Cardiff University
School of Computer Science & Informatics

CM1101 — Improving the Game

LAB EXERCISE 5

9 October 2014

Instructions: This set of exercises will guide you towards improving your adventure game. We will implement a more advanced parser for the player's input and also implement the concept of items and inventory.

If you are still working on the previous sets of exercises (Lab 3 about movement in the game and Lab 4 about Git) — do not worry, keep working on them at your own pace; start doing this set of exercises once you finish Lab 3 and Lab 4.

Remember, the lab tutors are here to help. If you get stuck — do not be shy, raise your hand and ask for advice. It is also ok to discuss the solutions with your peers (these labs are not assessed!), however make sure you understand everything by yourself.

Good luck!

1. Download `game2_template.zip` from Learning Central and unzip it into a folder on your `H:` drive. This template improves upon the previous version of the game. I have made some minor structural changes (see Appendix for the summary), but otherwise the main logic of the game remains almost the same.

   The file `items.py` now contains the definitions of items one may encounter in the game (either carried by the player, or lying around in the rooms). Just like rooms, items are dictionaries, with fields `"id"`, `"name"`, and `"description"`.

   Rooms now have an additional field, `items`, which is a list of items found in the room. Similarly, `inventory` (a global variable in module `player.py`) is a list of items carried by the player.

2. Update the function `normalise_input` (it is now in `parser.py`) so that it returns a *list* of words from the player's input. It should, as before, strip punctuation and whitespace. Additionally, it should now remove all "unimportant" words. See the comments and examples for `normalise_input` for more information.

   To achieve this, first complete a helper function `filter_words` which takes a list of words `words` and returns a copy of the list from which all words provided in the list `skip_words` have been removed.

3. Complete the function `list_of_items` (in `game.py`). It should, given a list of items, produce a string containing a comma-separated list of item names.

4. Using the function `list_of_items`, complete the functions `print_room_items` and `print_inventory_items` according to their respective documentation.

5. Update the function `print_room` (formerly known as `display_room`) so that it also prints a list of items found in the room.

6. Update the function `print_menu` so that it now prints the list of possible `take` and `drop` actions, in addition to the available exits (these have now been transformed into `go` actions). See the documentation in the comments and examples for more information.

7. I have updated the `menu` and the `main` functions slightly. The new modus operandi of the main game loop is as follows:
   - The current situation is displayed using `print_room` (which now additionally prints the items in the room) and `print_inventory_items` which shows what the player is carrying.
   - The list of actions is shown to the player using the updated `menu` function, which also takes the player's input and normalises it (but does not check for correctness).
   - Finally, the `execute_command` function tries to make sense of and execute the player's command.
8. The all-important `execute_command` function has been provided for you. It takes the player's input (as a list of words) and, depending on the first word, which is treated as the name of the action, executes either `execute_go`, `execute_take`, or `execute_drop`. The second word of the command is supplied as an argument to these functions. The first of these, `execute_go`, attempts to move the player in the specified direction (if the direction is a valid exit), the other two update the inventory and the list of items in a room accordingly. Complete these functions.

The game should be basically playable now. You have at this point achieved a decently smart Verb-Object *parser which "understands" sentences like "I would like to drop my laptop here". Moreover, you can now manipulate the game world (carry items around, drop and pick them up). Next, try making your game more interesting.*

9. Add the `"mass"` property to each item, and modify your game in such a way as to prevent the player from carrying more than three kilograms (or some other reasonable mass) at any given time.
10. Define some victory conditions for the game, and check (in the main game loop, after each move) whether the player has won. For example, the objective may be to find all items in the game and drop them off at the reception.

Appendix: Changes to the Game Template

I have made some changes to the game template since the previous version, to make it more logical. This is a summary of changes:

- The text processing functions like `normalise_input` have been moved into a separate module `parser.py`.
- The function `display_room` was renamed `print_room` for consistency.
- The function `print_menu_line` was renamed `print_exit` since the menu now offers not only a list of exits, but also a list of actions on items.
- The function `menu` no longer checks for correctness of the command (it is checked elsewhere).
- `current_room` is now a global variable in module `player.py`.