T1A3

Terminal application.

Jacob Solomon. September 2020.

Terminal application

Solving a problem

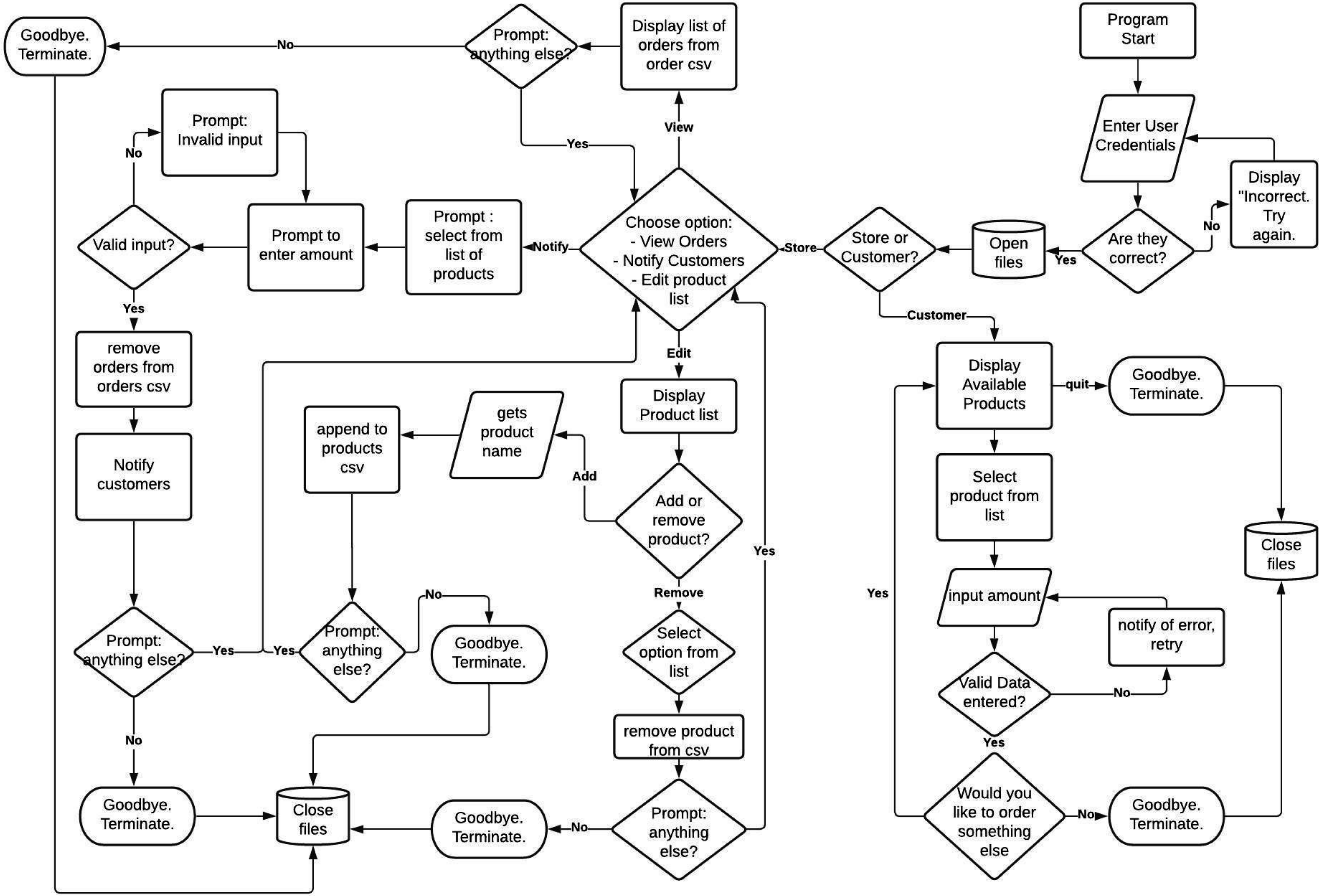
- Ordering/placing reservations for products with businesses is annoying/inefficient.

- How can this be better?

- What can i do?

Build a terminal app!

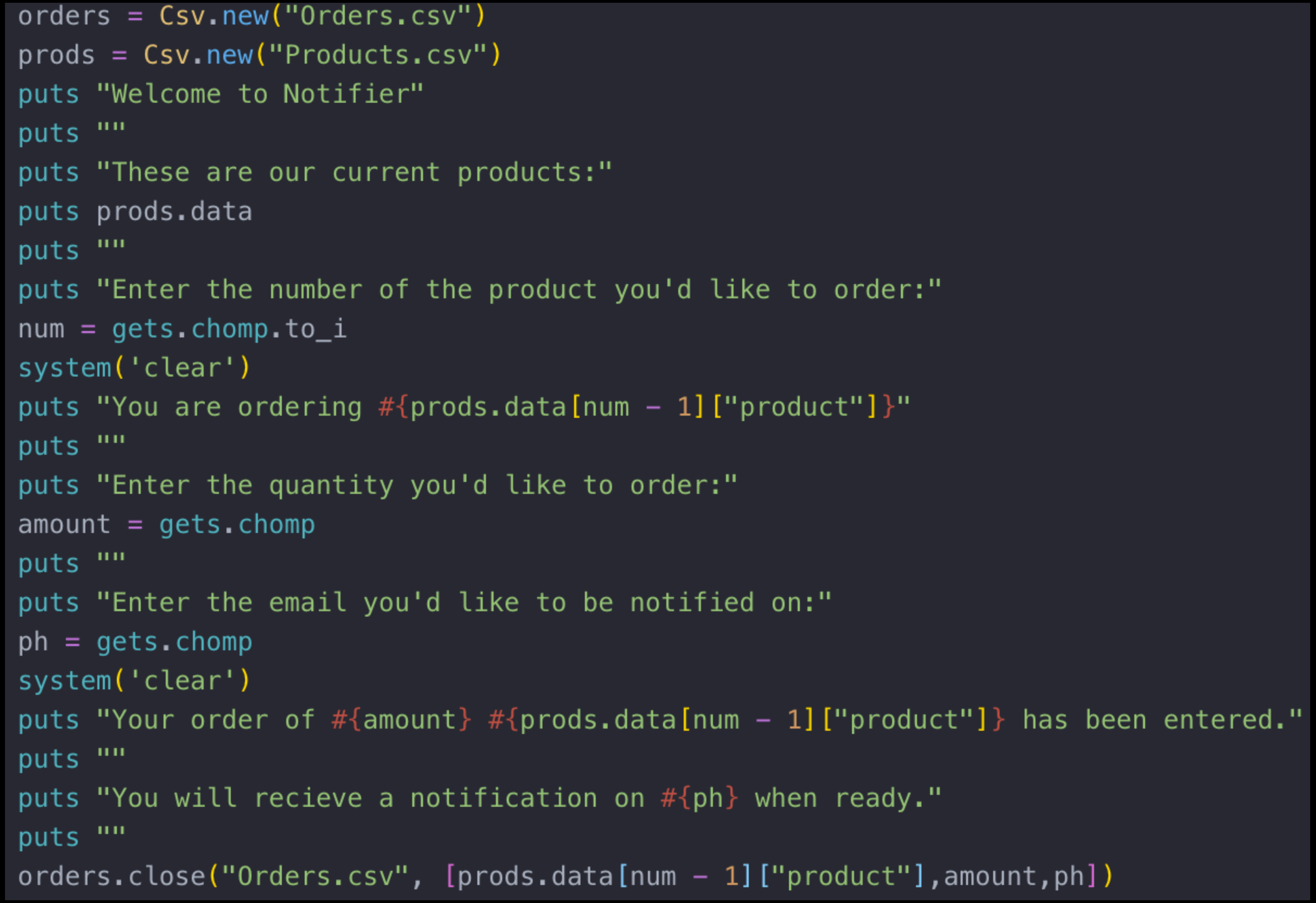
Flowchart


Program Start
Enter User Credentials
Are they correct?
Yes
No
Display "Incorrect. Try again.
Open files
Store or Customer?
Store
Customer
Choose option:
- View Orders
- Notify Customers
- Edit product list
View
Display list of orders from order csv
Prompt: anything else?
No
Yes
Goodbye. Terminate.
Notify
Prompt : select from list of products
Prompt to enter amount
Valid input?
No
Prompt: Invalid input
Yes
remove orders from orders csv
Notify customers
Prompt: anything else?
Yes
No
Goodbye. Terminate.
Close files
Edit
Display Product list
Add or remove product?
Add
gets product name
append to products csv
Prompt: anything else?
Yes
No
Goodbye. Terminate.
Remove
Select option from list
remove product from csv
Prompt: anything else?
Yes
No
Goodbye. Terminate.
Close files
Display Available Products
quit
Goodbye. Terminate.
Select product from list
input amount
Valid Data entered?
No
notify of error, retry
Yes
Would you like to order something else
Yes
No
Goodbye. Terminate.
Close files

Functions

customer place order

main file



```ruby
orders = Csv.new("Orders.csv")
prods = Csv.new("Products.csv")
puts "Welcome to Notifier"
puts ""
puts "These are our current products:"
puts prods.data
puts ""
puts "Enter the number of the product you'd like to order:"
num = gets.chomp.to_i
system('clear')
puts "You are ordering #{prods.data[num - 1]["product"]}"
puts ""
puts "Enter the quantity you'd like to order:"
amount = gets.chomp
puts ""
puts "Enter the email you'd like to be notified on:"
ph = gets.chomp
system('clear')
puts "Your order of #{amount} #{prods.data[num - 1]["product"]} has been entered."
puts ""
puts "You will recieve a notification on #{ph} when ready."
puts ""
orders.close("Orders.csv", [prods.data[num - 1]["product"],amount,ph])
```

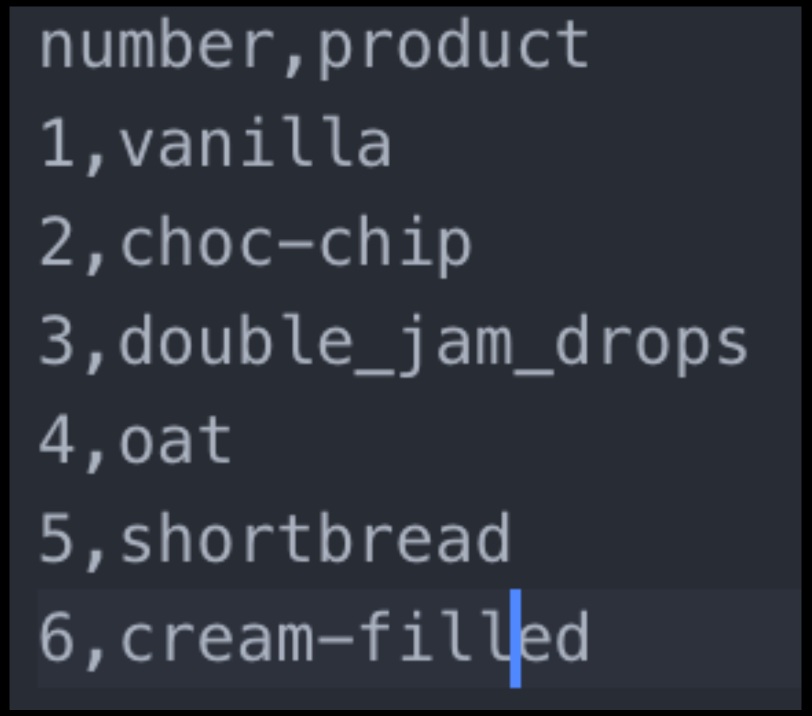
takes input

takes input

takes input

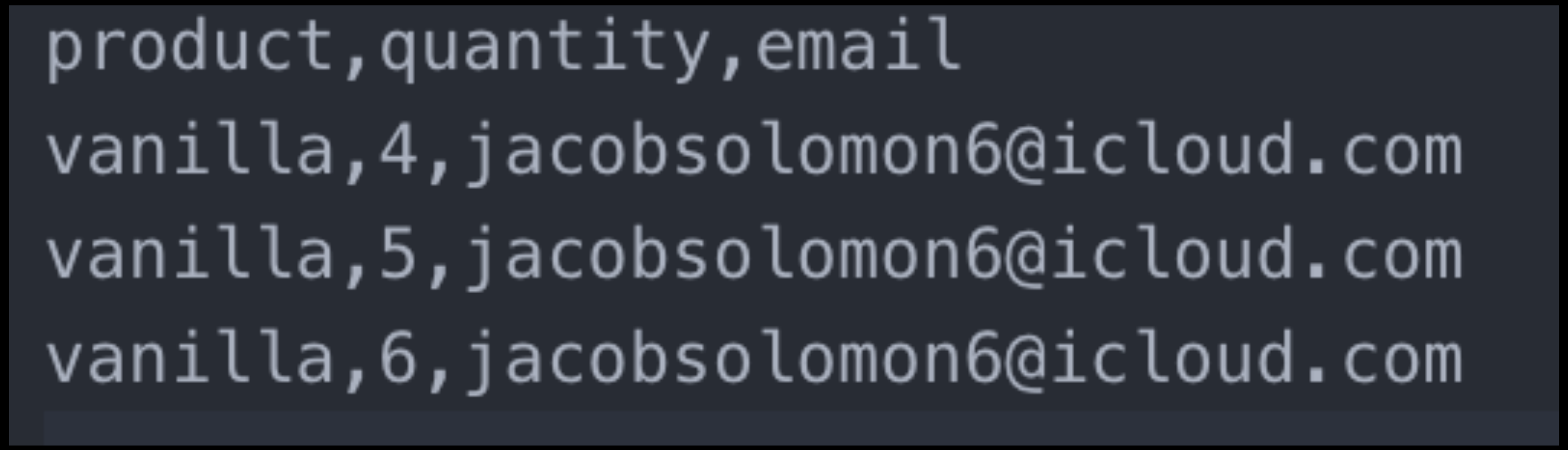
push to csv

product csv



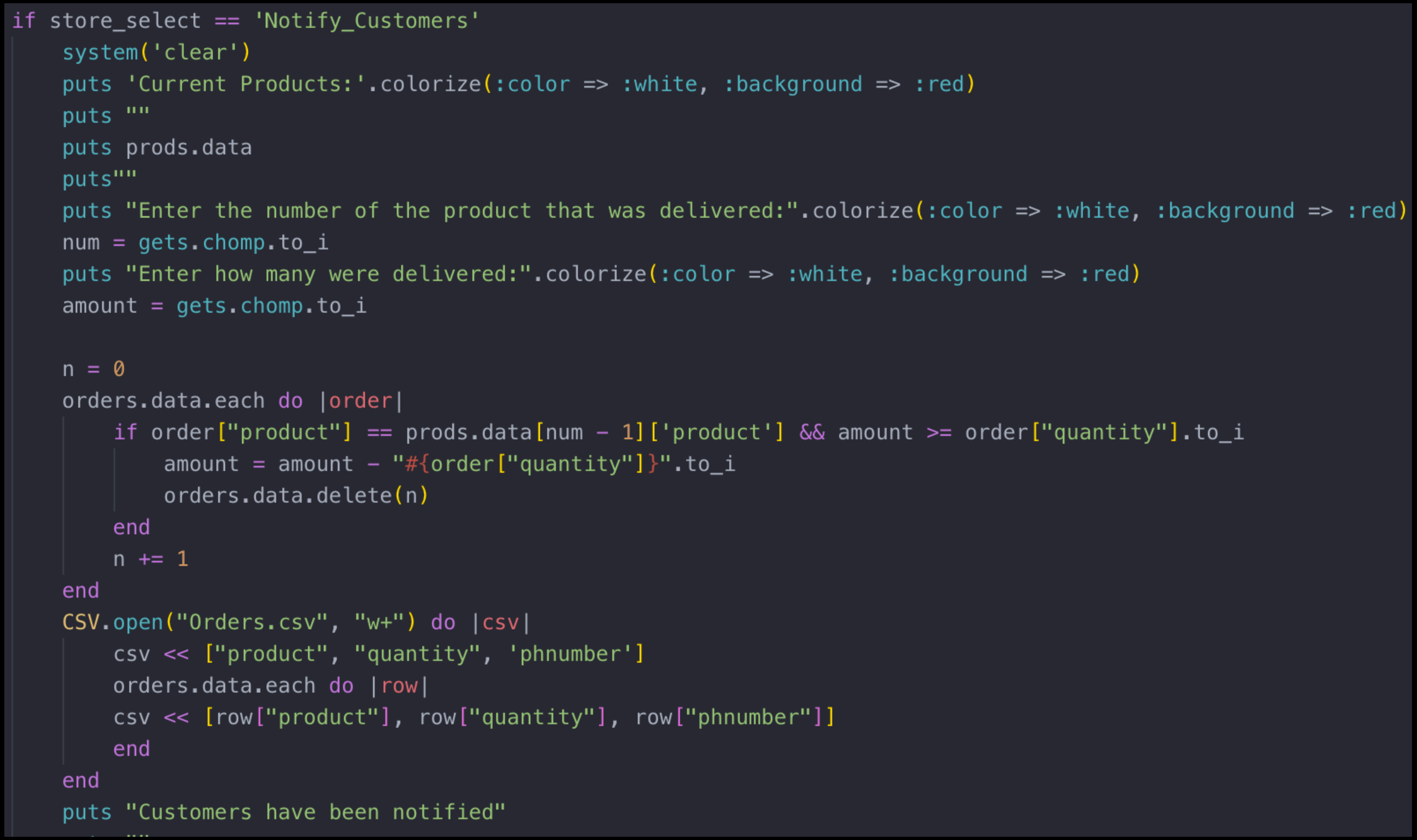
```
number,product
1,vanilla
2,choc-chip
3,double_jam_drops
4,oat
5,shortbread
6,cream-filled
```

orders csv



```
product,quantity,email
vanilla,4,jacobsolomon6@icloud.com
vanilla,5,jacobsolomon6@icloud.com
vanilla,6,jacobsolomon6@icloud.com
```

Notify function



```
if store_select == 'Notify_Customers'
    system('clear')
    puts 'Current Products:'.colorize(:color => :white, :background => :red)
    puts ""
    puts prods.data
    puts""
    puts "Enter the number of the product that was delivered:".colorize(:color => :white, :background => :red)
    num = gets.chomp.to_i
    puts "Enter how many were delivered:".colorize(:color => :white, :background => :red)
    amount = gets.chomp.to_i

    n = 0
    orders.data.each do |order|
        if order["product"] == prods.data[num - 1]['product'] && amount >= order["quantity"].to_i
            amount = amount - "#{order["quantity"]}".to_i
            orders.data.delete(n)
        end
        n += 1
    end
    CSV.open("Orders.csv", "w+") do |csv|
        csv << ["product", "quantity", 'phnumber']
        orders.data.each do |row|
        csv << [row["product"], row["quantity"], row["phnumber"]]
        end
    end
    puts "Customers have been notified"
```

gives instructions

receives input

iterates through orders data referencing names from products data

< Gem to message/email

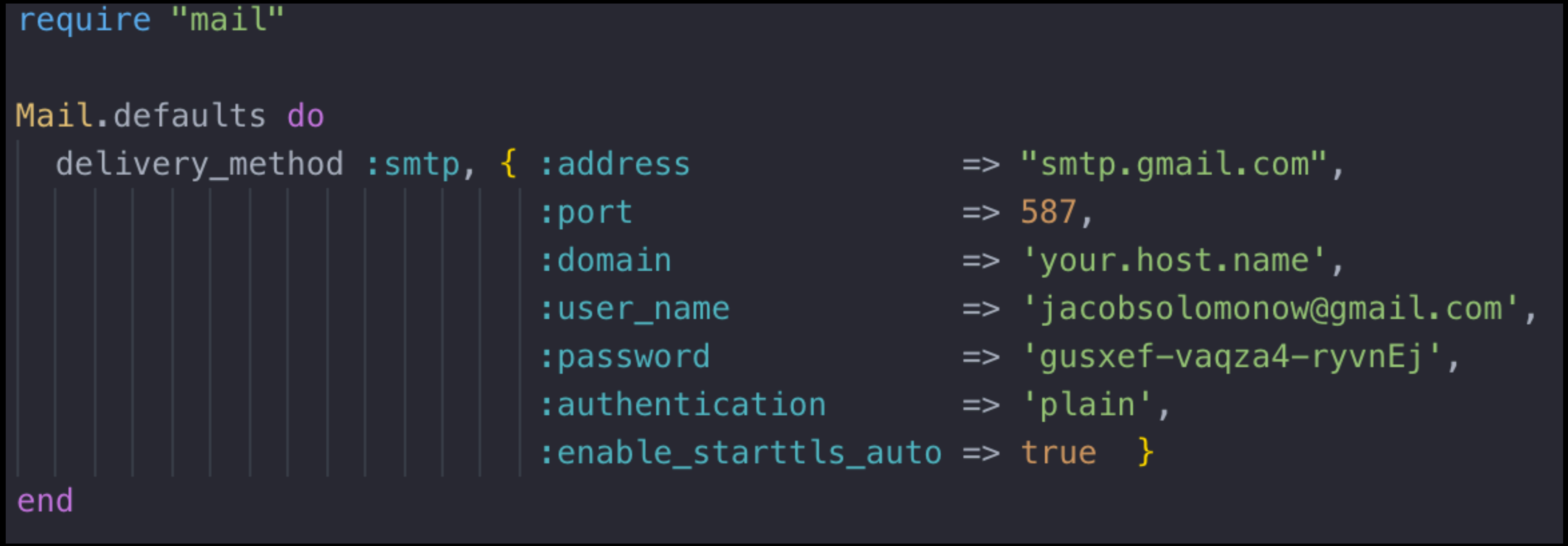
changes stock remaining amount

deletes row

saves changes to data to CSV

Mail Gem

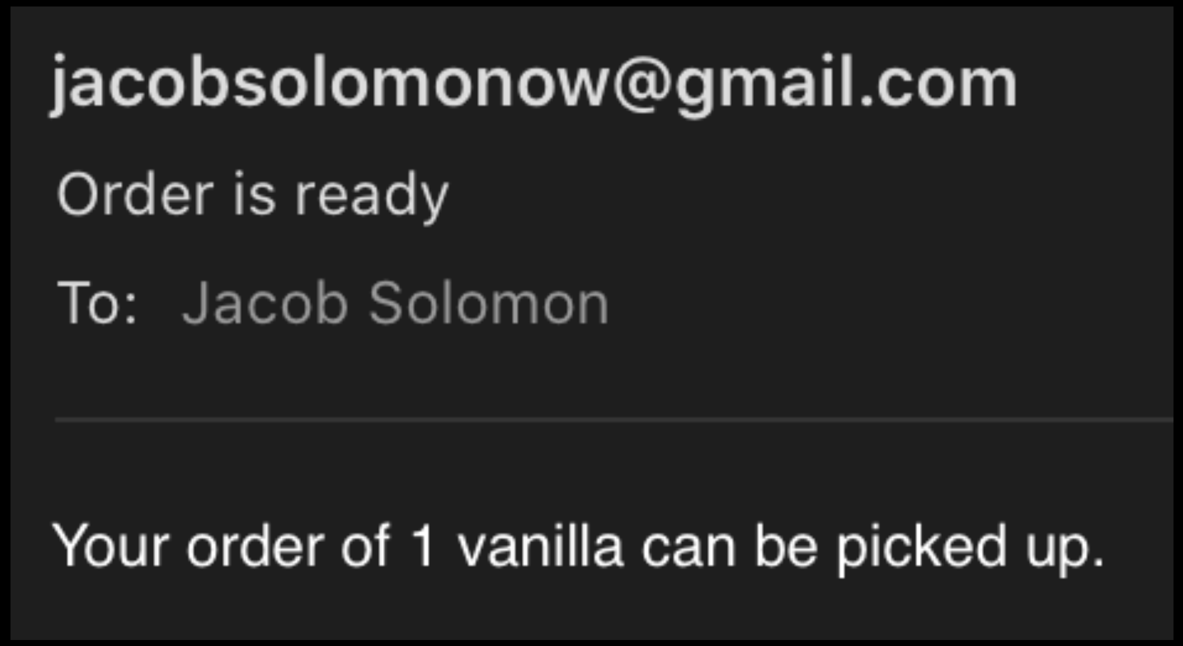
mail file



```ruby
require "mail"

Mail.defaults do
  delivery_method :smtp, { :address              => "smtp.gmail.com",
                           :port                 => 587,
                           :domain               => 'your.host.name',
                           :user_name            => 'jacobsolomonow@gmail.com',
                           :password             => 'gusxef-vaqza4-ryvnEj',
                           :authentication       => 'plain',
                           :enable_starttls_auto => true  }
end
```

setup



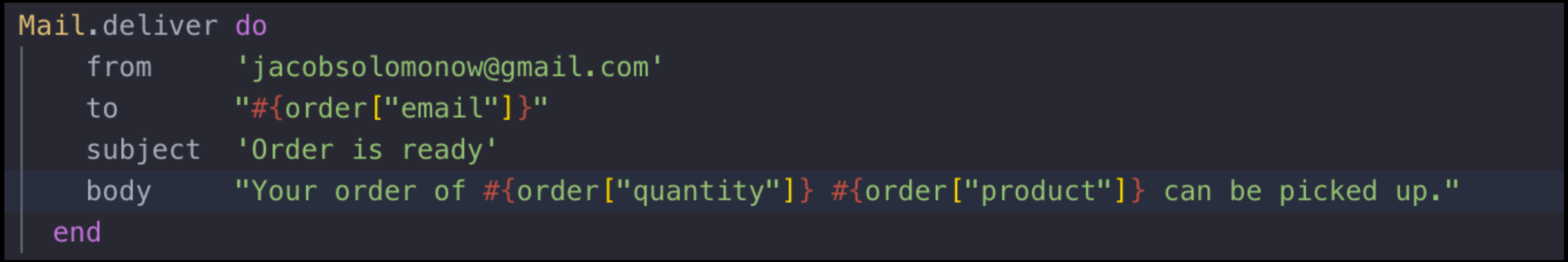
jacobsolomonow@gmail.com

Order is ready

To: Jacob Solomon

Your order of 1 vanilla can be picked up.

main file

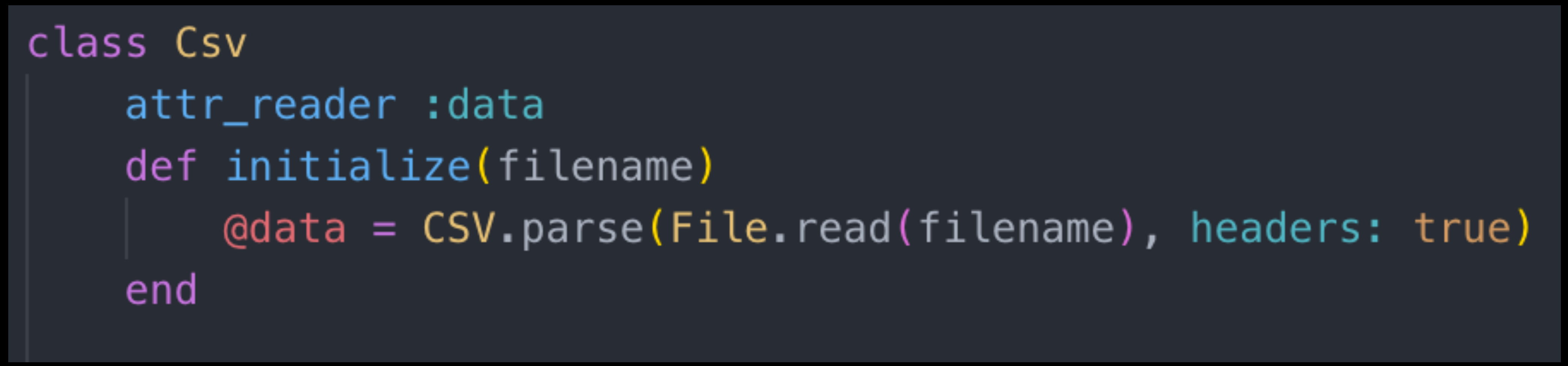


```ruby
Mail.deliver do
    from     'jacobsolomonow@gmail.com'
    to       "#{order["email"]}"
    subject  'Order is ready'
    body     "Your order of #{order["quantity"]} #{order["product"]} can be picked up."
  end
```

calls the function

Display orders

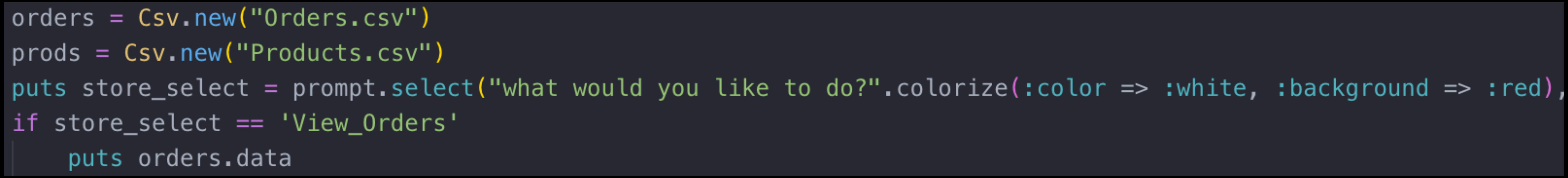
class file



```ruby
class Csv
    attr_reader :data
    def initialize(filename)
        @data = CSV.parse(File.read(filename), headers: true)
    end
```

parse csv to class

main file



```ruby
orders = Csv.new("Orders.csv")
prods = Csv.new("Products.csv")
puts store_select = prompt.select("what would you like to do?".colorize(:color => :white, :background => :red),
if store_select == 'View_Orders'
    puts orders.data
```

create new class orders

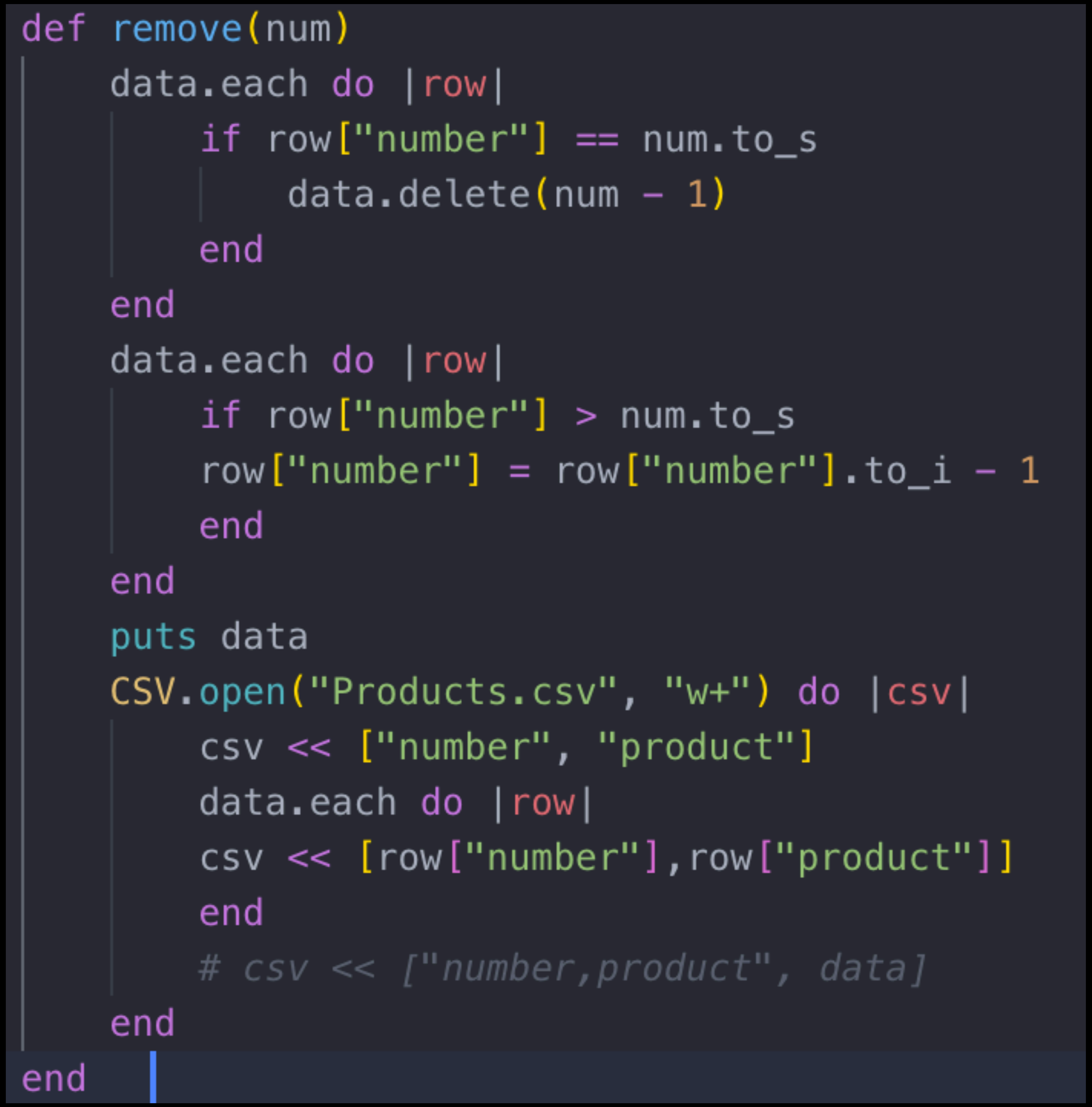
TTY prompt in action

colorize in action

displays data

Remove Product

class file



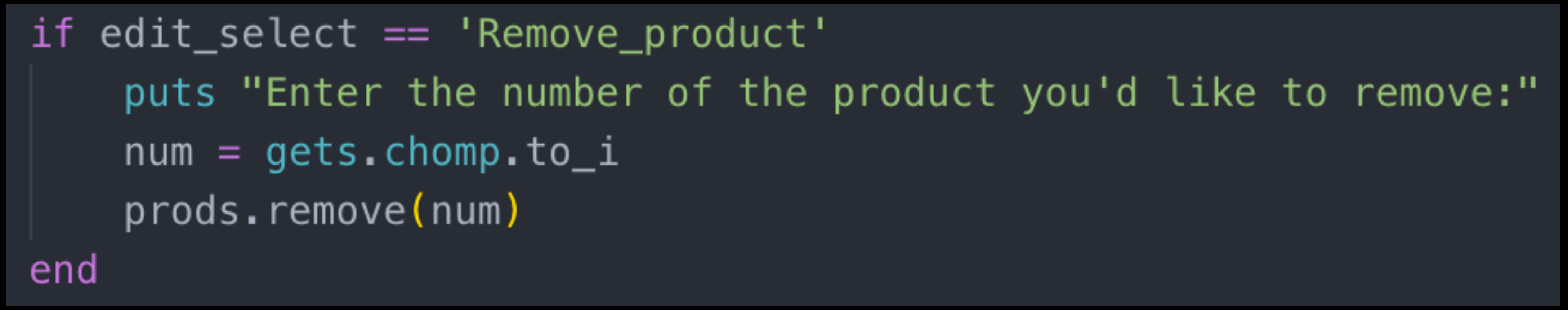
```ruby
def remove(num)
    data.each do |row|
        if row["number"] == num.to_s
            data.delete(num - 1)
        end
    end
    data.each do |row|
        if row["number"] > num.to_s
        row["number"] = row["number"].to_i - 1
        end
    end
    puts data
    CSV.open("Products.csv", "w+") do |csv|
        csv << ["number", "product"]
        data.each do |row|
        csv << [row["number"],row["product"]]
        end
        # csv << ["number,product", data]
    end
end
```

deletes the row

modifies list number

saves modified data to csv

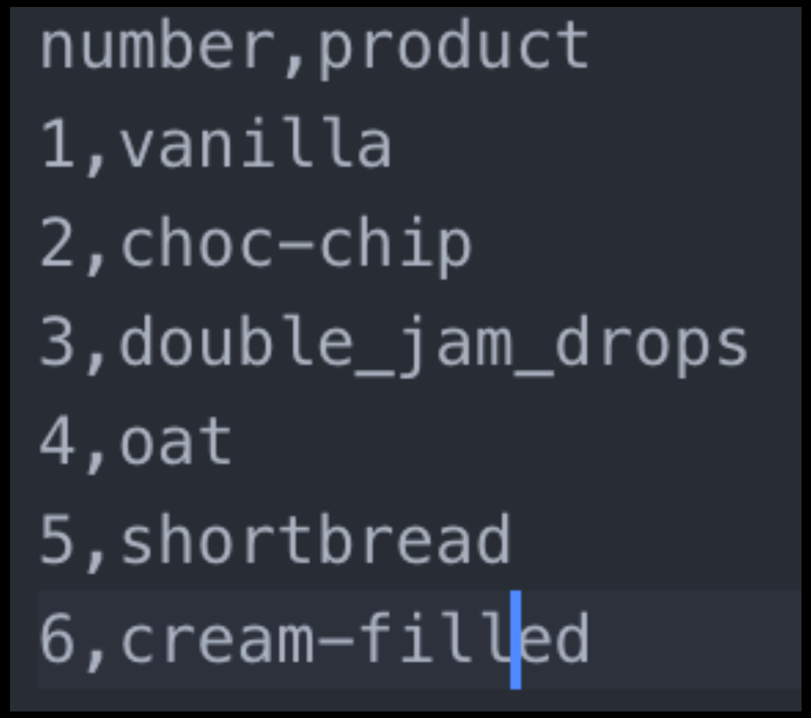
main file



```ruby
if edit_select == 'Remove_product'
    puts "Enter the number of the product you'd like to remove:"
    num = gets.chomp.to_i
    prods.remove(num)
end
```

calls the function

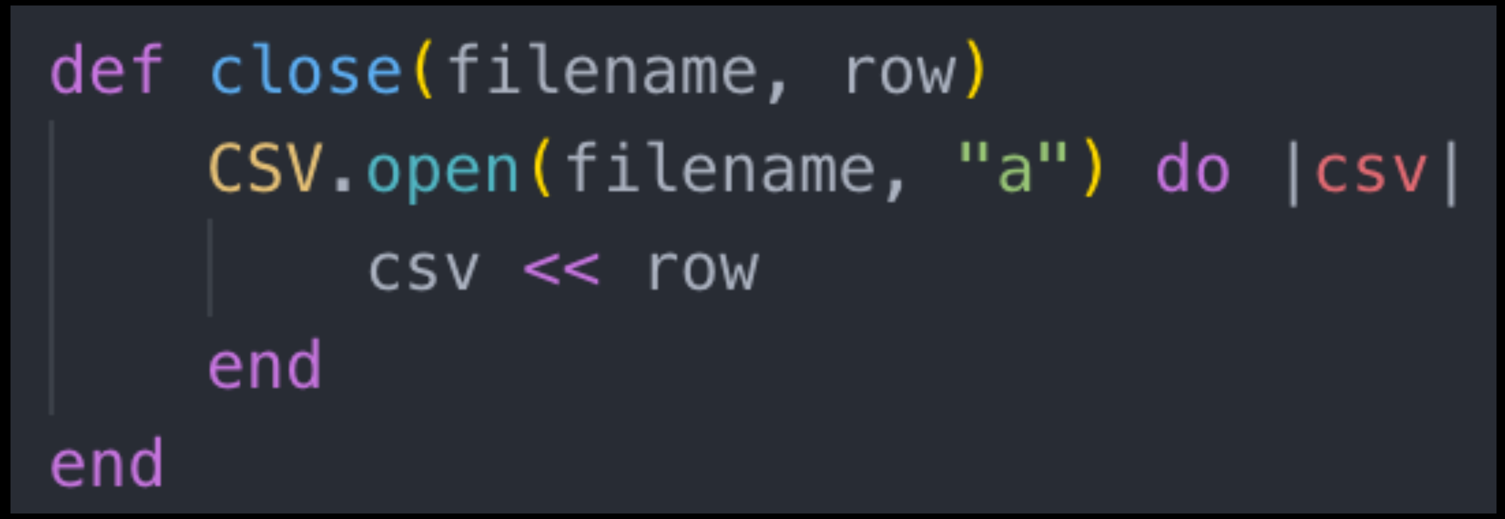
product csv



```
number,product
1,vanilla
2,choc-chip
3,double_jam_drops
4,oat
5,shortbread
6,cream-filled
```

Add product

class file

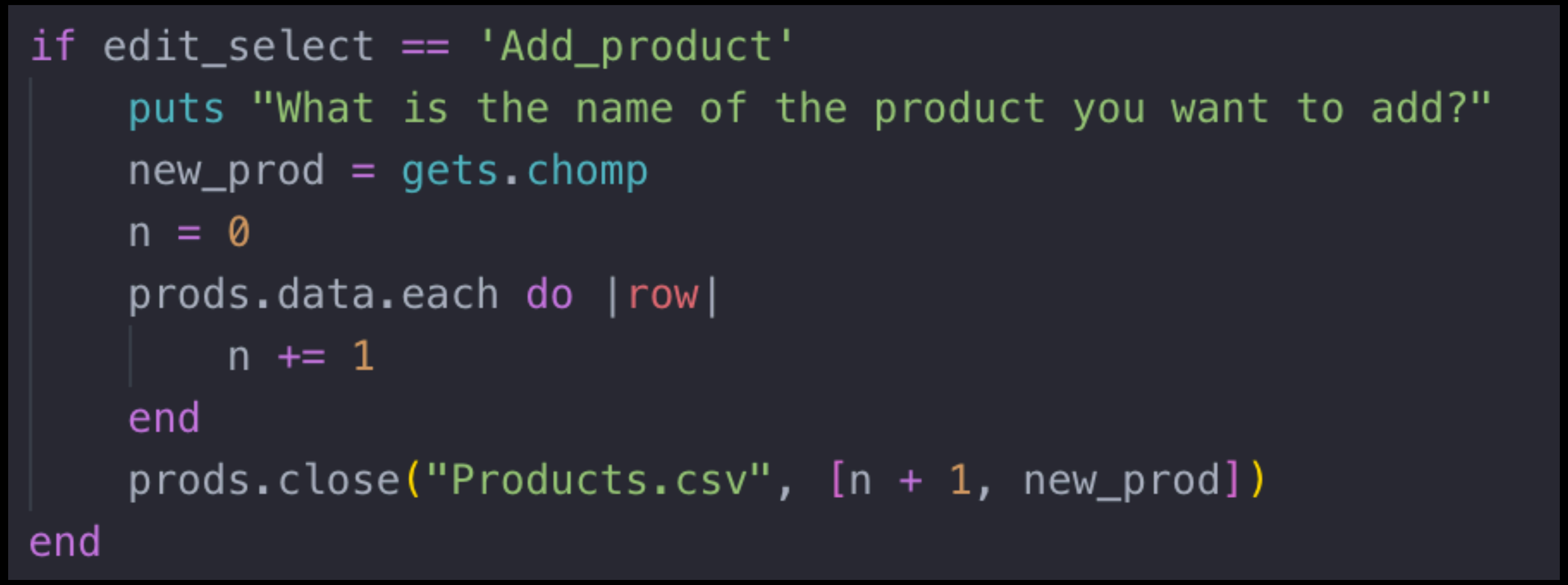


```ruby
def close(filename, row)
    CSV.open(filename, "a") do |csv|
        csv << row
    end
end
```

opens function

append option

main file

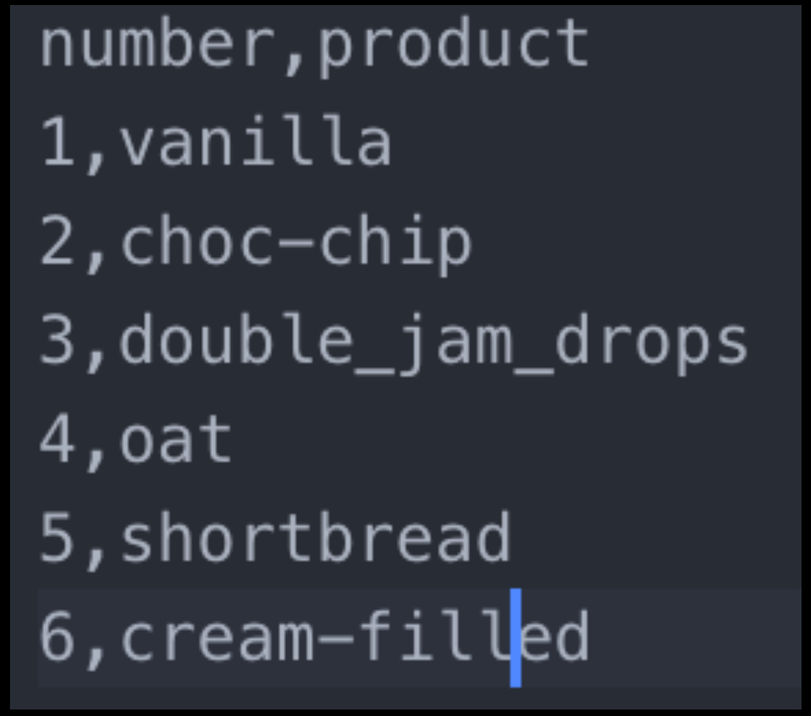


```ruby
if edit_select == 'Add_product'
    puts "What is the name of the product you want to add?"
    new_prod = gets.chomp
    n = 0
    prods.data.each do |row|
        n += 1
    end
    prods.close("Products.csv", [n + 1, new_prod])
end
```

takes input

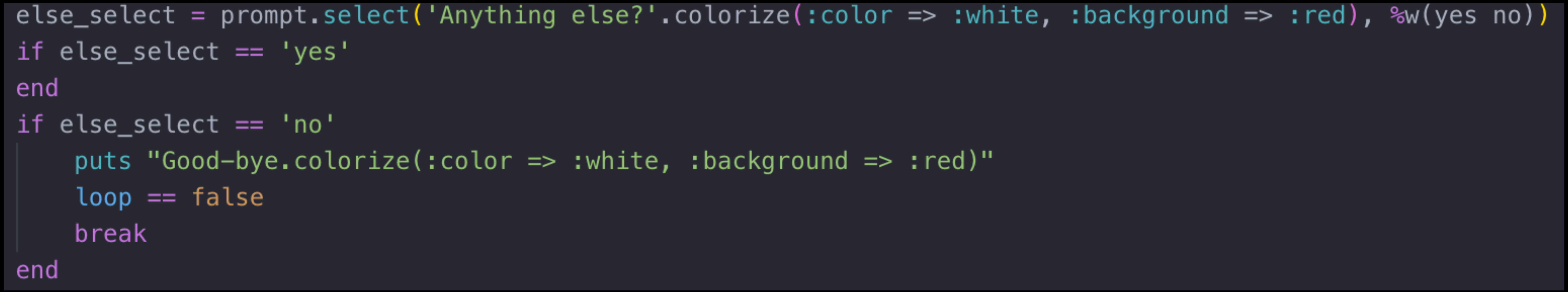
call the method and pass variable

product csv



```
number,product
1,vanilla
2,choc-chip
3,double_jam_drops
4,oat
5,shortbread
6,cream-filled
```

Anything else



```
else_select = prompt.select('Anything else?'.colorize(:color => :white, :background => :red), %w(yes no))
if else_select == 'yes'
end
if else_select == 'no'
    puts "Good-bye.colorize(:color => :white, :background => :red)"
    loop == false
    break
end
```

TTY & colorize double team

choose from yes or no

quit or return to top of loop

challenges, ethical issues, favourite parts

Challenges

- CSV input how i wanted it to
- Looking for gems!

Ethical issues

- What will get sold?
- Other people with same app?

Favourite parts

- Small wins
- breaking the flow chart down