HAUNTED AIRBNB ESCAPE GAME

Terminal app design & documentation by Hannah McDonald

GAME OVERVIEW

The player wakes up to find that their super host is actually supernatural.

In order to survive the night, they must find the key item and use it in the living room.

Other game items will either improve their guest rating or ruin it.

Use of the incorrect items will result in a guest rating of 0 and the g-host will kill them.

User Interaction

The user enters command line inputs to navigate their way around the Airbnb.

Command words are:

quit help backpack take use go

STRUCTURE
of Classes & Objects

StartMenu

Loops TTY prompt of Start or Quit

Start
Creates new game object

Quit
Program is exited

Game

Loops handle_input until user wins, loses or enters 'exit'

All game objects are initialized & accessed in game.
E.g:
kitchen = Room.new("kitchen")
kitchen.has_exit?("north")
All done in game.

go_room take_item use_item

Classes & Objects cont.

Player

Responsible for storing the backpack and defining methods used to access it.

print_backpack

remove_item

pick_up("key")

Item

Responsible for storing and accessing Item object information

is_key?

use_description

collection_description

Room

Responsible for storing and accessing Room object information

has_exit?("north")

get_exit("north")

has_item?("key")

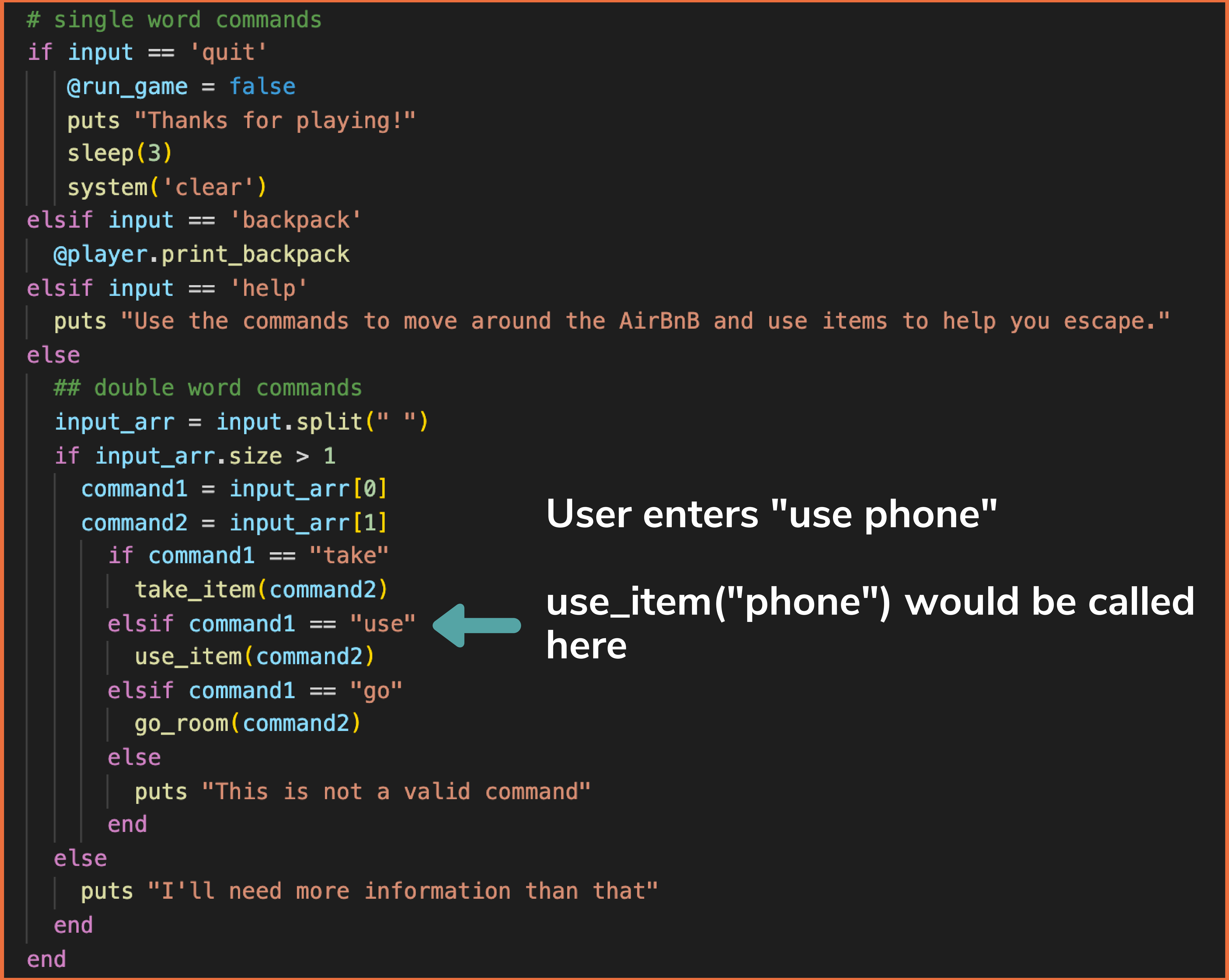
GAME LOGIC

Handling input

Uses if-statement to compare input to commands

E.g quit ends the game

If command matches take, use or go then second command entered is stored and given to another method



```ruby
# single word commands
if input == 'quit'
    @run_game = false
    puts "Thanks for playing!"
    sleep(3)
    system('clear')
elsif input == 'backpack'
  @player.print_backpack
elsif input == 'help'
  puts "Use the commands to move around the AirBnB and use items to help you escape."
else
  ## double word commands
  input_arr = input.split(" ")
  if input_arr.size > 1
    command1 = input_arr[0]
    command2 = input_arr[1]
      if command1 == "take"
        take_item(command2)
      elsif command1 == "use"
        use_item(command2)
      elsif command1 == "go"
        go_room(command2)
      else
        puts "This is not a valid command"
      end
  else
    puts "I'll need more information than that"
  end
end
```

User enters "use phone"

use_item("phone") would be called here

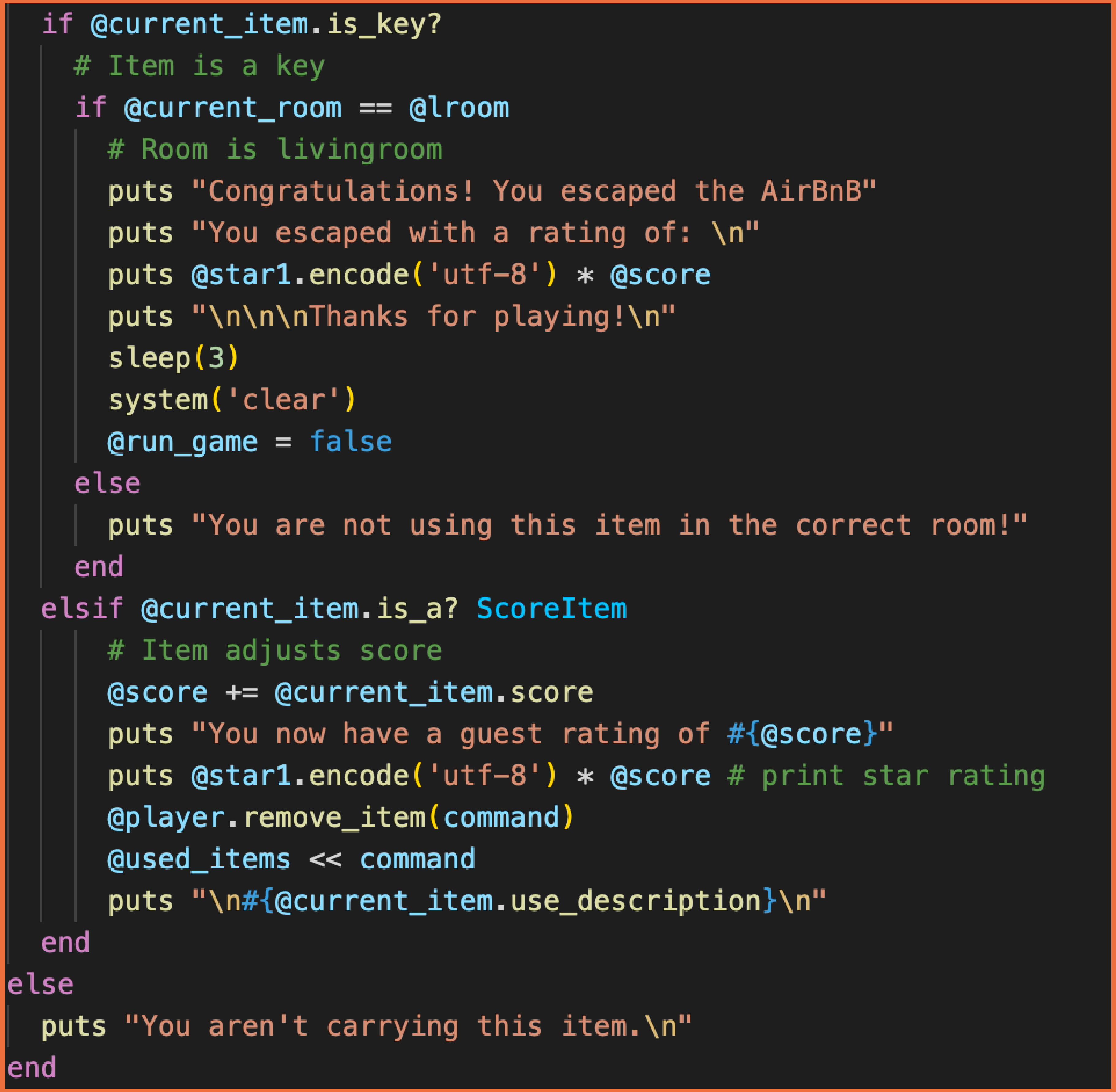
Logic for processing commands and giving appropriate output

use_item
take_item
go_room

All take the second user command and return the object user is referring to.

Use this object to change the games variables (e.g @current_room = @livingroom)

Then print appropriate feedback to user



```
  if @current_item.is_key?
    # Item is a key
    if @current_room == @lroom
      # Room is livingroom
      puts "Congratulations! You escaped the AirBnB"
      puts "You escaped with a rating of: \n"
      puts @star1.encode('utf-8') * @score
      puts "\n\n\nThanks for playing!\n"
      sleep(3)
      system('clear')
      @run_game = false
    else
      puts "You are not using this item in the correct room!"
    end
  elsif @current_item.is_a? ScoreItem
      # Item adjusts score
      @score += @current_item.score
      puts "You now have a guest rating of #{@score}"
      puts @star1.encode('utf-8') * @score # print star rating
      @player.remove_item(command)
      @used_items << command
      puts "\n#{@current_item.use_description}\n"
  end
else
  puts "You aren't carrying this item.\n"
end
```

Use Item

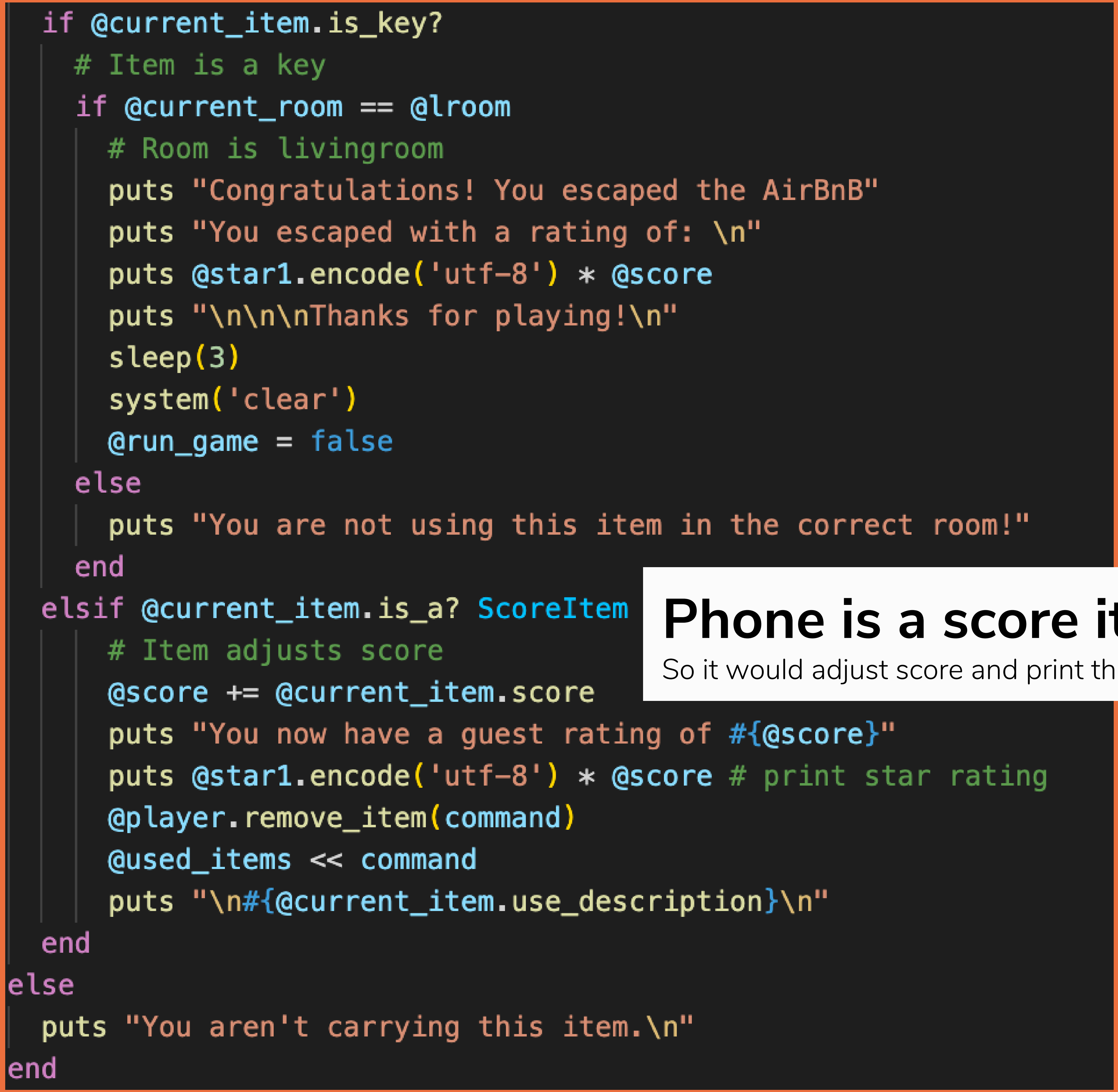
Logic for processing commands and giving appropirate output

use_item
take_item
go_room

All take the second user command and return the object user is referring to.

Use this object to change the games variables (e.g @current_room = @livingroom)

Then print appropriate feedback to user

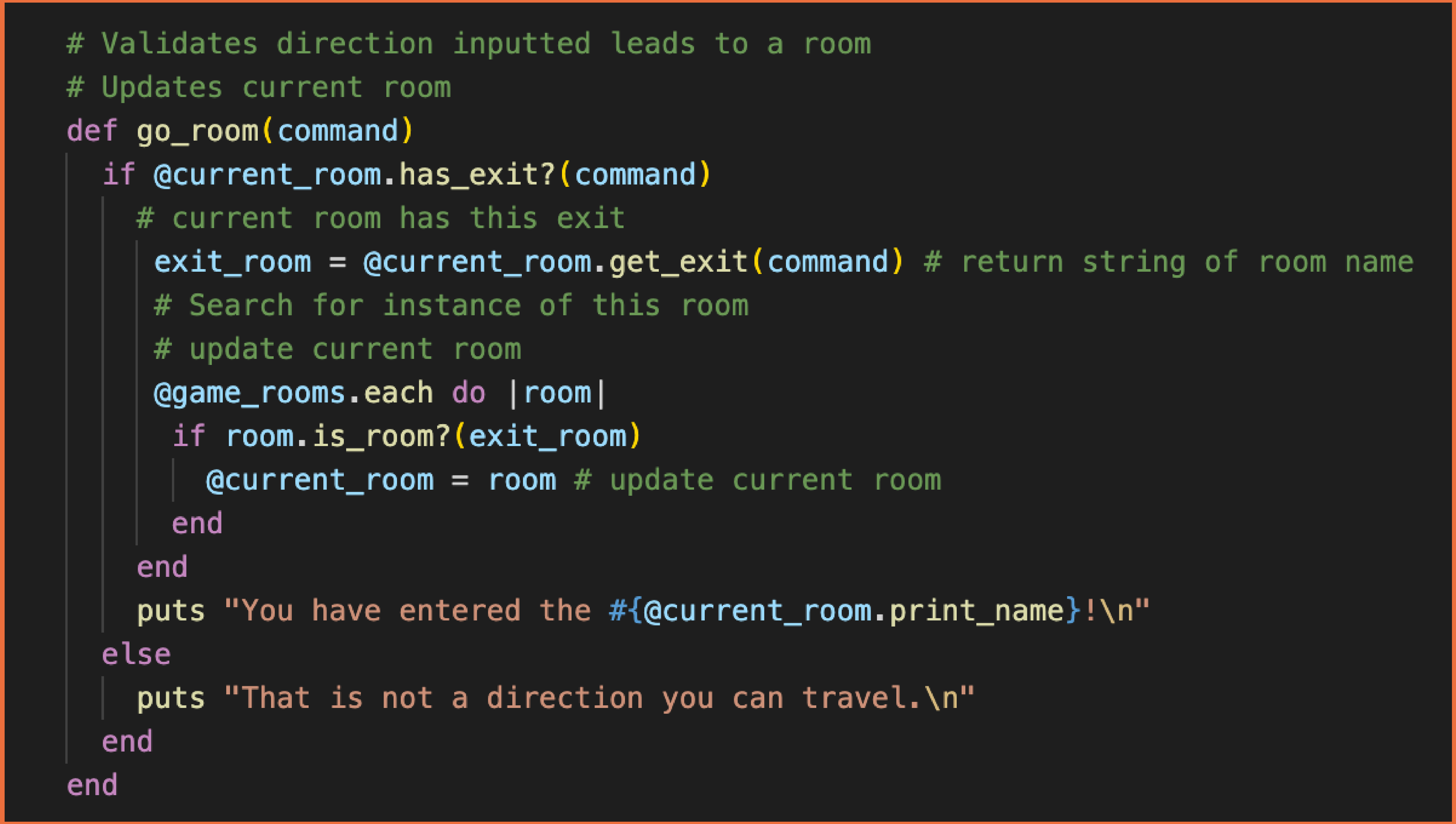


```
  if @current_item.is_key?
    # Item is a key
    if @current_room == @lroom
      # Room is livingroom
      puts "Congratulations! You escaped the AirBnB"
      puts "You escaped with a rating of: \n"
      puts @star1.encode('utf-8') * @score
      puts "\n\n\nThanks for playing!\n"
      sleep(3)
      system('clear')
      @run_game = false
    else
      puts "You are not using this item in the correct room!"
    end
  elsif @current_item.is_a? ScoreItem
      # Item adjusts score
      @score += @current_item.score
      puts "You now have a guest rating of #{@score}"
      puts @star1.encode('utf-8') * @score # print star rating
      @player.remove_item(command)
      @used_items << command
      puts "\n#{@current_item.use_description}\n"
  end
else
  puts "You aren't carrying this item.\n"
end
```

Phone is a score item

So it would adjust score and print this to user

Go Room logic



```ruby
# Validates direction inputted leads to a room
# Updates current room
def go_room(command)
  if @current_room.has_exit?(command)
    # current room has this exit
     exit_room = @current_room.get_exit(command) # return string of room name
     # Search for instance of this room
     # update current room
     @game_rooms.each do |room|
      if room.is_room?(exit_room)
        @current_room = room # update current room
      end
    end
    puts "You have entered the #{@current_room.print_name}!\n"
  else
    puts "That is not a direction you can travel.\n"
  end
end
```

Each room has a hash of exits

Eg. North => Livingroom, East = > Kitchen

go_room asks the current room a few questions:

- Is this direction in your exit hash?
- Which room does this exit lead to?

The answers to these questions allow go_room to find and update the current room.

Game_run is set to false when:

Game is won: current room is livingroom and the player uses the key.

Game is lost: AirBnB rating reaches 0.

Game is exited: User enters 'quit'

REVIEW

Challenges

Finding gems

Time management

Algorithmic thinking

Ethical Issues

None

THE BEST BITS

Using task management tools

Seeing a solution work

Gaining a better understading of OOP

THANKS FOR LISTENING!