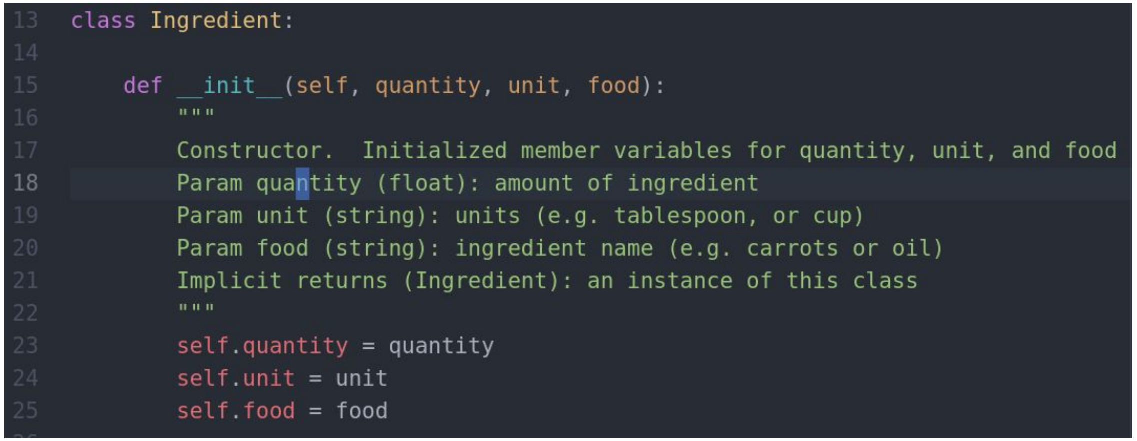
# Ingredient class

Class definition and constructor

Constructor should initialize all member variables



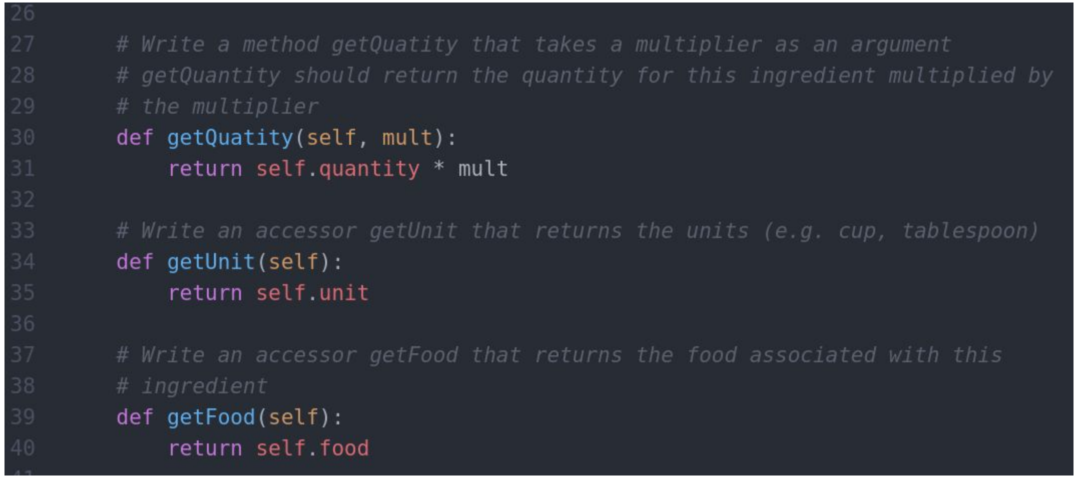
```
class Ingredient:

    def __init__(self, quantity, unit, food):
        """
        Constructor.  Initialized member variables for quantity, unit, and food
        Param quantity (float): amount of ingredient
        Param unit (string): units (e.g. tablespoon, or cup)
        Param food (string): ingredient name (e.g. carrots or oil)
        Implicit returns (Ingredient): an instance of this class
        """
        self.quantity = quantity
        self.unit = unit
        self.food = food
```

## Ingredient class - setters vs getters

setters - methods for setting member variables (aka mutators)

getters - methods for getting member variables (aka accessors)

Why use setters/getters instead of referencing member variables directly?



```

    # Write a method getQuatity that takes a multiplier as an argument
    # getQuantity should return the quantity for this ingredient multiplied by
    # the multiplier
    def getQuatity(self, mult):
        return self.quantity * mult

    # Write an accessor getUnit that returns the units (e.g. cup, tablespoon)
    def getUnit(self):
        return self.unit

    # Write an accessor getFood that returns the food associated with this
    # ingredient
    def getFood(self):
        return self.food
```

Ingredient defines getters for its member variables but not setters.

## Ingredient class - setters vs getters

Why use setters/getters instead of referencing member variables directly?

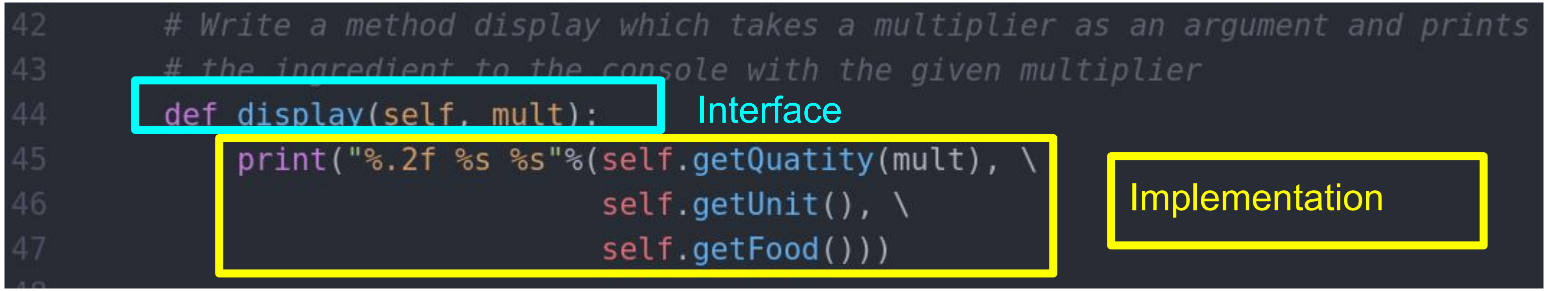
Setters/getters help *abstract* the details of the class away from the user.

-> e.g. we can change the class implementation and the user never need know!

# Interface vs implementation

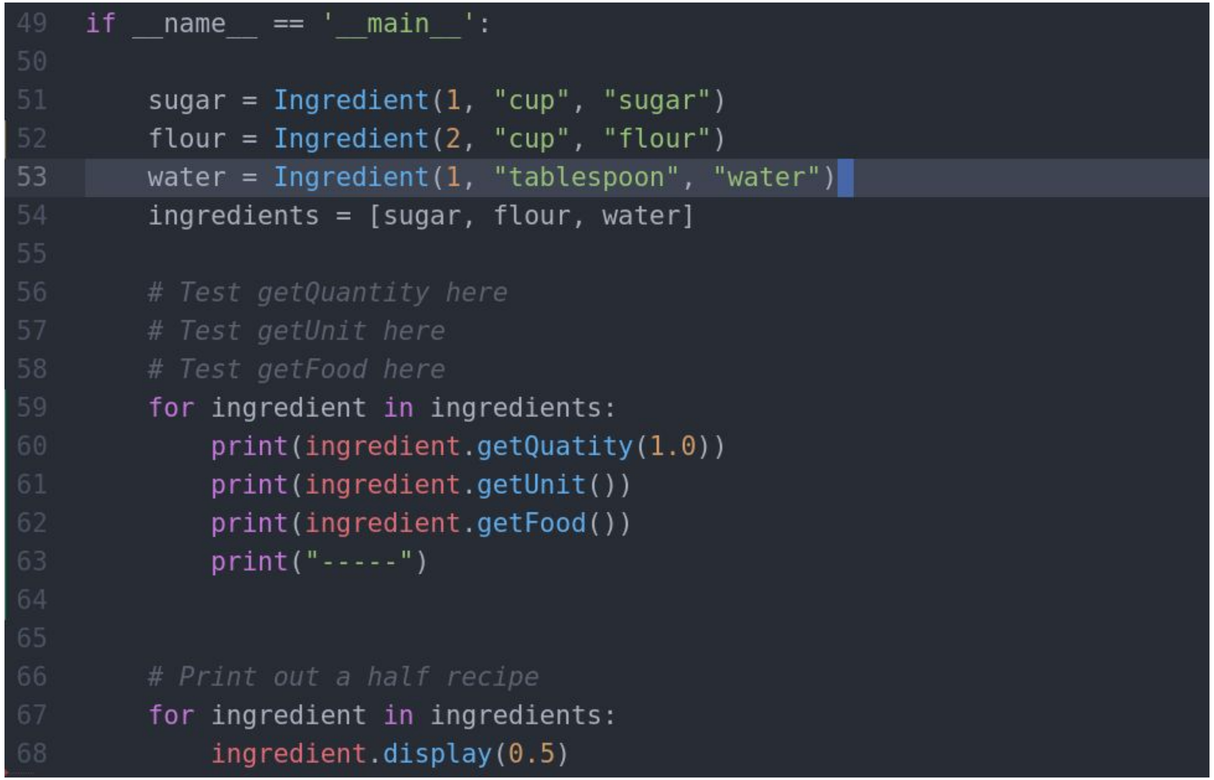
The methods of class define its **interface**. The interface defines how the user interacts with the object

The **implementation** is the body of the methods. In a good design, the user doesn't need to understand the implementation (classes should work like a **black box**)



```
    # Write a method display which takes a multiplier as an argument and prints
    # the ingredient to the console with the given multiplier
    def display(self, mult):
        print("%.2f %s %s"%(self.getQuatity(mult), \
                            self.getUnit(), \
                            self.getFood()))
```

## Ingredient class - testing

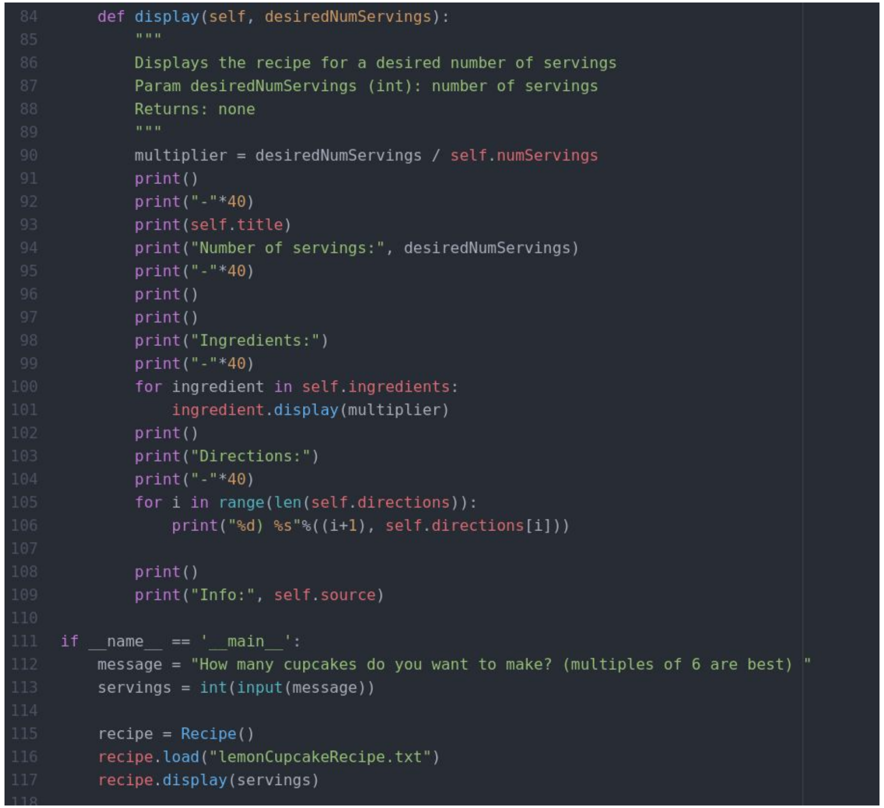


```python
if __name__ == '__main__':

    sugar = Ingredient(1, "cup", "sugar")
    flour = Ingredient(2, "cup", "flour")
    water = Ingredient(1, "tablespoon", "water")
    ingredients = [sugar, flour, water]

    # Test getQuantity here
    # Test getUnit here
    # Test getFood here
    for ingredient in ingredients:
        print(ingredient.getQuatity(1.0))
        print(ingredient.getUnit())
        print(ingredient.getFood())
        print("-----")


    # Print out a half recipe
    for ingredient in ingredients:
        ingredient.display(0.5)
```

# Using Ingredient from the class Recipe

Ingredients are created in the method Recipe.load. The method here prints the recipe for a desired number of servings.

Below, we test the Recipe class with lemonCupcakeRecipe.txt



```python
    def display(self, desiredNumServings):
        """
        Displays the recipe for a desired number of servings
        Param desiredNumServings (int): number of servings
        Returns: none
        """
        multiplier = desiredNumServings / self.numServings
        print()
        print("-"*40)
        print(self.title)
        print("Number of servings:", desiredNumServings)
        print("-"*40)
        print()
        print()
        print("Ingredients:")
        print("-"*40)
        for ingredient in self.ingredients:
            ingredient.display(multiplier)
        print()
        print("Directions:")
        print("-"*40)
        for i in range(len(self.directions)):
            print("%d) %s"%((i+1), self.directions[i]))

        print()
        print("Info:", self.source)

if __name__ == '__main__':
    message = "How many cupcakes do you want to make? (multiples of 6 are best) "
    servings = int(input(message))

    recipe = Recipe()
    recipe.load("lemonCupcakeRecipe.txt")
    recipe.display(servings)
```