```python
# Now we have a premise. We are in a room and we have two door to choose from.
# We are still in the blue room. Now we have to deal with the guard
# after taking or leaving the treasure chest.
#
# Run this code a few times and see what happens with different choices.
# It's good to test all options and see if that's what you expected.

##### ACTIONS #####
def you_died(why):
    # You expect a reason why the player died. It's a string.
    print("{}. Good job!".format(why))

    # This exits the program entirely.
    exit(0)

### END ACTIONS ###

### CHARACTERS ###
def guard():
    # The guard
    print("You approach the guard, he's still sleeping.")
    print("Suddenly you knocked a wooden cask with a mug on it... CRASSH!")
    print("\nOi, what you doing 'ere?")

    # BOOLEANS
    # Booleans are either True or False. In this example, we will do the following:
    # - Initially the guard doesn't move, so we set a variable called guard_moved to False
    # - We give the player an option to run or go to the door.
    # ----- If player decides to run and the guard has moved (guard_moved = True)
    #       Result: Game over
    #
    # ----- If player decides to run and the guard hasn't moved (guard_moved = False)
    #       Result: Guard stupidly looks the other way, and we set guard_moved = True
    #
    # ----- If player goes for the door and the guard has moved (guard_moved = True)
    #       Result: Freedom!
    #
    #       In the code "return" returns the code execution to the block of code where
    #       the function was called from, in this case, blissful_ignorance_of_illusion_room()
    #       Since there's nothing else to do in blissful_ignorance_of_illusion_room() after
    #       calling guard(), it automaticall returns to start_adventure() and returns to main().
    #
    #       At this stage in main(), after start_adventure() on line 152, you can now print out
    #       messages to the player that they have finished the game successfully (and alive).
    #
    # ----- If player goes for the door and the guard hasn't moved (guard_moved = False)
    #       Result: Game over.
    #
    # ----- If player types something gibberish and not recognised
    #       Result: Loops around until the player types run or door.
    #
    guard_moved = False

    # WHILE LOOP
    # This is how the question keeps getting asked if a player types in anything other
    # than run or door, or hasn't died or escaped yet.
    #
    # WARNING: while loops are dangerous but it is good to know about them and understand
    # how they work. They can cause a program to just go into an infinite loop, and looks
    # like nothing is happening. You can use for loops for most cases.
    #
    # There are ways to escape a while loop, in this example:
    # - When a player dies, it calls you_died() and it exits() the program.
    # - When a player escapes through the door, you return to the previous function which
    #   called this function.
    while True:
        next_action = raw_input("[run | door] > ").lower()
        if next_action == "run" and guard_moved:
            you_died("Guard was faster than he looks and your world goes dark...")
        elif next_action == "run" and not guard_moved:
            print("Guard jumps up and looks the other way, missing you entirely.")
            guard_moved = True
        elif next_action == "door" and guard_moved:
            print("You just slipped through the door before the guard realised it.")
            print("You are now outside, home free! Huzzah!")
            return
        elif next_action == "door" and not guard_moved:
            you_died("Guard was faster than he looks and your world goes dark...")
        else:
            print("Not sure what you meant there... try again.")
# END CHARACTERS #

##### ROOMS #####
def blissful_ignorance_of_illusion_room():
    # The variable treasure_chest is an object type called a list
    # A list maybe empty as well.
```

```
87      # So our treasure_chest list contains 4 items.
88      treasure_chest = ["diamonds", "gold", "silver", "sword"]
89      print("You see a room with a wooden treasure chest on the left, and a sleeping guard on the right in front of the door")
90
91      # Ask player what to do.
92      action = raw_input("What do you do? > ")
93
94      # This is a way to see if the text typed by player is in the list
95      if action.lower() in ["treasure", "chest", "left"]:
96          print("Oooh, treasure!")
97
98          print("Open it? Press '1'")
99          print("Leave it alone. Press '2'")
100         choice = raw_input("> ")
101
102         # Try just leaving 1 and 2 as a number
103         # Change to string and see what happens
104         if choice == "1":
105             print("Let's see what's in here... /grins")
106             print("The chest creaks open, and the guard is still sleeping. That's one heavy sleeper!")
107             print("You find some")
108
109             # for each treasure (variable created on the fly in the for loop)
110             # in the treasure_chest list, print the treasure.
111             for treasure in treasure_chest:
112                 print(treasure)
113
114             # So much treasure, what to do? Take it or leave it.
115             print("What do you want to do?")
116             print("Take all {} treasure, press '1'".format(len(treasure_chest)))
117             print("Leave it, press '2'")
118
119             treasure_choice = raw_input("> ")
120             if treasure_choice == "1":
121                 print("\tWoohoo! Bounty and a shiney new sword. /drops your crappy sword in the empty treasure chest.")
122                 print("\tYou just received [{}]".format(", ".join(treasure_chest)))
123             elif treasure_choice == "2":
124                 print("It will still be here (I hope), right after I get past this guard")
125
126             # Picked up treasure or left it, you will now encounter the guard.
127             # Let's call the guard() function here.
128             guard()
129     else:
130         # Let's call the guard() function here as well, no point writing a bunch of same code
131         # twice (or more). It's good to be able to re-use code.
132         print("The guard is more interesting, let's go that way!")
133         guard()
134
135
136 def painful_truth_of_reality_room():
137     print("There you see the great evil Cthulhu.")
138     print("He, it, whatever stares at you and you go insane.")
139     print("Do you flee for your life or eat your head?")
140
141     next_move = raw_input("> ")
142
143     # Flee to return to the start of the game, in the room with the blue and red door or die!
144     if "flee" in next_move:
145         start_adventure()
146     else:
147         # You call the function you_died and pass the reason why you died as
148         # a string as an argument.
149         you_died("You died. Well, that was tasty!")
150 ### END ROOMS ###
151
152 def start_adventure():
153     print("You enter a room, and you see a red door to your left and a blue door to your right.")
154     door_picked = raw_input("Do you pick the red door or blue door? > ")
155
156     # Pick a door and we go to a room and something else happens
157     if door_picked == "red":
158         painful_truth_of_reality_room()
159     elif door_picked == "blue":
160         blissful_ignorance_of_illusion_room()
161     else:
162         print("Sorry, it's either 'red' or 'blue' as the answer. You're the weakest link, goodbye!")
163
164 def main():
165     player_name =  raw_input("What's your name? >")
166     print("Your name is {}".format(player_name.upper()))
167
168     start_adventure()
169
170     print("\nThe end\n")
171     print("Thanks for playing, {}".format(player_name.upper()))
172
173
```

```
174 if __name__ == '__main__':
175     main()
```