Raid Documentation

**Method**:

* viewRecipeList(self)

**Description:**

* This method is responsible for displaying the list of recipes stored in RecipeManager.

**Parameters**:

* Self: the instance of the RecipeManager class.

**Functionality**:

1. Prints the table header with column names:
   * Uses the `format()` method with placeholders to specify the column widths and alignment.
   * The placeholders are filled with the column names: "ID", "RecipeName", "RecipeAuthor", "PrepTime", "CookTime", "ServingSize".
2. Prints a separator line:
   * Prints a line of dashes to separate the header from the recipe entries.
3. Iterates over each recipe in the `data` attribute of the RecipeManager instance
   * For each recipe in `self.data`, do the following:
4. Checks the length of the recipe name:
   * If the length of the `recipe.recipe\_name` is greater than or equal to 20 characters, it proceeds to the next step.
   * Otherwise, it skips to the `else` block.
5. Truncates the recipe name:
   * Assigns the truncated recipe name (excluding the last two characters) to the variable `rName`.
6. Prints the recipe information in a formatted table row:
   * Uses the `format()` method with placeholders to specify the column widths and alignment.
   * The placeholders are filled with the recipe ID, recipe name (`rName` if it was truncated, otherwise `recipe.recipe\_name`), recipe author, prep time, cook time, and serving size.
7. If the length of the recipe name was less than 20 characters, it executes the `else` block:
   * Prints the recipe information in a formatted table row without truncating the name.

**Usage**:

* To display the list of recipes, call the ‘viewRecipeList()’ method on an instance of the RecipeManager class. So, we will call this function on an instance of the RecipeManager class like this:
  + - recipe\_manager.viewRecipeList()

**title:**

* Class TestRecipeManagement()

**Description:**

* This code snippet defines a test case class named TestRecipeManagemen, which contains a unit test method to test the viewRecipeList() method of the RecipeManager class. The test verifies that method generates the expected output when display the recipe list.
* Important modules:
  + - Sys: that provide access to system-specific parameters and functions.
    - Io.StringIO: Implements an in-memory stream for capturing console output.
    - Unittest: Provides the framework for creating and running unit tests.

**Class**:

* TestRecipeManagement(unittest,TesstCase)

**Methods**:

* Test\_view\_recipe(self)

**Description**:

* This method tests the viewRecipeList() method of the RecipeManager class.

**Steps:**

1. Create an instance of the RecipeManager class.
2. Create a StringIO object to capture console output.
3. Redirect the standard output to StringIO object.
4. Invoke the viewRecipeList() method to display the recipe list.
5. Restore the standard output to its original state.
6. Retrieve the captured output from the StringIO object.
7. Compare the captured output to the expected output.

**Expected Behavior:**

* The captured output should match the expected output, which represents the formatted recipe list.

**Title:**

* TestRecipeManagerIntegration Class

**Description:**

* This code snippet defines a test case class named TestRecipeManagerIntegration, which contains a unit test method to test the functionality of adding and viewing recipes in the RecipeManager class. The test verifies that the addRecipe() and viewRecipeList() methods work as expected by checking the behavior and properties of the recipe list.

**Imported Modules:**

1. unittest: Provides the framework for creating and running unit tests.
2. Core.recipeProject: Includes the RecipeManager and Recipe classes required for testing.

**Class:**

* TestRecipeManagerIntegration(unittest.TestCase)

**Methods:**

* test\_addRecipe\_and\_viewRecipe(self)

**Description**:

* This method tests the addRecipe() and viewRecipeList() methods of the RecipeManager class.

**Steps:**

1. Create an instance of the RecipeManager class.
2. Create a new\_recipe object of the Recipe class with specific details.
3. Invoke the addRecipe() method of the RecipeManager instance with the new\_recipe.
4. Get the recipe\_list by invoking the viewRecipeList() method of the RecipeManager instance.
5. Perform the following assertions to verify the behavior and properties of the recipe list:
6. Check that the new\_recipe is not present in the recipe\_list.
7. Check that the length of the recipe\_list is equal to 1.
8. Check that the properties of the first recipe in the recipe\_list match the expected values.
9. Check that the ingredients of the first recipe in the recipe\_list match the expected values.
10. Check that the instructions of the first recipe in the recipe\_list match the expected values.
11. Check that the new\_recipe is present in the RecipeManager instance's data attribute.

**Expected Behavior:**

* The assertions should pass, indicating that the addRecipe() and viewRecipeList() methods function correctly.