Tia Koenig

Terminal Application – T1A3

Overview

My app is called “What to cook?”. Its main purpose is to allow the user to input a list of ingredients that will then search the recipes database and output recipes that contain the ingredients that were input by the user.

Additional features will include adding recipes to the recipes.csv file.

Deleting recipes from the recipes.csv file.

Adding recipes to the favourites.csv file.

Time permitting I’d like this to be unique to each user by having an accounts feature.

Find recipes feature

A walkthrough of how the find recipes feature works,

The user navigates through the menu

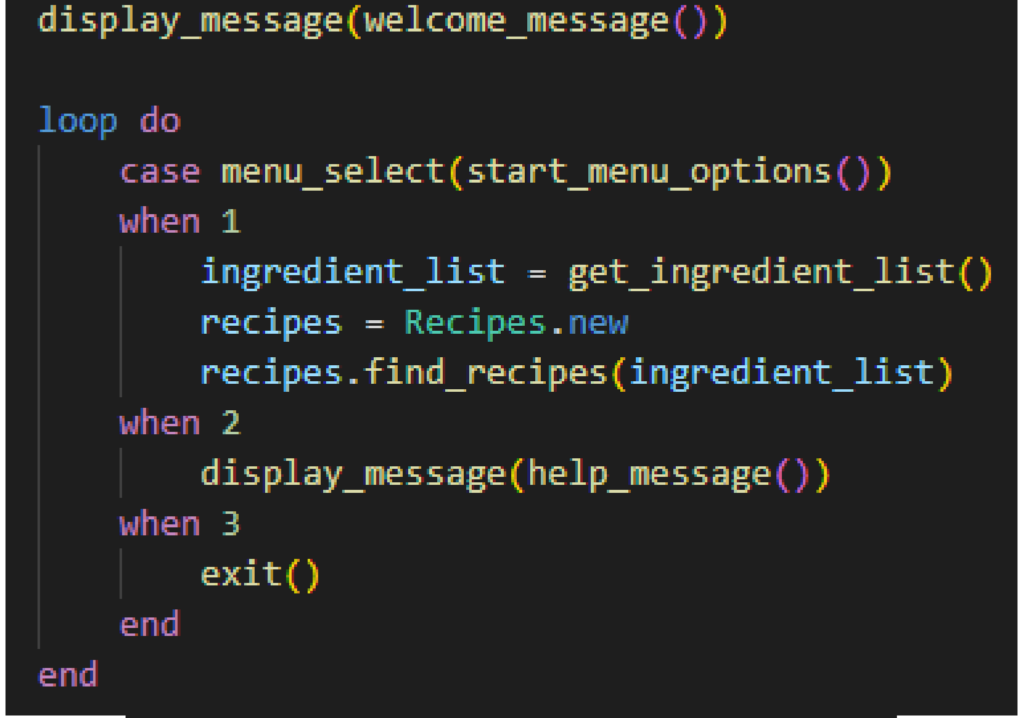
They select find a recipe

They are prompted to enter an ingredient

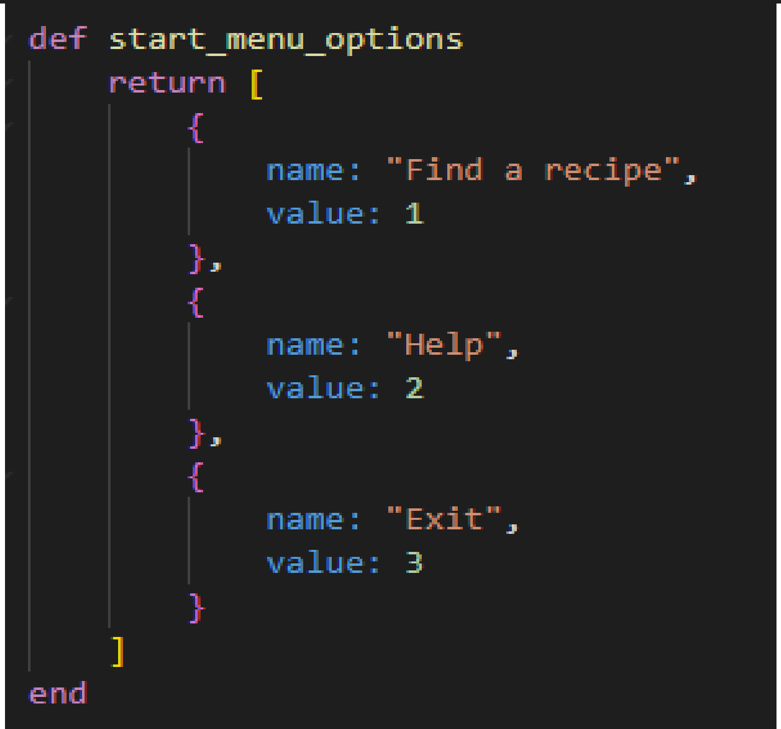
Once an ingredient is entered they will be asked if they have any other ingredients with a yes/no input option, if yes, they will be asked to input another ingredient, if no, the application will output recipes that include their ingredient/s

Find Recipe feature, logic breakdown

- Looping over the case statement, if the return value of menu_select is 1, it will fire off the get_ingredient_list method and store its value in the ingredient_list variable



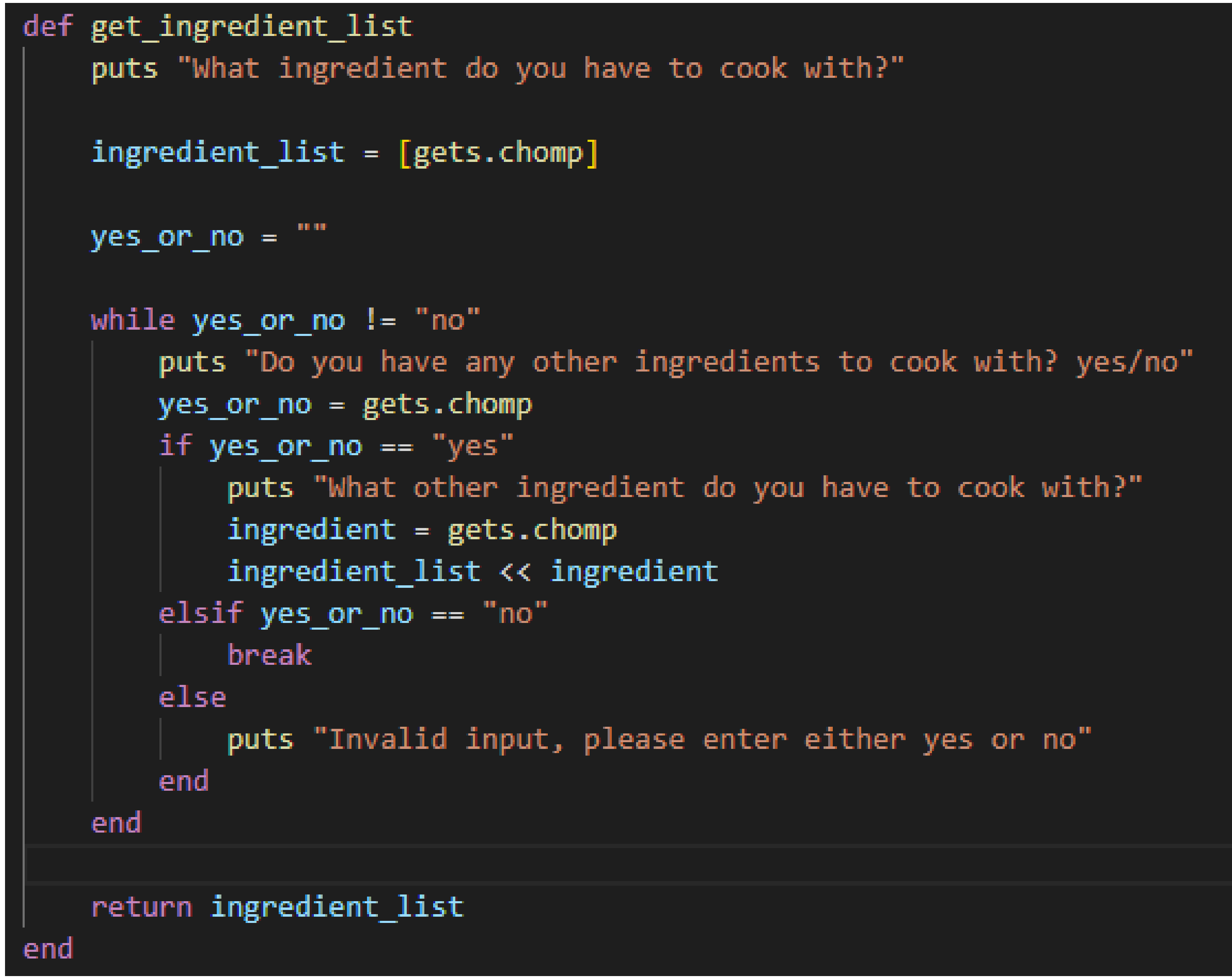
```
display_message(welcome_message())

loop do
    case menu_select(start_menu_options())
    when 1
        ingredient_list = get_ingredient_list()
        recipes = Recipes.new
        recipes.find_recipes(ingredient_list)
    when 2
        display_message(help_message())
    when 3
        exit()
    end
end
```



```
def start_menu_options
    return [
        {
            name: "Find a recipe",
            value: 1
        },
        {
            name: "Help",
            value: 2
        },
        {
            name: "Exit",
            value: 3
        }
    ]
end
```

Find Recipe feature, logic breakdown

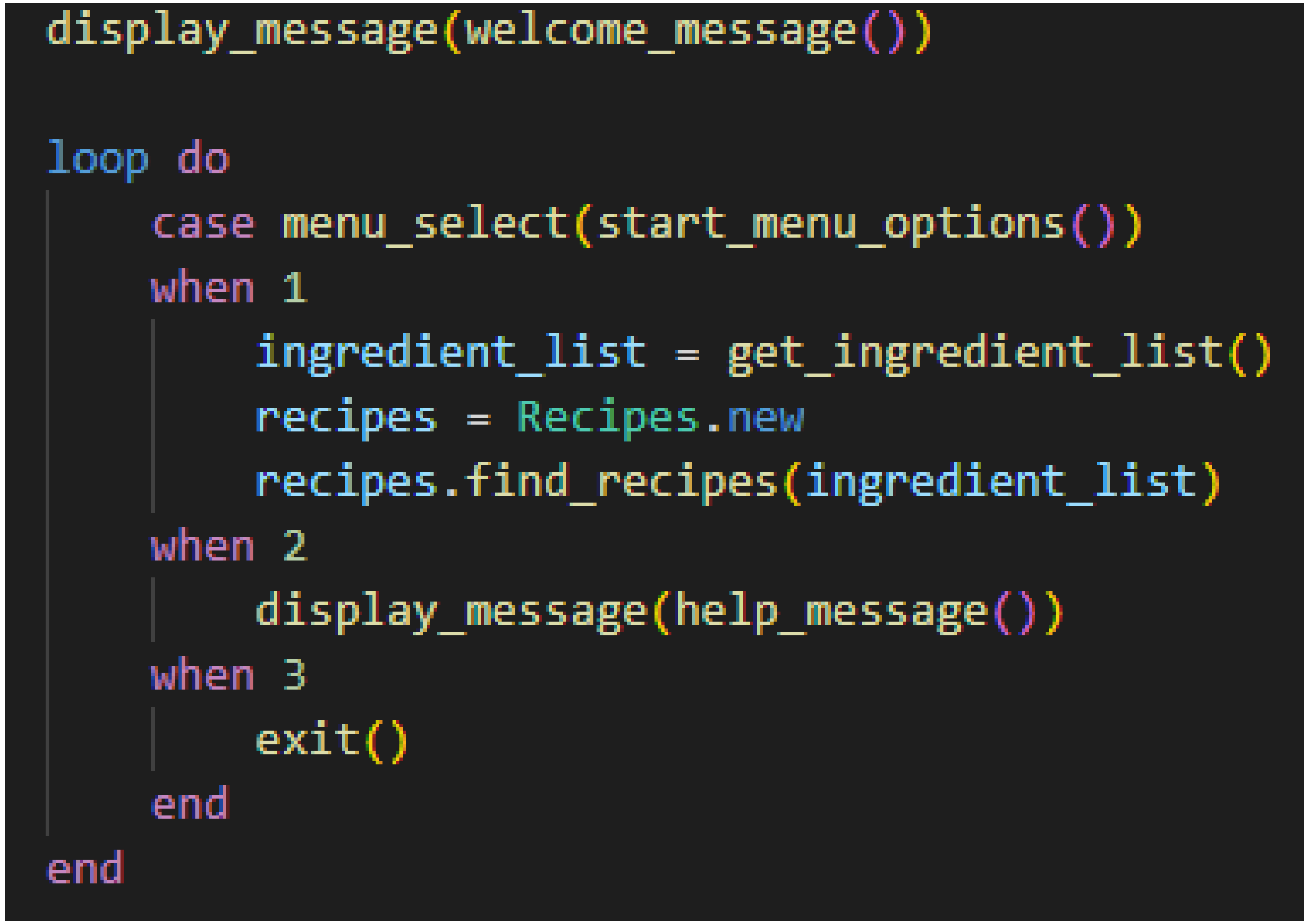
- Prompts the user to input an ingredient and stores the ingredient in an array called ingredient_list
- Then using a while loop to continuously prompt the user to input ingredients they have to cook with until they enter no
- For each iteration of the loop I am pushing the users input into the ingredient_list array
- Have implemented error handling in case the user does not enter yes or no when prompted
- After the user enters “no”, the loop breaks and will return the ingredient_list array



```ruby
def get_ingredient_list
    puts "What ingredient do you have to cook with?"

    ingredient_list = [gets.chomp]

    yes_or_no = ""

    while yes_or_no != "no"
        puts "Do you have any other ingredients to cook with? yes/no"
        yes_or_no = gets.chomp
        if yes_or_no == "yes"
            puts "What other ingredient do you have to cook with?"
            ingredient = gets.chomp
            ingredient_list << ingredient
        elsif yes_or_no == "no"
            break
        else
            puts "Invalid input, please enter either yes or no"
        end
    end

    return ingredient_list
end
```

Find Recipe feature, logic breakdown

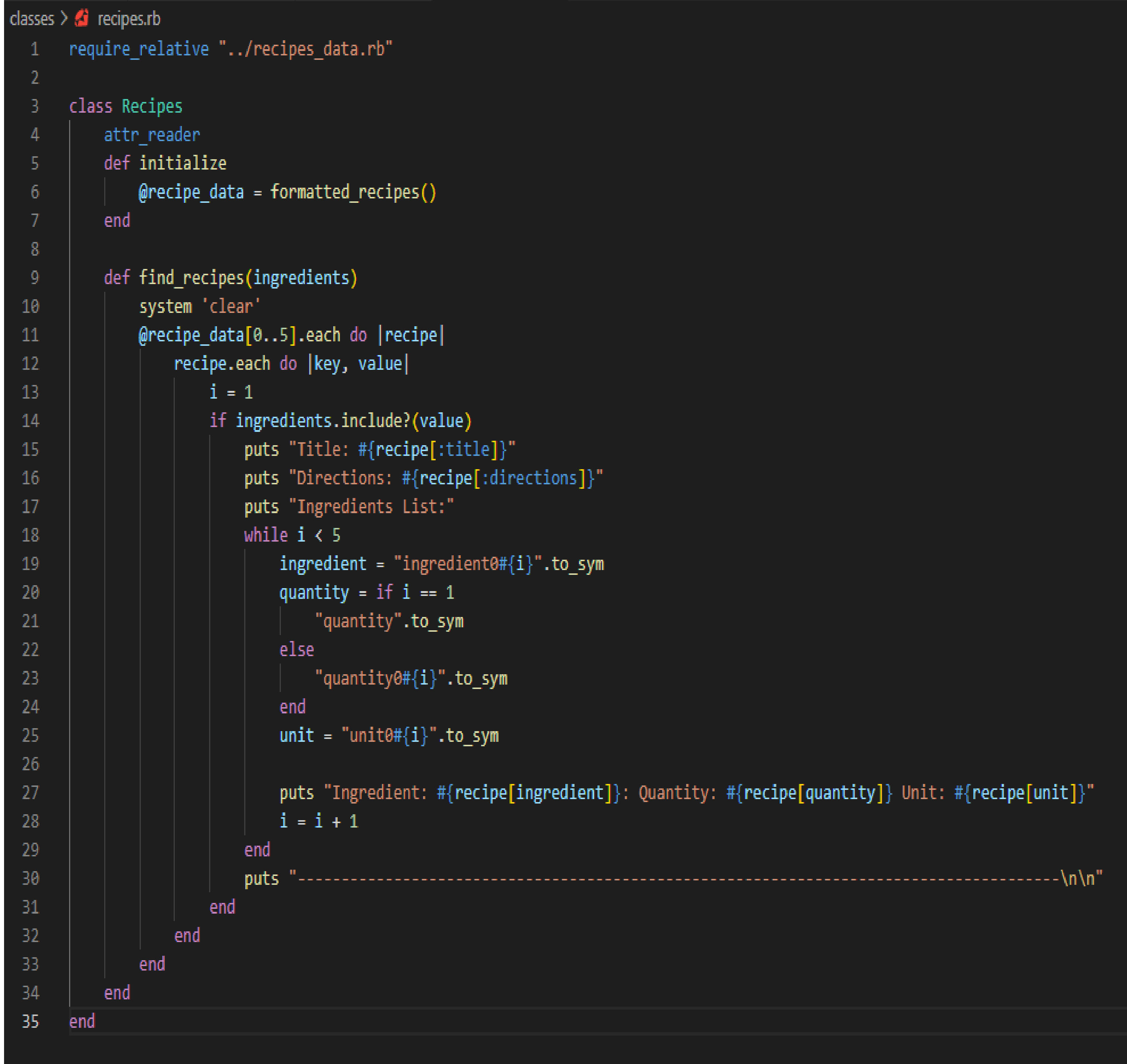
- After the get_ingredient_list method has finished, a new instance of the Recipes class is created and stored in the variable recipes.
- I'm then calling the class method find_recipes and passing in the ingredient_list array.



```
display_message(welcome_message())

loop do
    case menu_select(start_menu_options())
    when 1
        ingredient_list = get_ingredient_list()
        recipes = Recipes.new
        recipes.find_recipes(ingredient_list)
    when 2
        display_message(help_message())
    when 3
        exit()
    end
end
```

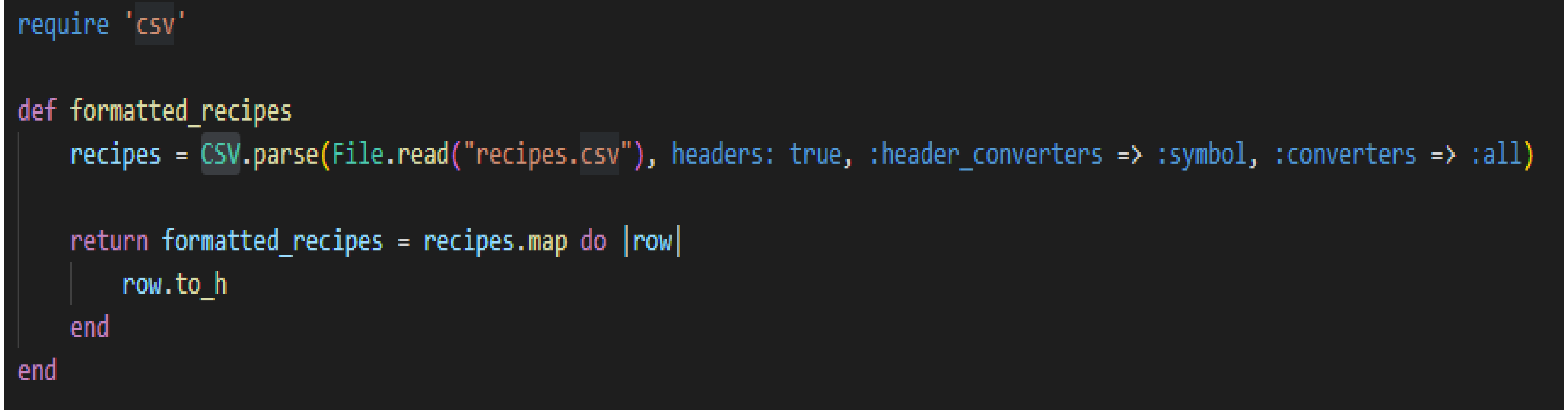
Find Recipe feature, logic breakdown

- For testing purposes, I am iterating over the first 5 recipes because there are 200+ in the CSV file.
- For each recipe in the recipe_data array I am iterating over each recipe.
- I am then checking if the ingredients array includes the recipe value.
- If it does I am outputting the recipe details.
- A complication I came across was that the CSV file has up to 20 ingredients but doesn't have values for each ingredient key.
- Still debugging to try and fix this problem, until then I am just getting the first 5 ingredients.



classes > recipes.rb

```ruby
require_relative "../recipes_data.rb"

class Recipes
    attr_reader
    def initialize
        @recipe_data = formatted_recipes()
    end

    def find_recipes(ingredients)
        system 'clear'
        @recipe_data[0..5].each do |recipe|
            recipe.each do |key, value|
                i = 1
                if ingredients.include?(value)
                    puts "Title: #{recipe[:title]}"
                    puts "Directions: #{recipe[:directions]}"
                    puts "Ingredients List:"
                    while i < 5
                        ingredient = "ingredient0#{i}".to_sym
                        quantity = if i == 1
                            "quantity".to_sym
                        else
                            "quantity0#{i}".to_sym
                        end
                        unit = "unit0#{i}".to_sym

                        puts "Ingredient: #{recipe[ingredient]}: Quantity: #{recipe[quantity]} Unit: #{recipe[unit]}"
                        i = i + 1
                    end
                    puts "---------------------------------------------------------------------------------------\n\n"
                end
            end
        end
    end
end
```



```ruby
require 'csv'

def formatted_recipes
    recipes = CSV.parse(File.read("recipes.csv"), headers: true, :header_converters => :symbol, :converters => :all)

    return formatted_recipes = recipes.map do |row|
        row.to_h
    end
end
```

Formatting data from CSV file, logic breakdown

- In order to format the CSV data I am first parsing the recipes.csv file, converting the headers to symbols and storing the value in the recipes variable.

- I am then mapping over the recipes array and for each row I am converting them to hashes, and returning the formatted_recipes array.

Challenges

At first I wanted to use a recipe API but the free version of the API I found limited me to 50 requests per day. Being a junior this wasn't feasible as I am constantly testing my code and the requests quickly added up.

Instead I chose to use a CSV file I found on GitHub, however, the data is structured quite inefficiently.

Ethical issues

The initial API I was using had a free tier and the CSV file was open source, so I have yet to run in to any ethical issues.

Favourite parts

Favourite part so far has been testing and seeing the code output what I want/expect it to do.

Gems

- tty-prompt
- rainbow