Quartermaster Developer Documentation

sudo-nano

December 30, 2023

Contents

1 Interface

The primary interface of the `quartermaster` program is the `prompt()` loop in `main.py`. It provides the user with a prompt, then takes their input and `match`es it to the list of valid commands. If it matches one, then it runs the appropriate function from `mechanics.py`.

2 Commands

2.1 calc

Usage: `calc <recipe> <quantity>`

The `calc` command takes two parameters, a recipe and the quantity of that recipe. For example,

3 Data Formatting

3.1 DataSet Class

The `DataSet` class is an object type for holding all the imported data in a `quartermaster` session. There is currently only one `DataSet` object, called `session`. All data imported from files is loaded into the session DataSet object.

3.2 TOML file formatting

Each type of data is stored in separate TOML files. Each TOML file must have `type = "<type_here>"` to indicate its type to the program. If it doesn't have this, the program will refuse to load it.

Currently implemented types:

- Ingredients
- Recipes

Types to implement later:

- People
- Groups
- Meal plans
- Sessions

3.3 Ingredient Files

Ingredient files must have `type = "ingredients"` to indicate that the file holds only ingredients, otherwise they will not be imported. This is meant to prevent accidentally importing other types into the session ingredient dataset.

Each ingredient is one table in its TOML file. Example below. See the included test_ingredients.toml file for more examples.

```
[rice]                  # Ingredient name in header
unit = "gram"           # Unit of measurement
price_per_unit = 0.01   # Price per unit of measurement

# List of lists, each sub-list describing quantity of purchase
# and cost of purchasing that quantity.
purchase_increments = [ [1000.0, 5.00] ]

# List of diet incompatibilities that conflict with this
# ingredient. Leave list empty if there are none.
diet_incompat = []
```

3.4 Recipe Files

4 Planned Features

- Meal Plan object
  - Allows easy planning for a limited subset of meal options
  - Ability to check meal plan compatibility with a set of people
  - Suggest necessary modifications for incompatibilities?
- Person object
  - Allows specification of dietary restrictions
  - Contains a float describing how many standard servings they will consume per meal
- Group object
  - Allows grouping of Person objects
  - Later, there will be a command to calculate the required supplies for a given group, meal plan, and length of time to supply the group.
- Session object
  - Allows saving and loading of session state