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
```

2.2.1 Performance Testing

```
# Using benchmark-ips for performance testing
require 'benchmark/ips'

# spec/performance/string_concatenation_spec.rb
RSpec.describe 'String Concatenation Performance' do
  let(:strings) { Array.new(1000) { "string_#{rand(1000)}" } }

  it 'compares string concatenation methods' do
    Benchmark.ips do |x|
      x.report('Array#join') do
        strings.join('')
      end

      x.report('String interpolation') do
        result = ''
```

```
        strings.each { |s| result = "#{result}#{s}" }
        result
      end

      x.report('String +=') do
        result = ''
        strings.each { |s| result += s }
        result
      end

      x.compare!
    end
  end
end

# Load testing with parallel execution
RSpec.describe 'Concurrent User Processing' do
  it 'handles multiple simultaneous requests' do
    threads = []
    results = []
    mutex = Mutex.new

    # Simulate 10 concurrent users
    10.times do |i|
      threads << Thread.new do
        user = create(:user, email: "user#{i}@example.com")
        service = UserProcessingService.new
        result = service.process_user(user)

        mutex.synchronize do
          results << result
        end
      end
    end

    threads.each(&:join)

    expect(results.length).to eq(10)
    expect(results.all?(&:success?)).to be true
  end
end
```

2.2.2 Test Doubles and Mocking Strategies

```
# Different types of test doubles
RSpec.describe 'Test Doubles Examples' do
  describe 'method stubs' do
    it 'stubs method return values' do
      user = double('User')
      allow(user).to receive(:name).and_return('John Doe')
      allow(user).to receive(:age).and_return(30)

      expect(user.name).to eq('John Doe')
      expect(user.age).to eq(30)
    end
  end

```

```
  describe 'message expectations' do
    it 'verifies method calls' do
      email_service = double('EmailService')
      expect(email_service).to receive(:send_email)
        .with('user@example.com', 'Welcome!')
        .once

      user_service = UserService.new(email_service)
      user_service.welcome_user('user@example.com')
    end
  end

  describe 'partial doubles' do
    it 'stubs real object methods' do
      user = create(:user)
      allow(user).to receive(:external_api_call).and_return({ status: 'success' })

      result = user.sync_with_external_service
      expect(result[:status]).to eq('success')
    end
  end

  describe 'spy doubles' do
    it 'records method calls for later verification' do
      logger = spy('Logger')

      service = PaymentService.new(logger: logger)
      service.process_payment(amount: 100)

      expect(logger).to have_received(:info).with(/Payment processed/)
    end
  end

  describe 'null object pattern' do
    it 'handles missing methods gracefully' do
      null_logger = double('NullLogger').as_null_object

      service = PaymentService.new(logger: null_logger)

      # These won't raise errors even if methods don't exist
      expect { service.process_payment(amount: 100) }.not_to raise_error
    end
  end
end
```

2.2.3 Testing Asynchronous Code

```
# Testing background jobs
RSpec.describe 'Background Job Testing' do
  describe WelcomeEmailJob do
    include ActiveJob::TestHelper

    it 'enqueues welcome email job' do
      expect {
        UserService.new.create_user(email: 'test@example.com')
```

```
      }.to have_enqueued_job(WelcomeEmailJob)
        .with(hash_including(email: 'test@example.com'))
    end

    it 'processes job immediately in test' do
      perform_enqueued_jobs do
        UserService.new.create_user(email: 'test@example.com')
      end

      expect(ActionMailer::Base.deliveries.count).to eq(1)
    end

    it 'processes job at specific time' do
      travel_to 1.hour.from_now do
        expect {
          WelcomeEmailJob.set(wait: 1.hour).perform_later('test@example
    .com')
        }.to have_enqueued_job.at(1.hour.from_now)
      end
    end
  end
end

# Testing with timeouts and polling
RSpec.describe 'Async Processing' do
  it 'waits for async operation to complete' do
    service = AsyncProcessingService.new
    service.start_processing

    # Poll until completion or timeout
    Timeout.timeout(5) do
      loop do
        break if service.processing_complete?
        sleep 0.1
      end
    end

    expect(service.result).to be_present
  end
end
```

2.2.4 Database Testing Strategies

```
# Testing database constraints and triggers
RSpec.describe 'Database Constraints' do
  it 'enforces unique constraint at database level' do
    create(:user, email: 'test@example.com')

    expect {
      # Bypass ActiveRecord validations
      User.connection.execute(
        "INSERT INTO users (email, created_at, updated_at)
         VALUES ('test@example.com', NOW(), NOW())"
      )
    }.to raise_error(ActiveRecord::RecordNotUnique)
  end

```

```ruby
  it 'tests database triggers' do
    user = create(:user)

    # Assuming we have a trigger that updates updated_at
    User.connection.execute(
      "UPDATE users SET email = 'new@example.com' WHERE id = #{user.id}
    "
    )

    user.reload
    expect(user.updated_at).to be > user.created_at
  end
end

# Testing transactions and rollbacks
RSpec.describe 'Transaction Testing' do
  it 'rolls back on failure' do
    initial_count = User.count

    expect {
      User.transaction do
        create(:user)
        create(:user)
        raise ActiveRecord::Rollback
      end
    }.not_to change(User, :count)

    expect(User.count).to eq(initial_count)
  end

  it 'handles nested transactions' do
    User.transaction do
      user1 = create(:user)

      expect {
        User.transaction(requires_new: true) do
          create(:user)
          raise ActiveRecord::Rollback
        end
      }.not_to change(User, :count).from(User.count)

      expect(user1).to be_persisted
    end
  end
end
```

2.2.5 Security Testing

```ruby
# Testing authorization
RSpec.describe 'Authorization Testing' do
  let(:user) { create(:user) }
  let(:admin) { create(:user, :admin) }
  let(:other_user) { create(:user) }

  describe 'access control' do
    it 'allows users to access their own resources' do
      post = create(:post, user: user)
```

```
10
11        ability = Ability.new(user)
12        expect(ability.can?(:read, post)).to be true
13        expect(ability.can?(:update, post)).to be true
14      end
15
16      it 'prevents users from accessing others resources' do
17        post = create(:post, user: other_user)
18
19        ability = Ability.new(user)
20        expect(ability.can?(:read, post)).to be true  # Public read
21        expect(ability.can?(:update, post)).to be false
22      end
23
24      it 'allows admins to access all resources' do
25        post = create(:post, user: user)
26
27        ability = Ability.new(admin)
28        expect(ability.can?(:manage, post)).to be true
29      end
30    end
31  end
32
33  # Testing input sanitization
34  RSpec.describe 'Input Sanitization' do
35    it 'prevents XSS attacks' do
36      malicious_input = '<script>alert("XSS")</script>'
37
38      post '/posts', params: { post: { content: malicious_input } }
39
40      created_post = Post.last
41      expect(created_post.content).not_to include('<script>')
42      expect(created_post.content).to include('&lt;script&gt;')
43    end
44
45    it 'prevents SQL injection' do
46      malicious_input = "'; DROP TABLE users; --"
47
48      expect {
49        User.where("name = ?", malicious_input).to_a
50      }.not_to change(User, :count)
51    end
52  end
```

2.2.6 API Testing Best Practices

```
1  # Shared examples for API testing
2  RSpec.shared_examples 'API authentication' do
3    context 'without authentication' do
4      it 'returns unauthorized status' do
5        subject
6        expect(response).to have_http_status(:unauthorized)
7      end
8    end
9
10   context 'with invalid token' do
```

```ruby
    let(:auth_headers) { { 'Authorization' => 'Bearer invalid_token' } }

    it 'returns unauthorized status' do
      subject
      expect(response).to have_http_status(:unauthorized)
    end
  end
end

RSpec.shared_examples 'API error handling' do
  it 'returns JSON error format' do
    subject

    json_response = JSON.parse(response.body)
    expect(json_response).to have_key('error')
    expect(json_response).to have_key('message')
  end
end

# API contract testing
RSpec.describe 'API Contract Testing' do
  describe 'GET /api/v1/users/:id' do
    let(:user) { create(:user) }
    let(:auth_headers) { { 'Authorization' => "Bearer #{user.auth_token}" } }

    subject { get "/api/v1/users/#{user.id}", headers: auth_headers }

    include_examples 'API authentication'

    context 'with valid authentication' do
      it 'returns user data in expected format' do
        subject

        json_response = JSON.parse(response.body)
        user_data = json_response['user']

        # Test the contract structure
        expect(user_data).to match({
          'id' => user.id,
          'email' => user.email,
          'first_name' => user.first_name,
          'last_name' => user.last_name,
          'created_at' => user.created_at.iso8601,
          'updated_at' => user.updated_at.iso8601
        })
      end

      it 'does not expose sensitive data' do
        subject

        json_response = JSON.parse(response.body)
        user_data = json_response['user']

        expect(user_data).not_to have_key('password_digest')
        expect(user_data).not_to have_key('auth_token')
        expect(user_data).not_to have_key('reset_password_token')
```

```
      end
    end
  end
end
```

2.2.7 Custom Matchers

```ruby
# spec/support/custom_matchers.rb
RSpec::Matchers.define :be_a_valid_email do
  match do |email|
    email =~ /\A[\w+\-.]+@[a-z\d\-]+(\.[a-z\d\-]+)*\.[a-z]+\z/i
  end

  failure_message do |email|
    "expected '#{email}' to be a valid email address"
  end
end

RSpec::Matchers.define :have_error_on do |attribute|
  match do |model|
    model.valid?
    model.errors[attribute].any?
  end

  failure_message do |model|
    "expected #{model.class} to have error on #{attribute}, but it didn't"
  end
end

RSpec::Matchers.define :contain_json do |expected|
  match do |actual|
    @actual_json = JSON.parse(actual)
    expected.all? do |key, value|
      @actual_json[key] == value
    end
  rescue JSON::ParserError
    false
  end

  failure_message do |actual|
    "expected JSON #{@actual_json} to contain #{expected}"
  end
end

# Usage examples
RSpec.describe 'Custom Matchers' do
  it 'validates email format' do
    expect('test@example.com').to be_a_valid_email
    expect('invalid-email').not_to be_a_valid_email
  end

  it 'checks model errors' do
    user = User.new(email: '')
    expect(user).to have_error_on(:email)
  end

```

```
  it 'validates JSON response' do
    json_response = '{"status": "success", "data": {"id": 1}}'
    expect(json_response).to contain_json('status' => 'success')
  end
end
```

2.2.8 Continuous Integration Testing

```
# .github/workflows/test.yml
name: Test Suite

on:
  push:
    branches: [ main, develop ]
  pull_request:
    branches: [ main ]

jobs:
  test:
    runs-on: ubuntu-latest

    services:
      postgres:
        image: postgres:13
        env:
          POSTGRES_PASSWORD: postgres
        options: >-
          --health-cmd pg_isready
          --health-interval 10s
          --health-timeout 5s
          --health-retries 5

    env:
      RAILS_ENV: test
      DATABASE_URL: postgres://postgres:postgres@localhost:5432/test_db

    steps:
    - uses: actions/checkout@v3

    - name: Set up Ruby
      uses: ruby/setup-ruby@v1
      with:
        ruby-version: 3.3
        bundler-cache: true

    - name: Setup Database
      run: |
        bundle exec rails db:create
        bundle exec rails db:schema:load

    - name: Run RuboCop
      run: bundle exec rubocop

    - name: Run Tests
      run: |
        bundle exec rspec --format progress \
                          --format RspecJunitFormatter \
```

```
                          --out tmp/rspec.xml

    - name: Upload Coverage
      uses: codecov/codecov-action@v3
      with:
        file: ./coverage/coverage.xml

    - name: Archive test results
      uses: actions/upload-artifact@v3
      if: always()
      with:
        name: test-results
        path: tmp/rspec.xml
```

2.2.9 Testing Checklist

Before Writing Tests:

- Understand the requirement clearly
- Identify edge cases and error scenarios
- Plan test data setup strategy
- Consider performance implications

Test Quality Checklist:

- Tests are independent and can run in any order
- Each test has a single, clear assertion
- Test names describe the expected behavior
- Setup data is minimal and relevant
- Tests run quickly (under 100ms for unit tests)
- No hardcoded dates or environment-specific values
- Tests clean up after themselves

Coverage Guidelines:

- Aim for 90%+ line coverage
- 100% coverage of critical business logic
- All public methods should have tests
- Edge cases and error conditions covered
- Integration points tested

2.3 Debugging Advanced Techniques

2.3.1 Advanced Pry Usage

```ruby
# .pryrc file in home directory
Pry.config.editor = 'code'  # or 'vim', 'nano', etc.

# Custom commands
Pry::Commands.create_command "sql" do
  description "Execute SQL query"
  def process(query)
    result = ActiveRecord::Base.connection.execute(query)
    ap result.to_a
  end
end

# Useful pry commands in debugging sessions:
# ls                    # List methods and variables
# cd object             # Change scope to object
# show-method method    # Show method source
# edit method           # Edit method in editor
# whereami              # Show current location
# wtf?                  # Show last exception backtrace
# hist                  # Show command history
# $                     # Show last result
```

2.3.2 Memory Debugging

```ruby
# Using memory_profiler gem
require 'memory_profiler'

# Profile a block of code
report = MemoryProfiler.report do
  1000.times { User.new(name: "User") }
end

puts report.pretty_print

# Profile specific methods
class UserService
  def self.profile_create_users
    MemoryProfiler.report do
      create_users(1000)
    end.pretty_print
  end
end

# Using allocation_tracer
require 'allocation_tracer'

AlocationTracer.trace do
  # Your code here
end
```

2.3.3 Performance Profiling

```ruby
# Using ruby-prof
require 'ruby-prof'

# Profile CPU time
RubyProf.start
# Your code here
result = RubyProf.stop

# Print a flat profile to text
printer = RubyProf::FlatPrinter.new(result)
printer.print(STDOUT)

# Generate HTML report
printer = RubyProf::GraphHtmlPrinter.new(result)
File.open('profile.html', 'w') { |file| printer.print(file) }

# Using benchmark-ips for micro-benchmarks
require 'benchmark/ips'

Benchmark.ips do |x|
  x.report("String#gsub") { "hello world".gsub(/world/, "universe") }
  x.report("String#sub")  { "hello world".sub(/world/, "universe") }
  x.compare!
end
```

2.4 Ruby Concurrency and Threading

2.4.1 Fiber-based Concurrency

```ruby
# Basic fiber usage
fiber = Fiber.new do |first|
  second = Fiber.yield first + 2
  Fiber.yield second + 3
end

puts fiber.resume(10)    # 12
puts fiber.resume(5)     # 8
puts fiber.resume        # nil (fiber dead)

# Using fibers for lazy evaluation
def fibonacci_fiber
  Fiber.new do
    a, b = 0, 1
    loop do
      Fiber.yield a
      a, b = b, a + b
    end
  end
end

fib = fibonacci_fiber
10.times { puts fib.resume }
```

2.4.2 Thread Safety

```ruby
# Thread-safe counter using Mutex
class ThreadSafeCounter
  def initialize
    @count = 0
    @mutex = Mutex.new
  end

  def increment
    @mutex.synchronize do
      @count += 1
    end
  end

  def value
    @mutex.synchronize { @count }
  end
end

# Using Queue for thread communication
queue = Queue.new

# Producer thread
producer = Thread.new do
  10.times do |i|
    queue.push("Item #{i}")
    sleep(0.1)
  end
  queue.close
end

# Consumer threads
consumers = 3.times.map do |id|
  Thread.new do
    while item = queue.pop
      puts "Consumer #{id} processing #{item}"
      sleep(0.2)
    end
  end
end

[producer, *consumers].each(&:join)
```

2.5 Design Patterns in Ruby

2.5.1 Observer Pattern

```ruby
require 'observer'

class User
  include Observable

  attr_reader :name, :email

  def initialize(name, email)
```

```ruby
    @name = name
    @email = email
  end

  def update_email(new_email)
    @email = new_email
    changed
    notify_observers(self, :email_changed)
  end
end

class EmailNotifier
  def update(user, event)
    case event
    when :email_changed
      puts "Sending email confirmation to #{user.email}"
    end
  end
end

class AuditLogger
  def update(user, event)
    puts "Audit: User #{user.name} - #{event}"
  end
end

# Usage
user = User.new("John", "john@old.com")
user.add_observer(EmailNotifier.new)
user.add_observer(AuditLogger.new)

user.update_email("john@new.com")
```

2.5.2 Strategy Pattern

```ruby
# Payment processing strategy
class PaymentProcessor
  def initialize(strategy)
    @strategy = strategy
  end

  def process(amount)
    @strategy.process_payment(amount)
  end
end

class CreditCardPayment
  def process_payment(amount)
    "Processing $#{amount} via Credit Card"
  end
end

class PayPalPayment
  def process_payment(amount)
    "Processing $#{amount} via PayPal"
  end
end
```

```ruby

class BankTransferPayment
  def process_payment(amount)
    "Processing $#{amount} via Bank Transfer"
  end
end

# Usage
processor = PaymentProcessor.new(CreditCardPayment.new)
puts processor.process(100)

processor = PaymentProcessor.new(PayPalPayment.new)
puts processor.process(100)
```

2.5.3 Decorator Pattern

```ruby
# Simple decorator using modules
module Timestamped
  def save
    puts "Timestamp: #{Time.current}"
    super
  end
end

module Encrypted
  def save
    puts "Encrypting data..."
    super
  end
end

class Document
  def save
    puts "Saving document..."
  end
end

# Usage
doc = Document.new
doc.extend(Timestamped)
doc.extend(Encrypted)
doc.save

# Output:
# Encrypting data...
# Timestamp: 2024-01-01 12:00:00 UTC
# Saving document...
```

2.6 Functional Programming in Ruby

2.6.1 Immutable Data Structures

```ruby
# Creating immutable objects
class ImmutableUser
  attr_reader :name, :email
```

```ruby

  def initialize(name:, email:)
    @name = name.freeze
    @email = email.freeze
    freeze
  end

  def with_email(new_email)
    self.class.new(name: @name, email: new_email)
  end
end

user = ImmutableUser.new(name: "John", email: "john@old.com")
new_user = user.with_email("john@new.com")

puts user.email      # john@old.com
puts new_user.email  # john@new.com
```

2.6.2 Functional Programming Techniques

```ruby
# Higher-order functions
def compose(f, g)
  ->(x) { f.call(g.call(x)) }
end

add_one = ->(x) { x + 1 }
multiply_by_two = ->(x) { x * 2 }

# Compose functions
add_then_multiply = compose(multiply_by_two, add_one)
puts add_then_multiply.call(5)  # 12

# Currying
def curry_add(x)
  ->(y) { x + y }
end

add_five = curry_add(5)
puts add_five.call(10)  # 15

# Using curry method
multiply = ->(x, y) { x * y }
multiply_curried = multiply.curry
double = multiply_curried.call(2)
puts double.call(5)  # 10
```

2.7 Ruby Metaprogramming Deep Dive

2.7.1 Dynamic Class Definition

```ruby
# Creating classes dynamically
def create_model_class(name, attributes)
  klass = Class.new do
    attr_accessor *attributes

```

```ruby
    define_method :initialize do |**args|
      attributes.each do |attr|
        instance_variable_set("@#{attr}", args[attr])
      end
    end

    define_method :to_h do
      attributes.each_with_object({}) do |attr, hash|
        hash[attr] = instance_variable_get("@#{attr}")
      end
    end
  end

  Object.const_set(name, klass)
end

# Usage
create_model_class('Product', [:name, :price, :description])

product = Product.new(
  name: 'Laptop',
  price: 999,
  description: 'Gaming laptop'
)

puts product.to_h
# {:name=>"Laptop", :price=>999, :description=>"Gaming laptop"}
```

2.7.2 Module Builder Pattern

```ruby
module Trackable
  def self.included(base)
    base.extend(ClassMethods)
    base.include(InstanceMethods)
  end

  module ClassMethods
    def track(*methods)
      methods.each do |method|
        alias_method "#{method}_without_tracking", method

        define_method method do |*args, &block|
          puts "Calling #{method} with #{args}"
          result = send("#{method}_without_tracking", *args, &block)
          puts "#{method} returned: #{result}"
          result
        end
      end
    end
  end

  module InstanceMethods
    def tracking_enabled?
      true
    end
  end
end
```

```ruby

class Calculator
  include Trackable

  def add(a, b)
    a + b
  end

  def multiply(a, b)
    a * b
  end

  track :add, :multiply
end

calc = Calculator.new
calc.add(2, 3)      # Logs method call and result
calc.multiply(4, 5) # Logs method call and result
```

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:

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

6.2 Numbers

Integers (whole numbers):

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

Floats (decimal numbers):

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

6.3 Booleans

True or false values:

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

6.4 Nil

Represents "no value":

```ruby
flaws = nil
```

7 Working with Strings

7.1 String Methods

```ruby
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"

# 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... (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: "
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
```

```ruby
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

```
# 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:

```
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"
```

```ruby
    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

```ruby
# 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

```ruby
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|
    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

```
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

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

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

    def has_honors
```

```ruby
        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

```ruby
# 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`:

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

    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:

```
# 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}")
    end

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

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

```ruby
  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 Comprehensive Testing Guide

32.3.1 Testing Philosophy and Types

Ruby testing follows a pyramid structure:

- **Unit Tests** - Test individual methods and classes in isolation
- **Integration Tests** - Test interaction between components
- **System Tests** - Test complete user workflows (end-to-end)

32.3.2 RSpec Framework (Industry Standard)

Installation and Setup:

```ruby
# Add to Gemfile
group :development, :test do
  gem 'rspec-rails'
  gem 'factory_bot_rails'
  gem 'faker'
  gem 'shoulda-matchers'
end

group :test do
  gem 'capybara'
  gem 'webmock'
  gem 'vcr'
  gem 'database_cleaner-active_record'
end

# Initialize RSpec
rails generate rspec:install
```

RSpec Configuration:

```ruby
# spec/spec_helper.rb
RSpec.configure do |config|
  config.expect_with :rspec do |expectations|
    expectations.include_chain_clauses_in_custom_matcher_descriptions =
    true
  end

  config.mock_with :rspec do |mocks|
    mocks.verify_partial_doubles = true
  end

  config.shared_context_metadata_behavior = :apply_to_host_groups
  config.filter_run_when_matching :focus
  config.example_status_persistence_file_path = "spec/examples.txt"
  config.disable_monkey_patching!
  config.warnings = true

  if config.files_to_run.one?
    config.default_formatter = "doc"
  end

  config.profile_examples = 10
  config.order = :random
  Kernel.srand config.seed
end
```

Rails Helper Configuration:

```ruby
# spec/rails_helper.rb
require 'spec_helper'
ENV['RAILS_ENV'] ||= 'test'
require_relative '../config/environment'
abort("Rails is running in production mode!") if Rails.env.production?
require 'rspec/rails'

# Database cleaner setup
require 'database_cleaner/active_record'

```

```ruby
RSpec.configure do |config|
  config.fixture_path = "#{::Rails.root}/spec/fixtures"
  config.use_transactional_fixtures = false
  config.infer_spec_type_from_file_location!
  config.filter_rails_from_backtrace!

  # Database cleaner configuration
  config.before(:suite) do
    DatabaseCleaner.strategy = :transaction
    DatabaseCleaner.clean_with(:truncation)
  end

  config.around(:each) do |example|
    DatabaseCleaner.cleaning do
      example.run
    end
  end
end

# Shoulda matchers configuration
Shoulda::Matchers.configure do |config|
  config.integrate do |with|
    with.test_framework :rspec
    with.library :rails
  end
end
```

32.3.3 Unit Testing Deep Dive

Testing Models:

```ruby
# spec/models/user_spec.rb
require 'rails_helper'

RSpec.describe User, type: :model do
  describe 'validations' do
    it { should validate_presence_of(:email) }
    it { should validate_uniqueness_of(:email).case_insensitive }
    it { should validate_length_of(:password).is_at_least(8) }

    it 'validates email format' do
      user = build(:user, email: 'invalid-email')
      expect(user).not_to be_valid
      expect(user.errors[:email]).to include('is invalid')
    end
  end

  describe 'associations' do
    it { should have_many(:posts).dependent(:destroy) }
    it { should belong_to(:organization) }
  end

  describe 'scopes' do
    let!(:active_user) { create(:user, active: true) }
    let!(:inactive_user) { create(:user, active: false) }

    describe '.active' do
```

```
    it 'returns only active users' do
      expect(User.active).to include(active_user)
      expect(User.active).not_to include(inactive_user)
    end
  end
end

describe 'instance methods' do
  let(:user) { create(:user, first_name: 'John', last_name: 'Doe') }

  describe '#full_name' do
    it 'returns the concatenated first and last name' do
      expect(user.full_name).to eq('John Doe')
    end

    context 'when last name is missing' do
      let(:user) { create(:user, first_name: 'John', last_name: nil)
 }

      it 'returns only the first name' do
        expect(user.full_name).to eq('John')
      end
    end
  end

  describe '#admin?' do
    context 'when user has admin role' do
      let(:user) { create(:user, role: 'admin') }

      it 'returns true' do
        expect(user.admin?).to be true
      end
    end

    context 'when user does not have admin role' do
      let(:user) { create(:user, role: 'member') }

      it 'returns false' do
        expect(user.admin?).to be false
      end
    end
  end
end

describe 'callbacks' do
  describe 'before_save' do
    it 'downcases the email' do
      user = create(:user, email: 'TEST@EXAMPLE.COM')
      expect(user.email).to eq('test@example.com')
    end
  end

  describe 'after_create' do
    it 'sends welcome email' do
      expect {
        create(:user)
      }.to change { ActionMailer::Base.deliveries.count }.by(1)
    end
```

```
    end
  end
end
```

Testing Services and POROs:

```
# app/services/payment_processor.rb
class PaymentProcessor
  def initialize(payment_gateway:, logger: Rails.logger)
    @payment_gateway = payment_gateway
    @logger = logger
  end

  def process(payment_data)
    validate_payment_data(payment_data)

    result = @payment_gateway.charge(payment_data)

    if result.success?
      @logger.info("Payment processed successfully: #{result.transaction_id}")
      { success: true, transaction_id: result.transaction_id }
    else
      @logger.error("Payment failed: #{result.error_message}")
      { success: false, error: result.error_message }
    end
  rescue StandardError => e
    @logger.error("Payment processing error: #{e.message}")
    { success: false, error: 'Payment processing failed' }
  end

  private

  def validate_payment_data(data)
    raise ArgumentError, 'Amount is required' unless data[:amount]
    raise ArgumentError, 'Invalid amount' if data[:amount] <= 0
  end
end

# spec/services/payment_processor_spec.rb
require 'rails_helper'

RSpec.describe PaymentProcessor do
  let(:payment_gateway) { double('PaymentGateway') }
  let(:logger) { double('Logger') }
  let(:processor) { described_class.new(payment_gateway: payment_gateway, logger: logger) }
  let(:payment_data) { { amount: 100, card_number: '1234567890123456' } }

  describe '#process' do
    context 'when payment is successful' do
      let(:gateway_result) { double('Result', success?: true, transaction_id: 'txn_123') }

      before do
        allow(payment_gateway).to receive(:charge).and_return(gateway_result)
        allow(logger).to receive(:info)
```

```
49      end
50
51      it 'returns success result with transaction ID' do
52        result = processor.process(payment_data)
53
54        expect(result).to eq({
55          success: true,
56          transaction_id: 'txn_123'
57        })
58      end
59
60      it 'logs successful payment' do
61        processor.process(payment_data)
62
63        expect(logger).to have_received(:info)
64          .with("Payment processed successfully: txn_123")
65      end
66
67      it 'calls payment gateway with correct data' do
68        processor.process(payment_data)
69
70        expect(payment_gateway).to have_received(:charge).with(
    payment_data)
71      end
72    end
73
74    context 'when payment fails' do
75      let(:gateway_result) { double('Result', success?: false,
    error_message: 'Insufficient funds') }
76
77      before do
78        allow(payment_gateway).to receive(:charge).and_return(
    gateway_result)
79        allow(logger).to receive(:error)
80      end
81
82      it 'returns failure result with error message' do
83        result = processor.process(payment_data)
84
85        expect(result).to eq({
86          success: false,
87          error: 'Insufficient funds'
88        })
89      end
90
91      it 'logs payment failure' do
92        processor.process(payment_data)
93
94        expect(logger).to have_received(:error)
95          .with("Payment failed: Insufficient funds")
96      end
97    end
98
99    context 'when payment data is invalid' do
100     before do
101       allow(logger).to receive(:error)
102     end
103
```

```
      it 'raises error for missing amount' do
        invalid_data = payment_data.except(:amount)

        result = processor.process(invalid_data)

        expect(result).to eq({
          success: false,
          error: 'Payment processing failed'
        })
      end

      it 'raises error for zero amount' do
        invalid_data = payment_data.merge(amount: 0)

        result = processor.process(invalid_data)

        expect(result).to eq({
          success: false,
          error: 'Payment processing failed'
        })
      end
    end

    context 'when gateway raises an exception' do
      before do
        allow(payment_gateway).to receive(:charge).and_raise(StandardError, 'Network error')
        allow(logger).to receive(:error)
      end

      it 'handles the exception gracefully' do
        result = processor.process(payment_data)

        expect(result).to eq({
          success: false,
          error: 'Payment processing failed'
        })
      end

      it 'logs the error' do
        processor.process(payment_data)

        expect(logger).to have_received(:error)
          .with("Payment processing error: Network error")
      end
    end
  end
end
```

32.3.4 Integration Testing

Testing Controllers:

```
# spec/controllers/users_controller_spec.rb
require 'rails_helper'

RSpec.describe UsersController, type: :controller do
```

```
  let(:user) { create(:user) }
  let(:admin) { create(:user, role: 'admin') }

  describe 'GET #index' do
    context 'when user is admin' do
      before { sign_in admin }

      it 'returns success status' do
        get :index
        expect(response).to have_http_status(:success)
      end

      it 'assigns all users' do
        user1 = create(:user)
        user2 = create(:user)

        get :index

        expect(assigns(:users)).to include(user1, user2, admin)
      end
    end

    context 'when user is not admin' do
      before { sign_in user }

      it 'redirects to root path' do
        get :index
        expect(response).to redirect_to(root_path)
      end

      it 'sets flash error message' do
        get :index
        expect(flash[:alert]).to eq('Access denied')
      end
    end

    context 'when user is not signed in' do
      it 'redirects to sign in page' do
        get :index
        expect(response).to redirect_to(new_user_session_path)
      end
    end
  end

  describe 'POST #create' do
    let(:valid_params) { { user: attributes_for(:user) } }
    let(:invalid_params) { { user: attributes_for(:user, email: '') } }

    context 'with valid parameters' do
      it 'creates a new user' do
        expect {
          post :create, params: valid_params
        }.to change(User, :count).by(1)
      end

      it 'redirects to user page' do
        post :create, params: valid_params
        expect(response).to redirect_to(user_path(User.last))
```

```
      end

      it 'sets success flash message' do
        post :create, params: valid_params
        expect(flash[:notice]).to eq('User created successfully')
      end
    end

    context 'with invalid parameters' do
      it 'does not create a user' do
        expect {
          post :create, params: invalid_params
        }.not_to change(User, :count)
      end

      it 'renders new template' do
        post :create, params: invalid_params
        expect(response).to render_template(:new)
      end

      it 'assigns user with errors' do
        post :create, params: invalid_params
        expect(assigns(:user).errors).not_to be_empty
      end
    end
  end
end
```

Testing API Endpoints:

```
# spec/requests/api/v1/users_spec.rb
require 'rails_helper'

RSpec.describe 'API::V1::Users', type: :request do
  let(:user) { create(:user) }
  let(:auth_headers) { { 'Authorization' => "Bearer #{user.auth_token}" } }

  describe 'GET /api/v1/users' do
    let!(:users) { create_list(:user, 3) }

    context 'with valid authentication' do
      before { get '/api/v1/users', headers: auth_headers }

      it 'returns success status' do
        expect(response).to have_http_status(:ok)
      end

      it 'returns users data' do
        json_response = JSON.parse(response.body)
        expect(json_response['users'].length).to eq(4) # 3 created + 1 authenticated user
      end

      it 'returns correct user structure' do
        json_response = JSON.parse(response.body)
        user_data = json_response['users'].first

        expect(user_data).to include(
```

```
28            'id',
29            'email',
30            'first_name',
31            'last_name',
32            'created_at'
33          )
34          expect(user_data).not_to include('password_digest')
35        end
36      end
37
38      context 'without authentication' do
39        before { get '/api/v1/users' }
40
41        it 'returns unauthorized status' do
42          expect(response).to have_http_status(:unauthorized)
43        end
44
45        it 'returns error message' do
46          json_response = JSON.parse(response.body)
47          expect(json_response['error']).to eq('Authentication required')
48        end
49      end
50    end
51
52    describe 'POST /api/v1/users' do
53      let(:valid_params) do
54        {
55          user: {
56            email: 'test@example.com',
57            password: 'password123',
58            first_name: 'Test',
59            last_name: 'User'
60          }
61        }
62      end
63
64      context 'with valid parameters' do
65        it 'creates a new user' do
66          expect {
67            post '/api/v1/users', params: valid_params
68          }.to change(User, :count).by(1)
69        end
70
71        it 'returns created status' do
72          post '/api/v1/users', params: valid_params
73          expect(response).to have_http_status(:created)
74        end
75
76        it 'returns user data' do
77          post '/api/v1/users', params: valid_params
78          json_response = JSON.parse(response.body)
79
80          expect(json_response['user']['email']).to eq('test@example.com'
    )
81          expect(json_response['user']['first_name']).to eq('Test')
82        end
83      end
84
```

```
    context 'with invalid parameters' do
      let(:invalid_params) { { user: { email: '', password: '123' } } }

      it 'does not create a user' do
        expect {
          post '/api/v1/users', params: invalid_params
        }.not_to change(User, :count)
      end

      it 'returns unprocessable entity status' do
        post '/api/v1/users', params: invalid_params
        expect(response).to have_http_status(:unprocessable_entity)
      end

      it 'returns validation errors' do
        post '/api/v1/users', params: invalid_params
        json_response = JSON.parse(response.body)

        expect(json_response['errors']).to include('email')
        expect(json_response['errors']).to include('password')
      end
    end
  end
end
```

32.3.5 System/Feature Testing with Capybara

```
# spec/system/user_registration_spec.rb
require 'rails_helper'

RSpec.describe 'User Registration', type: :system do
  before do
    driven_by(:selenium_chrome_headless)
  end

  describe 'successful registration' do
    it 'allows user to create an account' do
      visit new_user_registration_path

      fill_in 'Email', with: 'test@example.com'
      fill_in 'Password', with: 'password123'
      fill_in 'Password confirmation', with: 'password123'
      fill_in 'First name', with: 'John'
      fill_in 'Last name', with: 'Doe'

      click_button 'Sign up'

      expect(page).to have_content('Welcome! You have signed up
    successfully.')
      expect(page).to have_current_path(root_path)
      expect(page).to have_content('John Doe')
    end

    it 'sends welcome email' do
      visit new_user_registration_path

      fill_in 'Email', with: 'test@example.com'
```

```
      fill_in 'Password', with: 'password123'
      fill_in 'Password confirmation', with: 'password123'
      fill_in 'First name', with: 'John'
      fill_in 'Last name', with: 'Doe'

      expect {
        click_button 'Sign up'
      }.to change { ActionMailer::Base.deliveries.count }.by(1)

      welcome_email = ActionMailer::Base.deliveries.last
      expect(welcome_email.to).to include('test@example.com')
      expect(welcome_email.subject).to eq('Welcome to Our Platform')
    end
  end

  describe 'validation errors' do
    it 'shows errors for invalid input' do
      visit new_user_registration_path

      fill_in 'Email', with: 'invalid-email'
      fill_in 'Password', with: '123'
      click_button 'Sign up'

      expect(page).to have_content('Email is invalid')
      expect(page).to have_content('Password is too short')
      expect(page).to have_current_path(user_registration_path)
    end

    it 'shows error for duplicate email' do
      existing_user = create(:user, email: 'test@example.com')

      visit new_user_registration_path

      fill_in 'Email', with: 'test@example.com'
      fill_in 'Password', with: 'password123'
      fill_in 'Password confirmation', with: 'password123'
      click_button 'Sign up'

      expect(page).to have_content('Email has already been taken')
    end
  end

  describe 'JavaScript interactions' do
    it 'validates email format in real-time', js: true do
      visit new_user_registration_path

      email_field = find('#user_email')
      email_field.fill_in(with: 'invalid-email')
      email_field.trigger('blur')

      expect(page).to have_css('.field_with_errors')
      expect(page).to have_content('Please enter a valid email address'
  )
    end

    it 'shows password strength indicator', js: true do
      visit new_user_registration_path

```

```
      password_field = find('#user_password')
      password_field.fill_in(with: 'weak')

      expect(page).to have_css('.password-strength.weak')

      password_field.fill_in(with: 'StrongPassword123!')

      expect(page).to have_css('.password-strength.strong')
    end
  end
end
```

32.3.6 Test Data Management with FactoryBot

```
# spec/factories/users.rb
FactoryBot.define do
  factory :user do
    sequence(:email) { |n| "user#{n}@example.com" }
    first_name { Faker::Name.first_name }
    last_name { Faker::Name.last_name }
    password { 'password123' }
    active { true }
    role { 'member' }

    trait :admin do
      role { 'admin' }
    end

    trait :inactive do
      active { false }
    end

    trait :with_posts do
      after(:create) do |user|
        create_list(:post, 3, user: user)
      end
    end

    factory :admin_user, traits: [:admin]
    factory :inactive_user, traits: [:inactive]
  end
end

# spec/factories/posts.rb
FactoryBot.define do
  factory :post do
    title { Faker::Lorem.sentence }
    content { Faker::Lorem.paragraphs(number: 3).join("\n\n") }
    published { true }
    association :user

    trait :draft do
      published { false }
    end

    trait :with_comments do
      after(:create) do |post|
```

```
        create_list(:comment, 5, post: post)
      end
    end

    factory :draft_post, traits: [:draft]
    factory :post_with_comments, traits: [:with_comments]
  end
end

# Usage examples:
user = create(:user)                    # Creates user with default attributes
admin = create(:admin_user)             # Creates user with admin role
user_with_posts = create(:user, :with_posts)  # Creates user with 3 posts
users = create_list(:user, 5)          # Creates 5 users

# Build vs Create
built_user = build(:user)               # Builds object without saving
created_user = create(:user)            # Creates and saves to database
attributes = attributes_for(:user)      # Returns hash of attributes
```

32.3.7 Mocking and Stubbing External Services

```
# Using WebMock for HTTP requests
# spec/support/webmock.rb
require 'webmock/rspec'

RSpec.configure do |config|
  config.before(:each) do
    WebMock.disable_net_connect!(allow_localhost: true)
  end
end

# Example usage in specs
RSpec.describe ExternalApiService do
  describe '#fetch_user_data' do
    let(:service) { described_class.new }
    let(:user_id) { 123 }
    let(:api_response) do
      {
        id: user_id,
        name: 'John Doe',
        email: 'john@example.com'
      }.to_json
    end

    before do
      stub_request(:get, "https://api.example.com/users/#{user_id}")
        .with(headers: { 'Authorization' => 'Bearer token123' })
        .to_return(
          status: 200,
          body: api_response,
          headers: { 'Content-Type' => 'application/json' }
        )
    end

```

```ruby
    it 'fetches user data from external API' do
      result = service.fetch_user_data(user_id)

      expect(result[:name]).to eq('John Doe')
      expect(result[:email]).to eq('john@example.com')
    end

    context 'when API returns error' do
      before do
        stub_request(:get, "https://api.example.com/users/#{user_id}")
          .to_return(status: 404, body: '{"error": "User not found"}')
      end

      it 'handles API errors gracefully' do
        expect {
          service.fetch_user_data(user_id)
        }.to raise_error(ExternalApiService::UserNotFoundError)
      end
    end
  end
end

# Using VCR for recording real HTTP interactions
# spec/support/vcr.rb
require 'vcr'

VCR.configure do |config|
  config.cassette_library_dir = 'spec/vcr_cassettes'
  config.hook_into :webmock
  config.ignore_localhost = true
  config.configure_rspec_metadata!

  # Filter sensitive data
  config.filter_sensitive_data('<API_KEY>') { ENV['API_KEY'] }
  config.filter_sensitive_data('<AUTH_TOKEN>') { ENV['AUTH_TOKEN'] }
end

# Usage in specs
RSpec.describe WeatherService, vcr: true do
  describe '#current_weather' do
    it 'fetches current weather data' do
      VCR.use_cassette('weather_service/current_weather') do
        weather = WeatherService.new.current_weather('New York')
        expect(weather[:temperature]).to be_a(Numeric)
        expect(weather[:condition]).to be_present
      end
    end
  end
end
```

32.3.8 Testing Best Practices

Test Organization:

```ruby
# Good test structure
RSpec.describe SomeClass do
  # Use let for test data setup
```

```
  let(:user) { create(:user) }
  let(:service) { described_class.new(user) }

  # Group related tests
  describe '#public_method' do
    context 'when condition A' do
      it 'does something specific' do
        # Arrange
        setup_data

        # Act
        result = service.public_method

        # Assert
        expect(result).to eq(expected_value)
      end
    end

    context 'when condition B' do
      it 'does something else' do
        # Test implementation
      end
    end
  end

  describe 'private methods' do
    # Only test private methods if absolutely necessary
    # Usually through public interface
  end
end
```

Common Testing Patterns:

```
# Testing callbacks
it 'triggers callback' do
  expect(user).to receive(:send_welcome_email)
  user.save
end

# Testing state changes
it 'changes user status' do
  expect {
    service.activate_user
  }.to change(user, :status).from('pending').to('active')
end

# Testing side effects
it 'creates audit log' do
  expect {
    service.delete_user
  }.to change(AuditLog, :count).by(1)
end

# Testing exception handling
it 'raises specific error' do
  expect {
    service.invalid_operation
  }.to raise_error(ServiceError, 'Operation not allowed')
end
```

```ruby

# Testing with time
it 'sets timestamp correctly' do
  freeze_time do
    service.mark_completed
    expect(task.completed_at).to eq(Time.current)
  end
end
```

32.3.9 Minitest Alternative

For those preferring Minitest over RSpec:

```ruby
# test/models/user_test.rb
require 'test_helper'

class UserTest < ActiveSupport::TestCase
  def setup
    @user = users(:john)
  end

  test 'should be valid with valid attributes' do
    assert @user.valid?
  end

  test 'should require email' do
    @user.email = nil
    assert_not @user.valid?
    assert_includes @user.errors[:email], "can't be blank"
  end

  test 'should return full name' do
    @user.first_name = 'John'
    @user.last_name = 'Doe'
    assert_equal 'John Doe', @user.full_name
  end

  test 'should create user with factory' do
    user = create(:user)
    assert user.persisted?
    assert user.valid?
  end
end

# test/integration/user_flows_test.rb
class UserFlowsTest < ActionDispatch::IntegrationTest
  test 'user registration flow' do
    get new_user_registration_path
    assert_response :success

    post user_registration_path, params: {
      user: {
        email: 'test@example.com',
        password: 'password123',
        password_confirmation: 'password123'
      }
    }
```

```
45
46      assert_redirected_to root_path
47      follow_redirect!
48      assert_match 'Welcome', response.body
49    end
50  end
```

32.3.10 Debugging with Pry

```
# 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

```
# .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

```
# 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
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)

```
# 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

```
# 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

```
require 'benchmark'

# Compare different approaches
```

```ruby
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

```ruby
# 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/`