CSC111 Winter 2025 Project 1

Grace Wang & Phoebe Kuang

February 12, 2025

Running the game

We should be able to run your game by simply running `adventure.py`. If you have any other requirements (e.g., installing certain modules), describe them here. Otherwise, skip this section.

Game Map

Example game map below (edit it to show your actual game map):

```
Note:
- 11 can only be accessed from 6 with a PRESTO card item

- ids in brackets are upper levels of locations

 -1   -1   -1    -1     7(70)  -1
 -1    1   2(20)  4     8      10
 -1   -1   3(30)  5     9      -1
 -1   -1   -1     6    -1      -1
 11   -1   -1    -1    -1      -1
```

Starting location is: 1

Game solution

List of commands:
- "go east" - "go upstairs" - "pick up: key" - "go downstairs" - "go east" - "go east" - "talk to sadia" - "go north" - "go to dorm" - "get usb drive" - "2" - "pick up: usb drive" - "go downstairs" - "go south" - "go south" - "pick up: mug" - "go west" - "go west" - "go upstairs" - "get laptop charger" - "1" - "1" - "1" - "pick up: laptop charger" - "go downstairs" - "go east" - "go south" - "pick up: presto card" - "get on the streetcar" - "buy potion" - "go back to campus" - "go north" - "go west" - "go north" - "go west" - "put down items to submit work"

Note 1: Some key commands (such as pick up, drop and conditional commands related to special events) in our game are not stored in the `available_command` attribute of game locations, and therefore cannot be simulated in `proj1_simulation.py`. The winning simulation we provided is a simplified version of the above routine that only involves traveling between the locations. The player will need to type the above inputs in order, when running adventure.py, to win the game.

Note 2: The number inputs in the above walkthrough is associated with our puzzle. In the first puzzle we are encountering in this routine, the player can guess for a maximum of three times, and therefore we

have three inputs. (It is possible to win the puzzle within three guesses, so the rest of the three inputs will be invalid inputs, which does not affect the game process)

Lose condition(s)

How to lose the game: Run out of moves (set at 50). After the 50th non-menu command is made, the game will come to a stop.

List of commands: "go east", "go east", "go east", "go north", "go south", "go west", "go west", "go west", "go east", "go east" "go east", "go north", "go south", "go west", "go west", "go west", "go east", "go east" "go east", "go north", "go south", "go west", "go west", "go west", "go east", "go east" "go east", "go north", "go south", "go west", "go west", "go west", "go east", "go east" "go east", "go north", "go south", "go west", "go west", "go west", "go east", "go east" "go east", "go north", "go south", "go west", "go west", "go west", "go east", "go east"

Which parts of your code are involved in this functionality: The last if-elif-else statement in `adventure.py`,

Inventory

1. All location IDs that involve items in the game: 1, 2, 3, 4, 6, 7, 9, 11
2. Item data:
   Item start location ID: the ID of location where this item can be collected for the first time.
   Item target location ID: the ID of location where this item can be used. (may be the same as start location ID)
   (a) For Item 1:
   - Item name:"mug"
   - Item start location ID: 9
   - Item target location ID: 1 (for submission)

   (b) For Item 2:
   - Item name: "usb drive"
   - Item start location ID: 70 (appears after solving puzzle)
   - Item target location ID: 1 (for submission)

   (c) For Item 3:
   - Item name: "laptop charger"
   - Item start location ID: 30 (appears after solving puzzle)
   - Item target location ID: 1 (for submission)

   (d) For Item 4:
   - Item name: "potion"
   - Item start location ID: 11
   - Item target location ID: 1 (for submission)

   (e) For Item 5:
   - Item name: "hotdog"
   - Item start location ID: 4
   - Item target location ID: 4 (consumed after buying)

   (f) For Item 6:
   - Item name: "key"

- Item start location ID: 2
- Item target location ID: 70 (used to open the door)

(g) For Item 6:

- Item name: "presto card"
- Item start location ID: 6 (used to get on streetcar)
- Item target location ID: 6

3. Exact command(s) that should be used to pick up an item (choose any one item for this example), and the command(s) used to use/drop the item (can copy the list you assigned to `inventory_demo` in the `project1_simulation.py` file)

   "mug"
   - pick up: "go east", "go east", "go east", "go south", "pick up: mug"
   - drop: "drop: mug" (needs to have "mug" in inventory)

4. Which parts of your code (file, class, function/method) are involved in handling the `inventory` command:

   `game_entities.py` - `Player` class - `inventory` attribute & `Player` methods

Score

1. Briefly describe the way players can earn scores in your game. Include the first location in which they can increase their score, and the exact list of command(s) leading up to the score increase:

   Player can earn score by unlocking a new location for the first time, picking up items, solving puzzles and triggering a special event.

2. Copy the list you assigned to `scores_demo` in the `project1_simulation.py` file into this section of the report:
   "go east", "go east", "go east", "go south", "go north", "go under the bridge", "ford, ford, teleport"
   (if we run it in `adventure.py`, the expected score should be 80)

3. Which parts of your code (file, class, function/method) are involved in handling the `score` functionality:
   `game_entities.py` - `Player` class - `score` attribute

Enhancements

1. Enhancement #1

   - Special teleport event at Location 10. The player can choose to be teleported to any basic locations (1 - 9).
   - Medium complexity
   - This special event required us to call an added helper method `ford_ford_teleport()` in `Adventure.py` which takes user input as the location id for next event, uses an added `print_basic_locations()` and `all_location_ids()` methods in the AdventureGame Class.
   - demo: ["go east", "go east", "go east", "go south", "go north", "go under the bridge", "ford, ford, teleport"]

2. Enhancement #2

- Extra puzzle before retrieving item: `lying_backpacks_game()` and `shuffling_drawers_game()` in `AdventureGame` class
- Medium complexity
- We implemented mini games inside our game to keep players engaged. We ran into difficulty when trying to disable interactions with the item when puzzle is not solved. We eventually added helper functions to delete and add commands to locations so that we can handle conditional appearances of items in locations.
- demo: ["go east", "go upstairs", "go upstairs", "pick up: key", "go downstairs", "go east", "go east", "go north", "go to dorm", "get usb drive", "2", "pick up: usb drive"]

3. Enhancement #3

- Complex puzzle: retrieving mug item at Location 11.
- low complexity: This involves using the "go [direction]" commands in `adventure.py`. The action "get on the street car" involves the added `get_item()` method in the `AdventureGame` class
- Player needs to go to Myhal (id 9) to pick up the broken mug, then go to College St (id 6) to pick up the presto card item, so that they can board the streetcar (special action: "get on the streetcar") from id 6 to get to T&T (id 11) to pick up the potion to fix the mug.
- demo: ["go east", "go east", "go east", "go south", "pick up: mug", "go west", "go south", "pick up presto card", "get on the streetcar", "buy: potion"]

4. Enhancement #4

- Puzzle for retrieving laptop charger
- high complexity: We used added helper methods `talk_to_sadia()` in `adventure.py`; `shuffling_drawers_game()`, `get_location()`, and `add_location_command()` and from the `AdventureGame` class; and `add_item()` from the `Location` class.
- In order for "go upstairs" option to appear at Location 3, player needs to first go to Location 8 and engage with the option "talk with sadia". Then, when the player go back to Location 3, "go upstairs" can be chose and there will be a puzzle to solve. After the puzzle is solved, laptop charger can be retrieved.
  Similar to enhancement #2, we use helper functions that can add commands to a certain location to handle this.
- demo: ["go east", "go east", "go east", "talk to sadia", "go under the bridge", "ford, ford, teleport", "3", "go upstairs", "get laptop charger", "1", "1", "1", "pick up: laptop charger"]

5. Enhancement #5

- Undo item picking and dropping. In the game, items picking and dropping can also be reversed by the undo command.
- high complexity
- We modified our `game.ongoing` while loop structure so that all commands (except menu commands) lead to a new event. We added a `item_involved attribute` for the `Event` class to distinguish the item-involved events. Then, in our `undo()` helper function, we reverse "pick up" and "drop" using the added `get_location()` method from the `AdventureGame` class, `get_inventory_item()` method from the `Player` class, `pick_up()` and `drop()` helper functions in `adventure.py`, and the `remove_last_event()` method from the `EventList` class to undo the command.

- undo pick up demo: ["go east", "go upstairs", "pick up: key", "undo"]
- undo drop/pick up demo: ["go east", "go upstairs", "pick up: key", "undo", "pick up: key", "drop: key", "undo", "inventory"]

6. Enhancement #6

   - Special event - buy hotdog at location 4. The event will increase the player's `moves_left` instance attribute.
   - low complexity: we use added helper method `buy_hotdog` in `adventurepy` which calls an added `remove_location_command` in the `AdventureGame` class. We created an instance attribute `moves_left` for each Player instance. We used an added `get_item()` in the `AdventureGame`.
   - demo: ["go east", "go east", "buy hotdog"]