings", "Tolkien", 500)

# Using objects
puts book1.title
puts book1.is_long?
```

26.2 Object Methods

```ruby
class Student
    attr_accessor :name, :major, :gpa

    def initialize(name, major, gpa)
        @name = name
        @major = major
        @gpa = gpa
    end

    def has_honors
        if @gpa >= 3.5
            return true
        else
            return false
        end
    end
end

student1 = Student.new("Jim", "Business", 2.6)
student2 = Student.new("Pam", "Art", 3.6)

puts student1.has_honors  # false
puts student2.has_honors  # true
```

27 Building a Quiz

```ruby
class Question
    attr_accessor :prompt, :answer

    def initialize(prompt, answer)
        @prompt = prompt
        @answer = answer
    end
end

p1 = "What color are apples?\n(a) red\n(b) purple\n(c) orange"
p2 = "What color are bananas?\n(a) pink\n(b) red\n(c) yellow"
p3 = "What color are pears?\n(a) yellow\n(b) green\n(c) orange"

questions = [
    Question.new(p1, "a"),
    Question.new(p2, "c"),
    Question.new(p3, "b")
]

def run_test(questions)
    answer = ""
    score = 0

    for question in questions
        puts question.prompt
        answer = gets.chomp
        if answer == question.answer
```

```
            score += 1
        end
    end

    puts "You got " + score.to_s + "/" + questions.length.to_s
end

run_test(questions)
```

28 Inheritance

```
# Base class
class Chef
    def make_chicken
        puts "The chef makes chicken"
    end

    def make_salad
        puts "The chef makes salad"
    end

    def make_special_dish
        puts "The chef makes BBQ ribs"
    end
end

# Subclass inheriting from Chef
class ItalianChef < Chef
    def make_special_dish  # Override parent method
        puts "The chef makes eggplant parm"
    end

    def make_pasta  # New method specific to ItalianChef
        puts "The chef makes pasta"
    end
end

chef = Chef.new
italian_chef = ItalianChef.new

chef.make_special_dish          # "BBQ ribs"
italian_chef.make_special_dish  # "eggplant parm"
italian_chef.make_pasta         # Only available to ItalianChef
```

29 Modules

Create a separate file `useful_tools.rb`:

```
module Tools
    def self.say_hi(name)
        puts "Hello " + name
    end

```

```ruby
    def self.say_bye(name)
        puts "Goodbye " + name
    end
end
```

Use in main file:

```ruby
require_relative "useful_tools"

Tools.say_hi("Mike")
Tools.say_bye("Mike")
```

30 Interactive Ruby (IRB)

IRB allows you to test Ruby code interactively:

```ruby
# In Terminal
irb

# Now you can type Ruby code directly:
puts "Hello World"
2 + 3
name = "Mike"
puts name

# Exit IRB
exit
```

31 Best Practices

- Use descriptive variable and method names
- Use `snake_case` for variables and methods
- Use `CamelCase` for class names
- Always close files when not using blocks
- Handle potential errors with begin/rescue
- Use comments sparingly and only when necessary
- Keep methods short and focused on one task
- Use modules to organize related methods

32 Advanced Topics for Intermediate Ruby Developers

After mastering the basics, these advanced topics will help you become a professional Ruby developer:

32.1 Modern Development Environment

32.1.1 Version Management

Use ASDF (recommended in 2024-2025) for managing Ruby versions:

```
# Install ASDF
brew install asdf

# Add Ruby plugin
asdf plugin add ruby

# Install latest Ruby
asdf install ruby latest
asdf global ruby latest
```

32.1.2 Modern Editor Setup

For VS Code with Ruby LSP (the modern standard):

1. Install the "Ruby LSP" extension
2. Install the "Ruby Solargraph" extension for additional features
3. Configure settings for auto-formatting and linting

32.2 Advanced Language Features

32.2.1 Blocks, Procs, and Lambdas

```
# Blocks - anonymous functions passed to methods
[1, 2, 3].each { |num| puts num * 2 }

# Procs - objects that wrap blocks
double_proc = Proc.new { |x| x * 2 }
[1, 2, 3].map(&double_proc)

# Lambdas - special procs with method-like behavior
double_lambda = lambda { |x| x * 2 }
# or using stabby lambda syntax
double_lambda = ->(x) { x * 2 }

# Key differences:
# -- Lambdas check argument count, procs don't
# -- return in lambda returns from lambda, in proc returns from
   enclosing method
```

32.2.2 Metaprogramming Basics

```
# Dynamic method definition
class DynamicClass
  %w[name age email].each do |attr|
    define_method(attr) do
      instance_variable_get("@#{attr}")
```

```ruby
    end

    define_method("#{attr}=") do |value|
      instance_variable_set("@#{attr}", value)
    end
  end
end

# method_missing for flexible APIs
class FlexibleHash
  def initialize
    @data = {}
  end

  def method_missing(method_name, *args)
    if method_name.to_s.end_with?('=')
      @data[method_name.to_s.chomp('=')] = args.first
    else
      @data[method_name.to_s]
    end
  end
end

obj = FlexibleHash.new
obj.name = "Ruby"
puts obj.name  # "Ruby"
```

32.2.3 Advanced Enumerable Operations

```ruby
# Lazy evaluation for large datasets
(1..Float::INFINITY).lazy
  .select(&:even?)
  .take(10)
  .to_a  # [2, 4, 6, 8, 10, 12, 14, 16, 18, 20]

# Custom enumerators
class Fibonacci
  include Enumerable

  def each
    a, b = 0, 1
    loop do
      yield a
      a, b = b, a + b
    end
  end
end

Fibonacci.new.take(10)  # [0, 1, 1, 2, 3, 5, 8, 13, 21, 34]
```

32.3 Testing Frameworks

32.3.1 RSpec (Industry Standard)

Install and setup:

```ruby
# Add to Gemfile
gem 'rspec'

# Generate RSpec configuration
rspec --init
```

Basic RSpec example:

```ruby
# spec/calculator_spec.rb
require 'rspec'

describe Calculator do
  describe '#add' do
    it 'adds two numbers correctly' do
      calculator = Calculator.new
      result = calculator.add(2, 3)
      expect(result).to eq(5)
    end

    context 'with negative numbers' do
      it 'handles negative numbers' do
        calculator = Calculator.new
        result = calculator.add(-2, 3)
        expect(result).to eq(1)
      end
    end
  end
end
```

32.3.2 Debugging with Pry

```ruby
# Add to Gemfile
gem 'pry-byebug'

# Usage in code
def complex_method
  data = fetch_data
  binding.pry  # Debugger will stop here
  process_data(data)
end

# Debugging commands:
# next - next line
# step - step into method
# continue - continue execution
# whereami - show current location
```

32.4 Code Quality Tools

32.4.1 RuboCop Configuration

```yaml
# .rubocop.yml
AllCops:
  TargetRubyVersion: 3.3
```

```
  NewCops: enable

Style/Documentation:
  Enabled: false

Metrics/LineLength:
  Max: 120

Metrics/MethodLength:
  Max: 20
```

32.4.2 SimpleCov for Test Coverage

```ruby
# Add to spec_helper.rb or test_helper.rb
require 'simplecov'
SimpleCov.start do
  add_filter '/spec/'
  add_filter '/test/'
end

# This will generate coverage reports in coverage/
```

32.5 Essential Gems for Professional Development

32.5.1 Must-Have Development Gems

```ruby
# Gemfile
group :development, :test do
  gem 'rspec-rails'
  gem 'factory_bot_rails'
  gem 'pry-byebug'
  gem 'rubocop'
  gem 'simplecov'
end

group :test do
  gem 'capybara'      # Browser automation
  gem 'webmock'       # HTTP request stubbing
end

# Production gems
gem 'faraday'         # HTTP client
gem 'sidekiq'         # Background jobs
gem 'redis'           # Caching and sessions
```

32.6 Database Best Practices

32.6.1 ActiveRecord Optimization

```ruby
# Avoid N+1 queries
users = User.includes(:posts).where(active: true)

# Use find_each for large datasets
```

```ruby
User.find_each(batch_size: 1000) do |user|
  # Process each user
end

# Select specific columns
User.select(:id, :name, :email).where(active: true)

# Use scopes for reusable queries
class User < ApplicationRecord
  scope :active, -> { where(active: true) }
  scope :recent, -> { where('created_at > ?', 1.week.ago) }
end
```

32.7 Web Development Frameworks

32.7.1 Ruby on Rails (Most Popular)

```ruby
# Install Rails
gem install rails

# Create new application
rails new my_app --database=postgresql

# Generate scaffold
rails generate scaffold Post title:string content:text

# Run migrations
rails db:migrate

# Start server
rails server
```

32.7.2 Sinatra (Lightweight)

```ruby
# app.rb
require 'sinatra'

get '/' do
  'Hello World!'
end

get '/users/:id' do
  user = User.find(params[:id])
  user.to_json
end

post '/users' do
  user = User.create(JSON.parse(request.body.read))
  user.to_json
end
```

32.8 Performance Optimization

32.8.1 Memory Management

```ruby
# Use symbols for repeated identifiers
hash = { name: 'Ruby', type: 'Language' }  # Good
hash = { 'name' => 'Ruby', 'type' => 'Language' }  # Creates new strings

# Memoization for expensive operations
def expensive_calculation
  @result ||= begin
    # Expensive computation here
    complex_algorithm
  end
end

# Use lazy evaluation for large collections
large_array.lazy.map(&:expensive_operation).take(10)
```

32.8.2 Benchmarking

```ruby
require 'benchmark'

# Compare different approaches
Benchmark.bm do |x|
  x.report("each:") { array.each { |item| process(item) } }
  x.report("map:") { array.map { |item| process(item) } }
end

# Using benchmark-ips gem
require 'benchmark/ips'

Benchmark.ips do |x|
  x.report("string interpolation") { "Hello #{name}!" }
  x.report("string concatenation") { "Hello " + name + "!" }
  x.compare!
end
```

32.9 Security Best Practices

```ruby
# Input validation
class User < ApplicationRecord
  validates :email, presence: true, format: { with: URI::MailTo::EMAIL_REGEXP }
  validates :age, numericality: { greater_than: 0, less_than: 150 }
end

# SQL injection prevention
User.where("name = ?", params[:name])  # Good
User.where("name = '#{params[:name]}'")  # BAD - SQL injection risk

# XSS prevention in views (Rails automatically escapes)
<%= user.name %>  # Automatically escaped
<%== user.bio %>  # Raw HTML (use carefully)
```

32.10 Creating and Publishing Gems

```
# Create new gem
bundle gem my_gem

# Gem structure
my_gem/
|-- lib/
|   \-- my_gem.rb
|-- spec/
|-- my_gem.gemspec
|-- Gemfile
\-- README.md

# Build and publish
gem build my_gem.gemspec
gem push my_gem-1.0.0.gem
```

33 Ruby Style Guide and Community Standards

Follow the community Ruby Style Guide:

- Use 2 spaces for indentation
- Line length: 80-120 characters
- Use `snake_case` for variables and methods
- Use `CamelCase` for classes and modules
- Use `SCREAMING_SNAKE_CASE` for constants
- Prefer single quotes for strings unless interpolation is needed
- Use trailing commas in multi-line arrays and hashes

34 Next Steps for Professional Ruby Development

After mastering these intermediate concepts:

- Learn Ruby on Rails in depth for web development
- Study design patterns and clean architecture
- Master test-driven development (TDD) and behavior-driven development (BDD)
- Explore advanced metaprogramming and DSL creation
- Learn about Ruby internals and performance optimization
- Contribute to open source Ruby projects
- Study concurrent programming with Ruby fibers and threads
- Learn DevOps practices for Ruby applications (Docker, CI/CD)

35 Recommended Learning Resources

Books:

- "Metaprogramming Ruby 2" by Paolo Perrotta
- "Practical Object-Oriented Design in Ruby" by Sandi Metz
- "Ruby Under a Microscope" by Pat Shaughnessy
- "Rails Antipatterns" by Chad Pytel and Tammer Saleh

Online Resources:

- Ruby Style Guide: `https://rubystyle.guide/`
- RubyGems.org for exploring gems
- Ruby Weekly newsletter
- RubyFlow community news
- Ruby Documentation: `https://ruby-doc.org/`