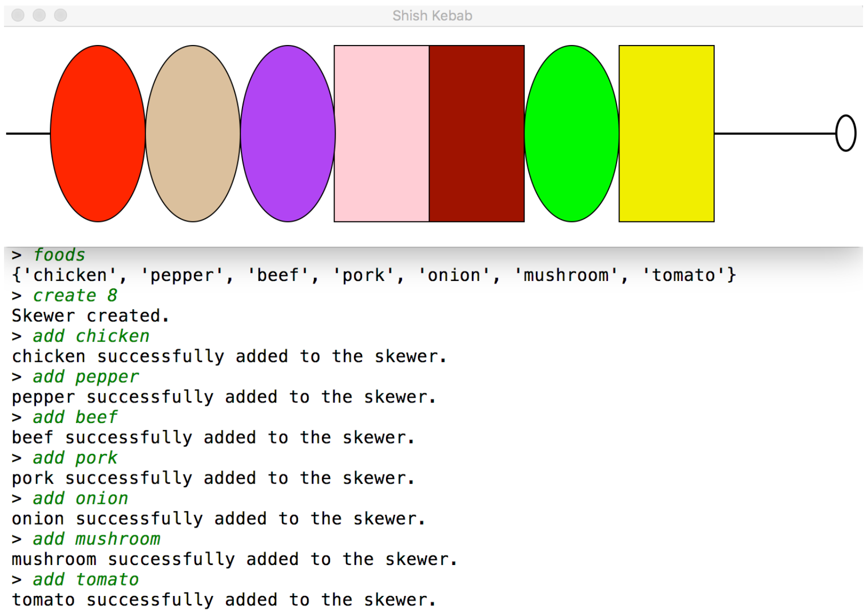
Computational Problem Solving CSCI-603
Shish Kebab Lab 5

10/16/2017



```
> foods
{'chicken', 'pepper', 'beef', 'pork', 'onion', 'mushroom', 'tomato'}
> create 8
Skewer created.
> add chicken
chicken successfully added to the skewer.
> add pepper
pepper successfully added to the skewer.
> add beef
beef successfully added to the skewer.
> add pork
pork successfully added to the skewer.
> add onion
onion successfully added to the skewer.
> add mushroom
mushroom successfully added to the skewer.
> add tomato
tomato successfully added to the skewer.
```

1 Implementation

1.1 Starter Code

Download the zip file with the starter code from MyCourses. Go to the Content section and navigate to this lab's folder. You should see a file, `code.zip`, that you can download and put into a new project.

Here is a short description of each source file:

- `kebab.py`: The main program. You will modify this program to implement the `calories` command.
- `skewer.py`: This is the definition of the class that represents the skewer. It contains a pointer, `top`, to the top item `KebabSpot` on the skewer. If there is no top item, the value should be `None`. You will have to modify this code to support the `calories` command.
- `kebab_spot.py`: This is the definition of the `LinkedNode` from lecture that combines to make the collection of food items on the skewer. Each `KebabSpot` has an `item`, a `Food` object, and a pointer to the next item, `next`. If there is no successor (e.g. the last/bottom item on the skewer), it will point to `None`. You will have to write all the functions here from scratch.
- `food.py`: This is the class that represents the food items. You will implement the `Food` class and also add a new item, a mushroom which is a vegetable that has 7 calories.
- `skewer_exception`: This is the definition of the exception class, `SkewerException`. It is used in scenarios where the skewer has not been created. You should not change this file.
- `kebab_graphics.py`: This is the class that handles the graphics for drawing the shish kebab. You will need to add the mushroom to the dictionary, `COLORS`. This dictionary

maps the food item name (a string, e.g. "mushroom"), to its color string (a name, e.g. "tan").

- `graphics.py`: This is a graphics library designed by John Zelle, a professor of Computer Science at Wartburg College. It is a wrapper built around the Tkinter GUI package. You should not modify this file in any way.

Please note that initially the program will not run correctly. This is because the underlying representation of the food items and the spots on the skewer do not exist.

1.2 Commands

These are the commands that the program responds to:

```
> help
Kebab commands:
add item - adds an item to the skewer
calories - get the total number of calories of items on the skewer
create N - creates a skewer to hold N items
destroy - destroys the current skewer
display - displays all the items on the skewer, in order
eat - eat the front item on the skewer
foods - display the food items that can be added to the skewer
front - the front item on the skewer
has item - is an item on the skewer?
quit - exit the program
status - the capacity and current number of items on the skewer
vegan - does the skewer have any meat?
>
```

To begin with, the `calories` command will not appear. You will implement it later. If an unrecognized command is entered, it displays the help message.

1.2.1 Create

The `create N` commands creates a new skewer of a certain maximum capacity. This needs to be done before any other operations that use the skewer are done, e.g. adding or removing items. If a skewer already exists, it is destroyed and the new one is created.

```
> create 5
Skewer created.
```

1.2.2 Foods

The `foods` command displays the valid strings of valid food items that can be added to the skewer. The valid foods and the ones that are vegetables can be found in the `food` module:

```
FOODS = {'beef', 'pork', 'chicken', 'onion', 'pepper', 'tomato'}
VEGGIES = {'onion', "pepper", 'tomato'}
```

Later on you will be adding a mushroom.

```
> foods
{'mushroom', 'pork', 'onion', 'beef', 'tomato', 'pepper', 'chicken'}
```

1.2.3 Add

The `add item` command adds a new food item to the front of the skewer, as long as the skewer is not at maximum capacity. The new item will display in the graphical window.

```
> add onion
onion successfully added to the skewer.
```

1.2.4 Front

The `front` command displays the item at the front of the skewer, as long as the skewer is not empty.

```
> add pork
pork successfully added to the skewer.
> front
pork is on the front of the skewer.
```

1.2.5 Display

The `display` command displays to standard output a list of the items on the skewer, from front to back. If the skewer is empty the list will be empty.

```
> add beef
beef successfully added to the skewer.
> display
The skewer contains: [ beef, pork, onion ]
```

1.2.6 Eat

The `eat` command removes the front food item from the skewer, as long as the skewer is not empty. The item will disappear from the graphical window when removed.

```
> eat
Ate beef . Yum!
```

1.2.7 Calories

The `calories` command displays the total number of calories of all items on the skewer.

```
> calories
The skewer has 130 calories.
```

The calories for each item are defined in a dictionary in the `food` module:

```
CALORIES = {
    'beef': 200,
    'chicken': 140,
    'pork': 100,
    'onion': 30,
    'pepper': 25,
    'tomato': 10,
    'mushroom': 7
}
```

1.2.8 Has

The `has item` command tells whether a certain item is on the skewer or not.

```
> has onion
onion does exist on the Skewer.
> has beef
beef doesn't exist on the Skewer.
```

1.2.9 Status

The `status` command shows how many items are currently on the skewer and what its capacity is.

```
2 out of 5 items on the skewer.
```

1.2.10 Vegan

The `vegan` command tells whether the skewer contains all vegetables or not.

```
> vegan
The skewer contains meat.
> eat
Ate pork . Yum!
> vegan
The skewer is vegan friendly.
```

The constant `VEGGIES` in the `food` module defines the valid vegetables.

1.3 Destroy

The `destroy` command destroys the current skewer. A new one needs to be created before any more items or queries can be done.

```
> destroy
> status
Skewer has not been created yet.
```

1.3.1 Quit

The `quit` command exits the simulation and closes the graphical window.

1.4 Requirements

For this assignment, you are required to implement a *node-based* stack. **You are not allowed to use the built in Python list!**

1.5 Tasks

These are the four tasks that you should follow in order:

1. Implement the `Food` class in `food.py`. Each food item has:
   (a) A name (string).
   (b) Whether it is a vegetable or not (boolean).
   (c) Number of calories (int).
   A food item should be constructable by passing in the name of the food item. Use the global constants in the `food` module to resolve the other two members.
2. Implement the `KebabSpot` class in `kebab_spot.py`. Like the `LinkedNode` from lecture, it contains two things:
   (a) The item (guaranteed to be a `Food` object).
   (b) The next item (reference to next `Food` object, or None if none remain).
   The methods have been stubbed out for you. Typically, the `Skewer` class in `skewer.py` calls on these routines with the front element on the skewer. You can choose to implement these routines iteratively or recursively. Once you finish this part, you should be able to work with all of the commands, except for `calories` and mushrooms.
3. Add the mushroom item. It is a vegetable with 7 calories (see `CALORIES` in `food.py`). The color of the mushroom is "tan" (see `COLORS` in `kebab_graphics.py`).
4. Implement the `calories` command. You will need to change the main program, `kebab.py`, as well as add a routine to both `skewer.py` and `kebab_spot.py`

Make sure when you are finished that you test out all the commands and scenarios.

2 Grading

This assignment will be graded using the following rubric:

- (20%) Problem Solving
- (70%) Functionality:
  - 5% `Food` class implementation.
  - 45% `KebabSpot` class implementation.
  - 5% Addition of mushroom as a valid item.
  - 15% Implementation of the `calories` command
- (10%) Code Style and Documentation: Proper commenting of modules, classes and methods (e.g. using docstring's).

3 Submission

The try command to submit is:

```
try grd-603 lab5-1 *.py
```

We are asking that you send all code to try - even the source files you aren't supposed to modify.

Recall that to verify your latest submission has been saved successfully, you can run try with the query option, -q:

```
try -q grd-603 lab5-1
```