ICS 31 - Project 2

Adventure Game

**Due: Sunday June 8, 2025 11:59pm**



# 1. Overview

You will develop a text-based adventure game that enables interactive exploration of a game world described in a JSON file. The game will:  
a) Reads a description of the game world from a JSON file (adventure.json by default).  
b) Allows a user to navigate through rooms by reading and writing to the console/terminal.

The starter ZIP (project2.zip) containing:  
 - adventure.py: Starter game script  
 - adventure.json: Sample world configuration  
You will need to create a custom.json file that describes a unique game world of your design.

Clarifications:  
- adventure.json: Provided sample world  
- custom.json: Your uniquely designed world (must be G-rated)  
- Signup for a unique theme via the [Google Spreadsheet](https://docs.google.com/spreadsheets/d/1IKx9tqhfP_zkb5fUE1UAykFoQs2LQ9SFps-WDLEDVSQ/edit?gid=0#gid=0) (first-come, first-served)

# 2. Game World Structure (JSON)

Your JSON file will contain nested dictionaries. When you load the JSON file into your adventure.py, you will get a dictionary structure like this:

Top-level dictionary key: Room name

Associated value: A dictionary containing:

1. "text": A string describing that room.
2. "moves": Another dictionary, where each key is a command (e.g., north, jump, pick up) and the value is the name of the room that the command navigates to.

Clarification:

Valid directions are not limited to compass directions. There can be any action (e.g., "jump", "run") that sends the player to another room.

Example snippet of a single room (hypothetical):

***"ROOM\_NAME": {  
 "text": "Room description",  
 "moves": {  
 "action": "NEXT\_ROOM"  
 },  
 "objects": [  
 {  
 "name": "item name",  
 "type": "normal" | "special"  
 }  
 ]  
}***- Room Keys: Top-level keys represent room names.  
- "text": Description displayed to the user.  
- "moves": Dictionary of valid actions and corresponding destination rooms.  
- "objects" (optional): List of items in the room.

# 3. Game Interface

## Output

When the game starts, it will begin in a specific room (called START). Your code should:

1. Print the current room’s description.
2. Print a list of valid directions (or actions) that the player can take.
3. Add a blank line after these two items.

If the room is called FINISH, then the game should print “You win!!!” plus any custom message relevant to your theme, and then end.

## Input

Reading Player Input:

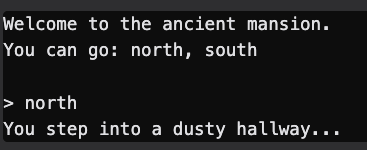
* After displaying the room’s information, read one line of input from the user.

Handling Player Choices:

* If the user types "quit" or "exit", end the program.
* If the user types a valid move (regardless of case differences), navigate to the corresponding room and repeat the output steps.
* If the user input does not match a valid move for the current room, reprint the list of valid options and prompt again.

# 4. Example

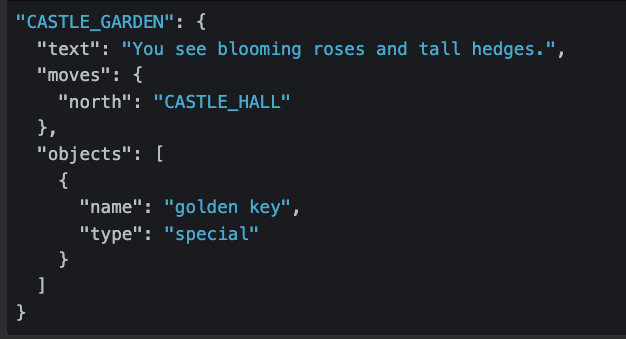
A demonstration of the game’s flow (using sample data) was provided in class.



# 5. Functional Requirements

Below are the technical requirements for this project:

1. Choose/Propose a Theme
   1. Select a theme for your adventure in the provided Google doc. Examples include Star Wars settings, fantasy castles, etc.
2. Create a Unique World
   1. Use at least 10 distinct locations.
   2. Each location must be connected logically (with directions/actions leading to other locations).
   3. Draw a map or flowchart showing how the rooms connect.
      1. Use arrows on the connections to clearly label which way you can traverse
3. Load JSON Data
   1. Your program must load room data from the JSON file at runtime (initially adventure.json, then your custom.json).
4. Display Description/Options
   1. Correctly display the text for the current room, followed by available directions/actions.
5. User Navigation
   1. Move between rooms based on valid user commands until reaching FINISH.
6. Ending the Game
   1. End the game when the user enters the FINISH room or types "quit"/"exit".
7. Graceful Handling of Bad Commands
   1. If the user enters an invalid command, re-display the valid moves.
8. Random Start Room
   1. Exclude the FINISH room from random selection.
   2. This means each playthrough starts in a different (but valid) location.
9. User Profiles and Save Feature
   1. Ask for a username when the game starts.
   2. Save each user’s current location so that when they return, they start where they left off.
   3. If a user previously won, start them again from a random valid location (excluding FINISH).
   4. Save the user’s current location in a file called **save.json**
10. Add 5 Types of Objects
    1. Rooms can contain objects. Each room’s “objects” key will have a list of objects.
    2. “normal” objects stay in the room; “special” objects can be picked up by the player.
    3. Update your room description function to list objects found (e.g., “You see a golden key.”).
    4. If multiple items exist, print them in a single sentence (e.g., “You see item1 and item2 and item3.”).



# 6. Required Functions (Autograder)

You must implement the following helper functions as specified

move\_user(data, current, move)  
- Parameters:  
 - data: game world dictionary  
 - current: current room name  
 - move: user’s action  
- Returns: new room name (string)  
  
describe(data, current)  
- Parameters:  
 - data: game world dictionary  
 - current: current room name  
- Returns: room description (string)  
  
These functions must match exactly for compatibility with the autograder. They should integrate neatly with your existing logic.

# 6. Development Tips

1. Use the provided adventure.json to develop and test your core game logic first.
2. Once your code is working, swap in custom.json for your final world.
3. Keep your code organized and use functions to handle specific tasks (e.g., reading JSON data, printing room data, handling user input).
4. Test incrementally—implement one feature at a time and verify it before adding the next.

# 7. Submission & Academic Integrity

- This is an individual assignment.  
- Discussion and idea sharing is encouraged, but code must be your own.

Your submission should include the following 3 files:

* adventure.py
* custom.json
* map.pdf

**Make sure any room has a possible path to the FINISH, and that FINISH is a string that holds the name of the room which is designated as goal for your game.**