```python
# Let's clean it up (or refactor), and create a function to ask for your name and return it.
# This will keep main() as clean as possible.
# New code starts at line 147
#
# 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?")

    # Guard is not moving initially
    guard_moved = False

    # - 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.
    # So our treasure_chest list contains 4 items.
    treasure_chest = ["diamonds", "gold", "silver", "sword"]
    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")

    # Ask player what to do.
    action = raw_input("What do you do? > ")

    # This is a way to see if the text typed by player is in the list
    if action.lower() in ["treasure", "chest", "left"]:
        print("Oooh, treasure!")

        print("Open it? Press '1'")
        print("Leave it alone. Press '2'")
        choice = raw_input("> ")

        # Try just leaving 1 and 2 as a number
        # Change to string and see what happens
        if choice == "1":
            print("Let's see what's in here... /grins")
            print("The chest creaks open, and the guard is still sleeping. That's one heavy sleeper!")
            print("You find some")

            # for each treasure (variable created on the fly in the for loop)
            # in the treasure_chest list, print the treasure.
            for treasure in treasure_chest:
                print(treasure)

            # So much treasure, what to do? Take it or leave it.
            print("What do you want to do?")
            print("Take all {} treasure, press '1'".format(len(treasure_chest)))
            print("Leave it, press '2'")

            treasure_choice = raw_input("> ")
            if treasure_choice == "1":
                print("\tWoohoo! Bounty and a shiney new sword. /drops your crappy sword in the empty treasure chest.")
```

```
87                print("\tYou just received [{}]".format(", ".join(treasure_chest)))
88            elif treasure_choice == "2":
89                print("It will still be here (I hope), right after I get past this guard")
90
91            # Picked up treasure or left it, you will now encounter the guard.
92            # Let's call the guard() function here.
93            guard()
94    else:
95        # Let's call the guard() function here as well, no point writing a bunch of same code
96        # twice (or more). It's good to be able to re-use code.
97        print("The guard is more interesting, let's go that way!")
98        guard()
99
100
101 def painful_truth_of_reality_room():
102     print("There you see the great evil Cthulhu.")
103     print("He, it, whatever stares at you and you go insane.")
104     print("Do you flee for your life or eat your head?")
105
106     next_move = raw_input("> ")
107
108     # Flee to return to the start of the game, in the room with the blue and red door or die!
109     if "flee" in next_move:
110         start_adventure()
111     else:
112         # You call the function you_died and pass the reason why you died as
113         # a string as an argument.
114         you_died("You died. Well, that was tasty!")
115 ### END ROOMS ###
116
117 def get_player_name():
118     # LOCAL VARIABLES
119     # The player enters their name and gets assigned to a variable called "name"
120     name = raw_input("What's your name? > ")
121
122     # This is just an alternative name that the game wants to call the player
123     alt_name = "Rainbow Unicorn"
124     answer = raw_input("Your name is {}, is that correct? [Y|N] > ".format(alt_name.upper()))
125     if answer.lower() in ["y", "yes"]:
126         name = alt_name
127         print("You are fun, {}! Let's begin our adventure!").format(name.upper())
128     elif answer.lower() in ["n", "no"]:
129         print("Ok, picky. {} it is. Let's get started on our adventure.".format(name.upper()))
130     else:
131         print("Trying to be funny? Well, you will now be called {} anyway.".format(alt_name.upper()))
132         name = alt_name
133
134     # Now notice that we are returning the variable called name.
135     # In main(), it doesn't know what the variable "name" is, as it only exists in
136     # get_player_name() function.
137     # This is why indentation is important, variables declared in this block only exists in that block
138     return name
139
140 def start_adventure():
141     print("You enter a room, and you see a red door to your left and a blue door to your right.")
142     door_picked = raw_input("Do you pick the red door or blue door? > ")
143
144     # Pick a door and we go to a room and something else happens
145     if door_picked == "red":
146         painful_truth_of_reality_room()
147     elif door_picked == "blue":
148         blissful_ignorance_of_illusion_room()
149     else:
150         print("Sorry, it's either 'red' or 'blue' as the answer. You're the weakest link, goodbye!")
151
152 def main():
153     # Calls get_player_name and returns the player name
154     player_name =  get_player_name()
155
156     ######################################################################
157     # ACTIVITIES
158     #
159     # Read some of the best practices when writing Python code
160     #   http://legacy.python.org/dev/peps/pep-0008/
161     # Main thing is if you are using tabs, make sure it's 4-spaces,
162     # most editors will convert it (check preferences/settings).
163     #
164     # Modify the code
165     # - add taunting the guard or talking
166     # - sword fight with the guard, and keep track of health points (HP)
167     # - puzzles like 1+2 during an encounter
168     # - modifiy blissful_ignorance_of_illusion_room()'s if statement
169     #   so it takes into account player typing "right" or "guard"
170     #   Hint: Add another elif before the else statement
171     #
172     # So many if statements, this can be made simpler and easier to
173     # maintain by using Finite State Machine (FSM)
```

```
174     # You can find info about it, but it will mainly be touching
175     # object-orient programming, which is another lesson for another day.
176     #
177     ######################################################################
178
179     start_adventure()
180
181     print("\nThe end\n")
182     print("Thanks for playing, {}".format(player_name.upper()))
183
184
185 if __name__ == '__main__':
186     main()
```