```
 |0   |1   |2   |3   |4   |5   |6   |7   |8
#!/usr/bin/python
# The Legend of King Eldred
# Code Angel

import map
import game_items

# Define constants
MAP_MAX_ROW = 7
MAX_MAP_COLUMN = 7
START_ROOM = '01'
END_ROOM = '24'


def main():

    # Initialise variables
    map_row = 0
    map_column = 0

    new_row = 0
    new_column = 0

    game_over = False
    alive = True

    room = ''

    gold = 0

    # Set up map
    town_map = map.get_town_map()
    room_descriptions = map.get_room_descriptions()
    doors = map.get_doors()
    chests = map.get_chests()
    deadly_exits = map.get_deadly_exits()
    signs = map.get_signs()
    talking = map.get_talking()
    hints = map.get_hints()

    all_items = game_items.get_items()
 |0   |1   |2   |3   |4   |5   |6   |7   |8
```

```
|0  |1  |2  |3  |4  |5  |6  |7  |8

    # Set up command list
    command_list = ['help', 'hint', 'inv', 'take', 'drop', 'look', 'examine', 'open', 'unlock', 'read', 'talk']
    direction_list = ['n', 'north', 's', 'south', 'e', 'east', 'w', 'west']
    action_list = ['smash', 'dig', 'burn', 'give', 'drink', 'fill', 'say', 'pull', 'pierce', 'oil', 'turn']

    command_list.extend(direction_list)
    command_list.extend(action_list)

    room_items = []

    inventory = []

    display_intro()

    # Find the start room and display
    for row_num, row in enumerate(town_map):
        for col_num, next_room in enumerate(row):
            if next_room == START_ROOM:
                map_row = row_num
                map_column = col_num
                room = town_map[map_row][map_column]

                display_room_description(room_descriptions, room)
                room_items = get_room_items(room, all_items)
                chest_in_room = get_room_chest(room, chests)
                display_room_items(room_items, all_items, chest_in_room)

    while game_over is False:

        is_valid_room = True
        is_closed_door = False

        # Get the user command
        user_command = input('\nWhat do you want to do? ')
        is_valid_command = check_valid_command(command_list, user_command)

        # If the command is a direction, get the next room
        if user_command in direction_list:
            new_row = get_new_map_row(map_row, user_command)
            new_column = get_new_map_column(map_column, user_command)
|0  |1  |2  |3  |4  |5  |6  |7  |8
```

```python
            is_valid_room = check_valid_room(town_map, new_row, new_column)
            is_closed_door = check_for_door(room, user_command, doors)

            is_deadly_exit = check_for_safe_exit(room, user_command, deadly_exits)
            if is_deadly_exit is True:
                game_over = True
                alive = False

        # While the command is not valid, or room is not valid or door is closed or player has died
        # then display message and get another user command
        while is_valid_command is False or is_valid_room is False or is_closed_door is True and game_over is False:
            if is_valid_command is False:
                print("I don't know how to do that.")
            elif is_closed_door is True:
                print('The door is not open.')
            else:
                print("You can't go that way.")

            is_valid_room = True
            is_closed_door = False

            user_command = input('\nWhat do you want to do? ')
            is_valid_command = check_valid_command(command_list, user_command)

            # If the command is a direction, get the next room
            if user_command in direction_list:
                new_row = get_new_map_row(map_row, user_command)
                new_column = get_new_map_column(map_column, user_command)
                is_valid_room = check_valid_room(town_map, new_row, new_column)
                is_closed_door = check_for_door(room, user_command, doors)

                is_deadly_exit = check_for_safe_exit(room, user_command, deadly_exits)
                if is_deadly_exit is True:
                    game_over = True
                    alive = False

        # If user command is a valid direction, get the next room and display
        if user_command in direction_list:
            map_row = new_row
            map_column = new_column
            room = town_map[map_row][map_column]
```

```
|0  |1  |2  |3  |4  |5  |6  |7  |8

            display_room_description(room_descriptions, room)
            room_items = get_room_items(room, all_items)
            chest_in_room = get_room_chest(room, chests)
            display_room_items(room_items, all_items, chest_in_room)

            # If reached the end room it is game over
            if room == END_ROOM:
                game_over = True

        # Command: Display inventory
        elif user_command == 'inv':
            display_inventory_description(inventory, all_items, gold)

        # Command: Look
        elif user_command == 'look':
            room_items = get_room_items(room, all_items)
            chest_in_room = get_room_chest(room, chests)
            display_room_description(room_descriptions, room)
            display_room_items(room_items, all_items, chest_in_room)

        # Command: Help
        elif user_command == 'help':
            display_all_commands()

        # Command: Hint
        elif user_command == 'hint':
            gold = get_hint(room, hints, gold)

        # Command: Take
        elif 'take' in user_command:
            take_item(user_command, room_items, all_items, inventory)

        # Command: Drop
        elif 'drop' in user_command:
            drop_item(user_command, room_items, all_items, inventory, room)

        # Command: Open or unlock door
        elif 'door' in user_command:
            manage_door(doors, room, user_command, inventory, all_items)

|0  |1  |2  |3  |4  |5  |6  |7  |8
```

```python
        # Command: Open or unlock chest
        elif 'chest' in user_command:
            found_gold = manage_chest(chests, room, user_command, inventory, all_items)
            gold += found_gold
            if found_gold > 0:
                print('You now have ' + str(gold) + ' gold pieces.')

        # Command: Examine item
        elif 'examine' in user_command:
            chest_in_room = get_room_chest(room, chests)
            examine_item(room, user_command, room_items, all_items, inventory, chest_in_room)

        # Command: Read
        elif 'read' in user_command:
            read_sign(signs, room)

        # Command: Talk
        elif 'talk' in user_command:
            talk_to_character(talking, room)

        # Carry out some other action
        else:
            death = carry_out_action(user_command, inventory, room_items, all_items, room,
                                     deadly_exits, room_descriptions, doors, chests, signs)
            if death is True:
                game_over = True
                alive = False

    # Game is over - if the player is alive they win
    if alive is True:
        print()
        print('You have conquered Wildemoor and solved The Legend of King Eldred.')
        if 'crown' in inventory:
            print('You found the Crown of Anquira which brings untold power and wealth to whomever wears it.')
        else:
            print('You failed to find the Crown of Anquira.')

        print('You leave Wildemoor with ' + str(gold) + ' gold pieces.')
    else:
        print('And so The Legend of Kind Eldred takes the life of another brave explorer...')

```

```
|0  |1  |2  |3  |4  |5  |6  |7  |8

# Test if the user command is in the command list
# Return True if it is, False if not
def check_valid_command(command_list, user_command):

    valid_command = False

    for command_word in user_command.split():
        if command_word in command_list:
            valid_command = True

    return valid_command


# Head north - subtract 1 from row
# Head south - add one 1 row
def get_new_map_row(new_row, direction):

    if direction == 'n' or direction == 'north':
        new_row -= 1
    elif direction == 's' or direction == 'south':
        new_row += 1

    return new_row


# Head east - add 1 to column
# Head west - subtract 1 from column
def get_new_map_column(new_col, direction):
    if direction == 'e' or direction == 'east':
        new_col += 1
    elif direction == 'w' or direction == 'west':
        new_col -= 1

    return new_col


# A room is valid if it is not off the left, right, top or bottom of the map
# And it is not 2 dashes ('--')
def check_valid_room(town_map, new_row, new_column):

|0  |1  |2  |3  |4  |5  |6  |7  |8
```

```
|0  |1  |2  |3  |4  |5  |6  |7  |8
    valid_room = True

    if new_row < 0:
        valid_room = False
    elif new_row > MAP_MAX_ROW - 1:
        valid_room = False
    elif new_column < 0:
        valid_room = False
    elif new_column > MAX_MAP_COLUMN - 1:
        valid_room = False
    elif town_map[new_row][new_column] == '--':
        valid_room = False

    return valid_room


# Print the room description to the console window
def display_room_description(room_descriptions, room):
    print('\n' + room_descriptions.get(room))


# get a list of items in the current room
def get_room_items(current_room, all_items):

    room_items = []

    for items_key, item in all_items.items():
        item_room = item.get('room')
        if item_room == current_room:
            room_items.append(items_key)

    return room_items


# Test if the room has a chest (True / False)
def get_room_chest(current_room, chests):

    has_chest = False

    for chests_key, chest in chests.items():
        chest_room = chest.get('room')
|0  |1  |2  |3  |4  |5  |6  |7  |8
```

```
|0  |1  |2  |3  |4  |5  |6  |7  |8
        if chest_room == current_room:
            has_chest = True

    return has_chest


# Display any item descriptions, and whether there is a chest or not
def display_room_items(room_items, all_items, chest_in_room):

    item_found = False
    for item_key in room_items:
        item = all_items.get(item_key)

        if item_found is False:
            print('You can see:')
            item_found = True
        item_description = item.get('description')
        print(item_description)

    if chest_in_room is True:
        print('chest')


# Take item
def take_item(user_command, room_items, all_items, inventory):

    item_in_inventory = False
    item_in_room = False
    take_item_dict_key = ''

    # Test if the item the player wants to take is actually in the room
    for item_key in room_items:
        item = all_items.get(item_key)
        item_name = item.get('name')
        if item_name in user_command:
            item_in_room = True
            take_item_dict_key = item_key

    # Test if the item the player wants to take is already in the player inventory
    for item_key in inventory:
            item = all_items.get(item_key)
|0  |1  |2  |3  |4  |5  |6  |7  |8
```

```python
|0  |1  |2  |3  |4  |5  |6  |7  |8
            item_name = item.get('name')
            if item_name in user_command:
                item_in_inventory = True

    # The item is in the room...
    if item_in_room is True:
        item = all_items.get(take_item_dict_key)
        item_name = item.get('name')
        item_description = item.get('description')
        item_can_be_taken = item.get('inv')

        # The user wants the item
        if item_name in user_command:

            # The item can be taken
            if item_can_be_taken is True:
                print('You take the ' + item_description + '.')

                # Add the item to the inventory
                inventory.append(take_item_dict_key)

                # Remove it from the current list of items in the room
                room_items.remove(take_item_dict_key)

                # Remove it from the map by setting the all_items room value to room 0
                all_items.get(take_item_dict_key)['room'] = '0'

            # The item cannot be taken
            else:
                print('You cannot carry that item.')

    # The item is already in the inventory
    elif item_in_inventory is True:
        print('You are already carrying that.')

    # The player has tried to take something not in the list of room items
    else:
        print('You cannot take that.')


# Drop item
|0  |1  |2  |3  |4  |5  |6  |7  |8
```

```
|0   |1   |2   |3   |4   |5   |6   |7   |8
def drop_item(user_command, room_items, all_items, inventory, room):

    item_in_inventory = False

    # Test if item to be dropped is in the player inventory
    for item_key in inventory:
        item = all_items.get(item_key)
        item_name = item.get('name')
        item_description = item.get('description')

        # The item is found in the inventory
        if item_name in user_command:
            item_in_inventory = True

            print('You drop the ' + item_description + '.')

            # Remove the item from the inventory list
            inventory.remove(item_key)

            # Add the item to the current list of items in the room
            room_items.append(item_key)

            # Set the room value of the item to the current room
            all_items.get(item_key)['room'] = room

    # The item is not in the inventory
    if item_in_inventory is False:
        print('You are not carrying that item.')


# Examine an item
def examine_item(room, user_command, room_items, all_items, inventory, chest_in_room):

    item_found = False

    # Loop through all of the items in the inventory and the current room
    for item_key in inventory + room_items:
        item = all_items.get(item_key)
        item_name = item.get('name')
        item_examine = item.get('examine')
        item_can_be_taken = item.get('inv')
|0   |1   |2   |3   |4   |5   |6   |7   |8
```

```
|0  |1  |2  |3  |4  |5  |6  |7  |8

        # If the user has asked to examine one of these items...
        if item_name in user_command:
            item_found = True

            # If it is an item which can be taken, display the full examine description
            if item_can_be_taken is True:
                print(item_examine)

            # If it is not an item which can be taken...
            else:
                item_discovered = item.get('discovered')
                secret_item = item.get('secret item')

                # If it is not a secret item
                if secret_item == 'none':

                    # Display the item examine description and current status
                    status = item.get('status')
                    display_status = item.get('display status')
                    if display_status is None:
                        status = ''
                    print(item_examine + status)

                # If the item is a secret item and it has not already been discovered
                elif item_discovered is False:

                    # Display the item examine description
                    print(item_examine)

                    # Add the item to the list of room items
                    room_items.append(secret_item)

                    # Set the room of the secret item to the current room
                    all_items.get(secret_item)['room'] = room
                    item['discovered'] = True

                    # Display the room items again so that the secret item is displayed
                    display_room_items(room_items, all_items, chest_in_room)

                # The secret item has already been discovered
|0  |1  |2  |3  |4  |5  |6  |7  |8
```

```
|0  |1  |2  |3  |4  |5  |6  |7  |8
                else:
                    print('There is nothing more to be discovered here.')

    if item_found is False:
        print("You can't examine that item.")


# Test to see if the exit direction chosen is safe or deadly
def check_for_safe_exit(room, direction, deadly_exits):

    deadly_exit = False

    # Get the first character of direction (e.g. north = n)
    direction = direction[:1]

    # Loop through all of the possible deadly exits
    for deadly_exit_key in deadly_exits:
        check_exit = deadly_exits.get(deadly_exit_key)
        exit_room = check_exit.get('room')

        # If a deadly exit room matches the current room
        if exit_room == room:
            room_safe = check_exit.get('safe')
            exit_direction = check_exit.get('direction')

            # If the exit matches the safe exit
            if direction == exit_direction:
                if room_safe is True:
                    deadly_exit = False

                # If not it is the deadly exit
                else:
                    deadly_exit = True
                    death_text = check_exit.get('death')
                    print(death_text)

    return deadly_exit


# Check if there is a door in the direction the player has chosen
def check_for_door(room, direction, doors):
|0  |1  |2  |3  |4  |5  |6  |7  |8
```

```
|0  |1  |2  |3  |4  |5  |6  |7  |8

    closed_door = False

    direction = direction[:1]

    for door_key in doors:
        door = doors.get(door_key)
        door_room = door.get('room')

        # There is a door in the current room - get its status
        if door_room == room:
            door_open = door.get('open')
            door_locked = door.get('locked')
            door_direction = door.get('direction')

            # If the door is locked or not open then it is closed and the player will not be able to pass
            # without first unlocking or opening
            if direction == door_direction:
                if door_locked is True or door_open is False:
                    closed_door = True

    return closed_door


# Manage doors - the player has attempted to do something with a door
def manage_door(doors, room, user_command, inventory, all_items):

    this_room_door = {}

    # Loop through all of the doors checking for doors in the current room
    for door_dict_key in doors:
        door = doors.get(door_dict_key)
        door_room = door.get('room')
        if door_room == room:
            this_room_door = door

    # A door in the current room
    if this_room_door:

        # Get the status of the door
        door_open = this_room_door.get('open')
|0  |1  |2  |3  |4  |5  |6  |7  |8
```

```
|0  |1  |2  |3  |4  |5  |6  |7  |8
        door_locked = this_room_door.get('locked')
        unlock_key = this_room_door.get('unlock key')

        # Player exmaines door
        if 'examine' in user_command:
            door_examine = this_room_door.get('examine')
            print(door_examine)

        # Player opens door
        elif 'open' in user_command:

            if door_open is True:
                print('The door is already open.')
            elif door_locked is True:
                print('The door is locked.')
            else:
                print('You open the door.')
                this_room_door['open'] = True

        # Player unlocks door
        elif 'unlock' in user_command:
            if door_locked is False:
                print('The door is already unlocked')
            else:

                # The player has the key required to unlock the door in their inventory
                if unlock_key in inventory:
                    unlock_key_record = all_items.get(unlock_key)

                    unlock_key_description = unlock_key_record.get('description')
                    print('You unlock the door with the ' + unlock_key_description + '.')
                    this_room_door['locked'] = False
                    inventory.remove(unlock_key)

                # A code is required to unlock this door
                elif unlock_key == 'code':
                    unlock_key_code = this_room_door.get('unlock code')

                    # Correct code
                    if unlock_key_code in user_command:
                        print('The code ' + unlock_key_code + ' unlocks the door.')
|0  |1  |2  |3  |4  |5  |6  |7  |8
```

```
|0  |1  |2  |3  |4  |5  |6  |7  |8
                        this_room_door['locked'] = False

                    # Incorrect code
                    else:
                        print('That code does not unlock the door.')

                # The player does not have the key required to unlock this door
                else:
                    print('You do not have a key to unlock this door.')

    # No door found in the current room
    else:
        print('There is no door here.')


# Manage chests - the player has attempted to do something with a chest
def manage_chest(chests, room, user_command, inventory, all_items):

    this_room_chest = {}

    found_gold = 0

    # Loop through all of the chests checking for chests in the current room
    for chest_dict_key in chests:
        chest = chests.get(chest_dict_key)
        chest_room = chest.get('room')
        if chest_room == room:
            this_room_chest = chest

    # A chest in the current room
    if this_room_chest:

        # Get the status of the chest
        chest_open = this_room_chest.get('open')
        chest_locked = this_room_chest.get('locked')
        unlock_key = this_room_chest.get('unlock key')

        # Player exmaines chest
        if 'examine' in user_command:
            chest_examine = this_room_chest.get('examine')
            print(chest_examine)
|0  |1  |2  |3  |4  |5  |6  |7  |8
```

```
|0  |1  |2  |3  |4  |5  |6  |7  |8

        # Player opens chest
        elif 'open' in user_command:

            if chest_open is True:
                print('The chest is already open.')

            elif chest_locked is True:
                print('The chest is locked.')

            # Player opens chest, set found_gold to gold in chest and set gold in chest to 0
            else:
                print('You open the chest.')
                this_room_chest['open'] = True
                found_gold = this_room_chest.get('gold')
                print('Inside the chest you find ' + str(found_gold) + ' gold pieces.')
                this_room_chest['gold'] = 0

        # Player unlocks chest
        elif 'unlock' in user_command:
            if chest_locked is False:
                print('The chest is already unlocked')

            else:
                # The player has the key required to open the chest in their inventory
                if unlock_key in inventory:
                    unlock_key_record = all_items.get(unlock_key)
                    unlock_key_description = unlock_key_record.get('description')
                    print('You unlock the chest with the ' + unlock_key_description + '.')
                    this_room_chest['locked'] = False
                    inventory.remove(unlock_key)

                else:
                    print('You do not have a key to unlock this chest.')

    else:
        print('There is no chest here.')

    return found_gold


|0  |1  |2  |3  |4  |5  |6  |7  |8
```

```python
# Read a sign
def read_sign(signs, room):
    sign = signs.get(room)
    visible = signs.get(room).get('visible')

    # If there is a sign in the room and it is visible, display the sign message
    if sign and visible is True:
        sign_message = sign.get('read')
        print(sign_message)
    else:
        print('There is nothing to read here.')


# Talk to character
def talk_to_character(talking, room):
    talk_details = talking.get(room)

    # If there is any character in the room who has something to say, display their message
    if talk_details:
        says = talk_details.get('says')
        print(says)
    else:
        print('There is noone to talk to here.')


# Carry out all of the other special actions
def carry_out_action(user_command, inventory, room_items, all_items, room, deadly_exits,
                     room_descriptions, doors, chests, signs):

    action_completed = False
    death = False

    # Smash the bottle with hammer (need bottle and hammer in inventory)
    if 'smash' in user_command and 'bottle' in user_command and 'hammer' in user_command:
        if 'bottle' in inventory + room_items and 'hammer' in inventory + room_items:
            action_completed = True
            print('The hammer smashes the bottle.')
            print('A key falls to the floor.')
            inventory.remove('bottle')
            room_items.append('key 1')
            all_items.get('key 1')['room'] = room

    # Dig the dirt with the spoon (need spoon in inventory and be in the room that has dirt)
    if 'dig' in user_command and 'spoon' in user_command and 'dirt' in user_command:
        if 'spoon' in inventory + room_items and 'dirt' in room_items:
            action_completed = True
            print('You dig the dirt with the spoon for what seems like hours.')
            print('Eventually you uncover a rock. It looks as though it has something carved into it.')
            room_items.append('rock')
            all_items.get('rock')['room'] = room

    # Burn straw with torch (need torch in inventory and be in the room that has straw)
    if 'burn' in user_command and 'straw' in user_command and 'torch' in user_command:
        if 'torch' in inventory + room_items and 'straw' in room_items:
            action_completed = True
            print('You toss the lit torch onto the bed of straw.')
            print('The fire takes hold almost instantly and within minutes all the vipers are dead.')
            all_items.get('straw')['room'] = '00'
            deadly_exits.get('vipers')['safe'] = True
            room_descriptions[room] = 'It smells really bad in this tunnel, like burned sausages.  \
The floor and most of the walls are blackened with ash.'
            inventory.remove('torch')

    # Give the bone to the dog (need bone in inventory and be in the room that has the dog)
    if 'give' in user_command and 'dog' in user_command and 'bone' in user_command:
        if 'bone' in inventory + room_items and 'dog' in room_items:
            action_completed = True
            print('You throw the bone in the direction of the large black dog. The dog gets up slowly and paces \
over to the bone.')
            print('It sniffs twice at the bone then picks it up in its enormous mouth and wanders off.')
            all_items.get('dog')['room'] = '00'
            deadly_exits.get('dog')['safe'] = True
            room_descriptions[room] = 'To the east are a set of marble steps which lead up to an old \
oak door. The door is open. There is a small gap leading to a cave in the west.'
            inventory.remove('bone')

    # Drink from fountain (need to be in the room that has the fountain)
    if 'drink' in user_command and 'fountain' in user_command:
        if 'fountain' in room_items:
            action_completed = True
            print('You drink the water. It is crystal clear, ice cold and very refreshing.')

```

```
|0  |1  |2  |3  |4  |5  |6  |7  |8
    # Fill bucket (need bucket in inventory and be in room that has fountain)
    if 'fill' in user_command and 'bucket' in user_command:
        if 'bucket' in inventory + room_items and 'fountain' in room_items:
            action_completed = True
            print('You fill the bucket with ice cold crystal clear water from the fountain.')
            all_items.get('bucket')['status'] = 'filled'
            all_items.get('bucket')['description'] = 'bucket of water'
            all_items.get('bucket')['examine'] = 'An old bucket filled with ice cold water.'
            # inventory.remove('bucket')
            # inventory.append('bucket of water')

    # Drink from lake (need to be in room that has lake)
    if 'drink' in user_command and 'lake' in user_command:
        if 'lake' in room_items:
            action_completed = True
            print('You cup your hands together and drink some of the water. It tastes bitter.')
            print('You start to feel dizzy and everything around you is spinning.')
            print('You lie down and instantly fall into a deep, deep sleep.')
            print('You dream of kings and gold.')
            print('In your dream King Eldred appears as a vision and speaks some mysterious words.')
            print("'Learn the code of Python and all will be well with the world.'")
            print('You wake sometime later, dazed and confused.')

    # Give water to horse (need to have bucket filled with water and be in the room that has the horse)
    if 'give' in user_command and 'horse' in user_command and 'water' in user_command:
        if 'bucket' in inventory + room_items and 'horse' in room_items:
            action_completed = True

            if all_items.get('bucket').get('status') == 'filled':
                print('You hold the bucket of water out to the horse.')
                print('It eyes you suspiciously at first, and the bends its neck to drink from the bucket.')
                print('As it drinks, a piece of parchment falls from underneath its saddle.')
                inventory.remove('bucket')
                room_items.append('parchment')
                all_items.get('parchment')['room'] = room
            else:
                print('The bucket is empty.')

    # Give toy to Andrid (need to have toy in inventory and be in the room that has Andrid)
    if 'give' in user_command and 'toy' in user_command and 'andrid' in user_command:
        if 'toy' in inventory + room_items and 'andrid' in room_items:
|0  |1  |2  |3  |4  |5  |6  |7  |8
```

```
|0  |1  |2  |3  |4  |5  |6  |7  |8
            action_completed = True
            print("Andrid turns to you and smiles. 'Thank you so much. This is my favourite toy.'")
            print('"Do you want this old bone key?", she adds, "It is of no use to me!"')
            print('Andrid hands you an old bone key.')
            inventory.remove('toy')
            inventory.append('key 7')

    # Give egg to baker (need to have egg in inventory and be in room that has the baker)
    if 'give' in user_command and 'egg' in user_command and 'baker' in user_command:
        if 'egg' in inventory + room_items and 'baker' in room_items:
            action_completed = True
            print('The baker says:"This is fantastic - now I can bake my cake.')
            print("It's for the banker, it's her birthday today.")
            print("Don't tell her I said this but she looks about 100 years old!")
            print('Wait! Here, take this as a reward for your troubles. It may come in handy..."')
            print('The baker hands you a floor plan drawn onto a large sheet of parchment.')
            inventory.remove('egg')
            inventory.append('floor plan')

    # Say king to guard (need to be in room with the guard)
    if 'say' in user_command and 'guard' in user_command and 'king' in user_command:
        if 'guard' in room_items:
            action_completed = True
            print('The guard pulls a hand-crafted silver key from his chest pocket, turns and slowly unlocks \
the door behind him.')
            print("'You may enter', he growls as he stands aside")
            doors.get('king')['locked'] = False
            doors.get('king')['open'] = True
    elif 'say' in user_command and 'guard' in user_command:
        if 'guard' in room_items:
            action_completed = True
            print("The guard moves his hand to the scimitar tucked into he belt and mutters 'No talk unless
password.'")

    # Pull gold lever (need to be in room that has the gold lever)
    if 'pull' in user_command and 'gold lever' in user_command:
        if 'gold lever' in room_items:
            action_completed = True
            print('You pull the gold lever. For a brief second nothing happens.')
            print('You hear a click below you and so you look down. \
You realise you are standing on what appears to be a trap door.')
|0  |1  |2  |3  |4  |5  |6  |7  |8
```

```
|0  |1  |2  |3  |4  |5  |6  |7  |8
            print('Before you can react the trap door swings open and you are sent tumbling into a dark dungeon.')
            print("With no food or water in here you will not last for long. But long enough to reflect on the \
words you once saw on a sign: 'Water over earth, young over old, and silver over bronze and gold.'")
            death = True

    # Pull silver lever (need to be in room that has the silver lever)
    if 'pull' in user_command and 'silver lever' in user_command:
        if 'silver lever' in room_items:
            action_completed = True
            print('You pull the silver lever. For a brief second nothing happens.')
            print('You hear a click coming from inside the chest. It sounds as if the chest has somehow unlocked.')
            chests.get('lever room')['locked'] = False

    # Pierce oil can with nail (need to have oil can and nail in inventory)
    if 'pierce' in user_command and 'oil can' in user_command and 'nail' in user_command:
        if 'oil can' in inventory + room_items and 'nail' in inventory + room_items:
            action_completed = True
            all_items.get('oil can')['status'] = 'pierced'
            all_items.get('oil can')['description'] = 'pierced oil can'
            all_items.get('oil can')['examine'] = 'The oil can has been pierced and a small amount \
of oil is running out of the hole.'
            print('With some effort you manage to pierce a hole in the oil can. A small amount of oil \
seeps out of the hole.')

    # Oil lever or oil cogs (need to have pierced oil can and be in room that has the marble lever)
    if 'oil' in user_command and ('lever' in user_command or 'cogs' in user_command):
        if 'oil can' in inventory + room_items and 'marble lever' in room_items:
            action_completed = True
            oil_can_status = all_items.get('oil can').get('status')
            if oil_can_status == 'pierced':
                print('You pour a few drops of oil onto the rusted cogs of the marble lever.')
                all_items.get('marble lever')['status'] = 'oiled'
            else:
                print('The oil can is sealed and so no oil will come out.')

    # Pull lever (need to be in room with marble lever)
    if 'pull' in user_command and 'lever' in user_command:
        if 'marble lever' in room_items:
            action_completed = True
            lever_status = all_items.get('marble lever').get('status')
            if lever_status == 'rusted':
|0  |1  |2  |3  |4  |5  |6  |7  |8
```

```
                print('You try pulling the marble lever with all your strength but it will not shift. \
The cogs are badly rusted.')
            else:
                print('You try pulling the marble lever. At first it will not move.')
                print('But then slowly the freshly oiled cogs begin to turn and the lever moves.')
                print('The marble lid of the sarcophogus slides on to the floor providing a bridge across the \
deadly spikes.')
                print('')
                print('A skeleton lies inside the sarcophogus.')
                print('A beautiful marble chest sits at the feet of the skeleton.')
                print('A golden crown studded with large emeralds and diamonds sits on the skull of the skeleton.')
                print('There is a plaque on the side of the sarcophogus.')

                deadly_exits.get('spikes')['safe'] = True
                chests.get('marble')['room'] = room
                signs.get(room)['visible'] = True
                all_items.get('crown')['room'] = room
                room_items.append('crown')

                room_descriptions[room] = 'You are standing in the tomb of King Eldred. \
\nThere are 3 dials on the wall: the first copper, the second bronze and the third silver.\
\nEach dial has 3 settings: sun, crescent moon, and star.\
\nTo the east is a solid marble door.\
\nThe marble sarcophogus lid forms a safe bridge to cross a pit layered with spikes.'

    # Turn dials
    if 'turn' in user_command and 'dial' in user_command:
        if 'copper dial' in room_items:
            action_completed = True
            valid_dial = False
            if 'copper' in user_command or 'first' in user_command:
                if 'moon' in user_command:
                    all_items.get('copper dial')['status'] = 'moon'
                    print('The copper dial is now set to: moon')
                    valid_dial = True
                elif 'sun' in user_command:
                    all_items.get('copper dial')['status'] = 'sun'
                    print('The copper dial is now set to: sun')
                    valid_dial = True
                elif 'star' in user_command:
                    all_items.get('copper dial')['status'] = 'star'
```

```
|0  |1  |2  |3  |4  |5  |6  |7  |8
                    print('The copper dial is now set to: star')
                    valid_dial = True

            elif 'bronze' in user_command or 'second' in user_command:
                if 'moon' in user_command:
                    all_items.get('bronze dial')['status'] = 'moon'
                    print('The bronze dial is now set to: moon')
                    valid_dial = True
                elif 'sun' in user_command:
                    all_items.get('bronze dial')['status'] = 'sun'
                    print('The bronze dial is now set to: sun')
                    valid_dial = True
                elif 'star' in user_command:
                    all_items.get('bronze dial')['status'] = 'star'
                    print('The bronze dial is now set to: star')
                    valid_dial = True

            elif 'silver' in user_command or 'third' in user_command:
                if 'moon' in user_command:
                    all_items.get('silver dial')['status'] = 'moon'
                    print('The silver dial is now set to: moon')
                    valid_dial = True
                elif 'sun' in user_command:
                    all_items.get('silver dial')['status'] = 'sun'
                    print('The silver dial is now set to: sun')
                    valid_dial = True
                elif 'star' in user_command:
                    all_items.get('silver dial')['status'] = 'star'
                    print('The silver dial is now set to: star')
                    valid_dial = True

            if valid_dial is True:
                copper_dial = all_items.get('copper dial').get('status')
                bronze_dial = all_items.get('bronze dial').get('status')
                silver_dial = all_items.get('silver dial').get('status')

                if copper_dial == 'moon' and bronze_dial == 'moon' and silver_dial == 'sun':
                    print('You hear a series of sharp clicks, followed by a low rumble as the solid marble door
slowly \
swings open.')
                    print('Bright sunshine spills into the room from the exit to the east.')
|0  |1  |2  |3  |4  |5  |6  |7  |8
```

```
|0  |1  |2  |3  |4  |5  |6  |7  |8
                    doors.get('exit')['locked'] = False
                    doors.get('exit')['open'] = True

            else:
                print('You cannot turn the dial to that setting')

    # The special command did not work
    if action_completed is False:
        print("It is not possible to do that. Perhaps you don't have all the items required, \
or perhaps it just makes no sense!")

    # Did carrying out the command result in the player's death?
    return death


# Display inventory and gold
def display_inventory_description(inventory, all_items, gold):

        print('You are carrying:')
        for item_key in inventory:
            item = all_items.get(item_key)
            item_description = item.get('description')
            print(item_description)

        print('Gold: ' + str(gold))


# Get a hint
def get_hint(room, hints, gold):

    hint = hints.get(room)

    # If the current room has a hint
    if hint:
        hint_text = hint.get('hint')
        hint_cost = hint.get('cost')

        # If the player can afford the hint, display it
        if gold >= hint_cost:
            print(hint_text)
            gold -= hint_cost
|0  |1  |2  |3  |4  |5  |6  |7  |8
```

```
            print('The cost of that hint was ' + str(hint_cost) + ' gold. You have ' + str(gold) + ' remaining.')

        else:
            print('The hint costs ' + str(hint_cost) + ' gold but you only have ' + str(gold) + ' gold.')

    else:
        print('Sorry, no hints here.')

    return gold


# Display the game introduction
def display_intro():
    print('')
    print('Welcome to The Legend of King Eldred')
    print('------------------------------------')
    print('')
    print("To play the game, just type a command e.g. 'north', 'look', 'give carrot to donkey'.")
    print("To see the full list of commands available, type 'help'.")
    print("If you get really stuck, type 'hint'. But hints are not free...")
    print('')
    print('Your task is to explore the village of Wildemoor and its icy dungeons to find the hidden treasure.')
    print('Your task is also to stay alive!')
    print('')
    print('------------------------------------')
    print('')
    print('For many years Wildemoor was ruled by King Eldred. Eldred was a popular king who was very ')
    print('generous to his subjects. When he died, King Eldred was buried in a sealed tomb beneath Wildemoor.')
    print('It is said he was buried with great treasures including his crown which, according to ')
    print('legend brings untold power and wealth to the wearer.')


# Display all commands available in the game
def display_all_commands():
    print('The commands you can use in Legend of King Eldred are:')
    print('north, south, east, west (or n, s, e, w)')
    print('help - displays all possible commands')
    print('inv - inventory')
    print('hint - get a hint for a small fee')
    print('take <item> - you never know when something might come in handy')
    print('drop <item> - maybe you know this will not come in handy')
    print('look - review what is in the current location')
    print('examine <item> - it is always a good idea to examine every item in Wildemoor')
    print('open <item> - good for chests and doors, but they have to be unlocked first')
    print('unlock <item> - unlock a door or a chest, but you will need the correct key or code')
    print('read sign - you never know what useful information will be displayed on a sign')
    print("talk to <person> - it's good to talk")
    print('smash <item> with <item> - occasionally a little bit of vandalism is necessary')
    print('dig <item> with <item> - you dig...?')
    print('burn <item> with <item> - handy for the budding arsonists out there')
    print('give <item> to <person> - you never know what you might get back')
    print('drink - being an adventurer can be thirsty work')
    print('fill <item> with <something>')
    print('say <something> to <person>')
    print('pull <item> - handy for levers, all levers in games are there to be pulled')
    print('pierce <item> with <item> - make a hole in something with a sharp object')
    print('oil <item> - always solves those rusty problems')
    print("turn <dial> to <setting> - no point in having a dial if you can't turn it")


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