# Define items

door\_a = {"name": "door\_a", "type": "door"}

door\_b = {"name": "door\_b", "type": "door"}

door\_c = {"name": "door\_c", "type": "door"}

door\_d = {"name": "door\_d", "type": "door"}

key\_a = {"name": "key for door a", "type": "key", "target": door\_a}

key\_b = {"name": "key for door b", "type": "key", "target": door\_b}

key\_c = {"name": "key for door c", "type": "key", "target": door\_c}

key\_d = {"name": "key for door d", "type": "key", "target": door\_d}

couch = {"name": "couch", "type": "furniture"}

piano = {"name": "piano", "type": "furniture"}

double\_bed = {"name": "double\_bed", "type": "furniture"}

queen\_bed = {"name": "queen\_bed", "type": "furniture"}

dresser = {"name": "dresser", "type": "furniture"}

dining\_table = {"name": "dining\_table", "type": "furniture"}

# Define rooms

game\_room = {"name": "game\_room", "type": "room"}

bedroom\_1 = {"name": "bedroom\_1", "type": "room"}

bedroom\_2 = {"name": "bedroom\_2", "type": "room"}

living\_room = {"name": "living room", "type": "room"}

congratulations = {"name": "congratulations", "type": "room"}

# All rooms and doors

all\_rooms = [game\_room, bedroom\_1, bedroom\_2, living\_room, congratulations]

all\_doors = [door\_a, door\_b, door\_c, door\_d]

# Define inciate

INIT\_GAME\_STATE = {"current\_room" : game\_room , "collected\_keys" : () , "target\_room" : congratulations}

# Define objects relations

object\_relations = {

  "game\_room" : (couch, piano, door\_a) , "piano" : (key\_a) ,

  "bedroom\_1" : (queen\_bed, door\_a, door\_b, door\_c) , "queen\_bed" : (key\_b) ,

  "bedroom\_2" : (double\_bed, dresser, door\_b) , "double\_bed" : (key\_c) , "dresser" : (key\_d) ,

  "living\_room" : (dining\_table, door\_c, door\_d) ,

  "congratulations" : (door\_d) ,

  "door\_a" : (game\_room , bedroom\_1) ,

  "door\_b" : (bedroom\_1 , bedroom\_2) ,

  "door\_c" : (bedroom\_1 , living\_room) ,

  "door\_d" : (living\_room, congratulations)

}

def linebreak():

    """

    Print a line break

    """

    print("\n\n")

def start\_game():

    """

    Start the game

    """

    print("You wake up on a couch and find yourself in a strange house with no windows which you have never been to before. You don't remember why you are here and what had happened before. You feel some unknown danger is approaching and you must get out of the house, NOW!")

    play\_room(game\_state["current\_room"])

def play\_room(room):

    """

    Play a room. First check if the room being played is the target room.

    If it is, the game will end with success. Otherwise, let player either

    explore (list all items in this room) or examine an item found here.

    """

    game\_state["current\_room"] = room

    if(game\_state["current\_room"] == game\_state["target\_room"]):

        print("Congrats! You escaped the room!")

    else:

        print("You are now in " + room["name"])

        intended\_action = input("What would you like to do? Type 'explore' or 'examine'?").strip()

        if intended\_action == "explore":

            explore\_room(room)

            play\_room(room)

        elif intended\_action == "examine":

            examine\_item(input("What would you like to examine?").strip())

        else:

            print("Not sure what you mean. Type 'explore' or 'examine'.")

            play\_room(room)

        linebreak()

def explore\_room(room):

    """

    Explore a room. List all items belonging to this room.

    """

    items = [i["name"] for i in object\_relations[room["name"]]]

    print("You explore the room. This is " + room["name"] + ". You find " + ", ".join(items))

def get\_next\_room\_of\_door(door, current\_room):

    """

    From object\_relations, find the two rooms connected to the given door.

    Return the room that is not the current\_room.

    """

    connected\_rooms = object\_relations[door["name"]]

    for room in connected\_rooms:

        if(not current\_room == room):

            return room

def examine\_item(item\_name):

    """

    Examine an item which can be a door or furniture.

    First make sure the intended item belongs to the current room.

    Then check if the item is a door. Tell player if key hasn't been

    collected yet. Otherwise ask player if they want to go to the next

    room. If the item is not a door, then check if it contains keys.

    Collect the key if found and update the game state. At the end,

    play either the current or the next room depending on the game state

    to keep playing.

    """

    current\_room = game\_state["current\_room"]

    next\_room = ""

    output = None

    for item in object\_relations[current\_room["name"]]:

        if(item["name"] == item\_name):

            output = "You examine " + item\_name + ". "

            if(item["type"] == "door"):

                have\_key = False

                for key in game\_state["keys\_collected"]:

                    if(key["target"] == item):

                        have\_key = True

                if(have\_key):

                    output += "You unlock it with a key you have."

                    next\_room = get\_next\_room\_of\_door(item, current\_room)

                else:

                    output += "It is locked but you don't have the key."

            else:

                if(item["name"] in object\_relations and len(object\_relations[item["name"]])>0):

                    item\_found = object\_relations[item["name"]].pop()

                    game\_state["keys\_collected"].append(item\_found)

                    output += "You find " + item\_found["name"] + "."

                else:

                    output += "There isn't anything interesting about it."

            print(output)

            break

    if(output is None):

        print("The item you requested is not found in the current room.")

    if(next\_room and input("Do you want to go to the next room? Enter 'yes' or 'no'").strip() == 'yes'):

        play\_room(next\_room)

    else:

        play\_room(current\_room)

game\_state = INIT\_GAME\_STATE.copy()

start\_game()