

Final Project

There are 20 recipe files used as the starting data for the project. Each recipe is a .txt file.

Each begins with Recipe 1 followed by a new line character followed by the recipe name(title), new line character, and finally the ingredients used and their quantity. Each ingredient begins with the quantity followed by the ingredient name. Each ingredient details are on a separate line. Below is the sample recipe –

Recipe 1

Recipe name

Ingredient quantity and name

Ingredient quantity and name

Ingredient quantity and name

Ingredient quantity and name

Below are the steps and the flow of the program –

Reading the required .txt recipe files:

For this, I have used the existing Python library 'pathlib'. I was not sure how to read multiple files from a folder. Professor helped me with this portion and explained how to proceed with this. The variable 'data' has the content of all the recipes that are read from the given folder.

Storing the recipes in a proper data structure:

Now, 'data' has all the recipes together. Each recipe starts with the word 'Recipe', so I splitted accordingly. Now, the variable 'datanew' of list type has each recipe as a separate data element. I removed the empty data elements from this list. So, the length of 'datanew' should match the number of recipes. There is a leading white space at the beginning of each data element of the list. This is removed using the .lstrip() method.

Sorting 'datanew' such that the list is sorted:

When I printed 'datanew', I realized that the recipe files are not read sequentially but in any random order. So, I created the list 'mylist' that will have the recipes in proper order.

Logic for sorting the list:

Each recipe should ideally begin with the word Recipe followed by the Recipe number. The data can be sorted based on this number. At present, the split is done on the word Recipe, so each recipe begins with the recipe number. Thus, it is the 0th element. I used a for loop for going

through the entire 'datanew' recipes list. A list has index positions 0,1, 2 and so on. Recipes start from 1,2,3 and so on. Finally, mylist is the final list which has all the recipes in sequential order. The word 'Recipe' is added at the beginning of each recipe. mylist is printed to check if the results obtained are as required.

Finding the names of each recipes:

Each recipe begins with Recipe followed by the Recipe number and the title on the next line. For instance -

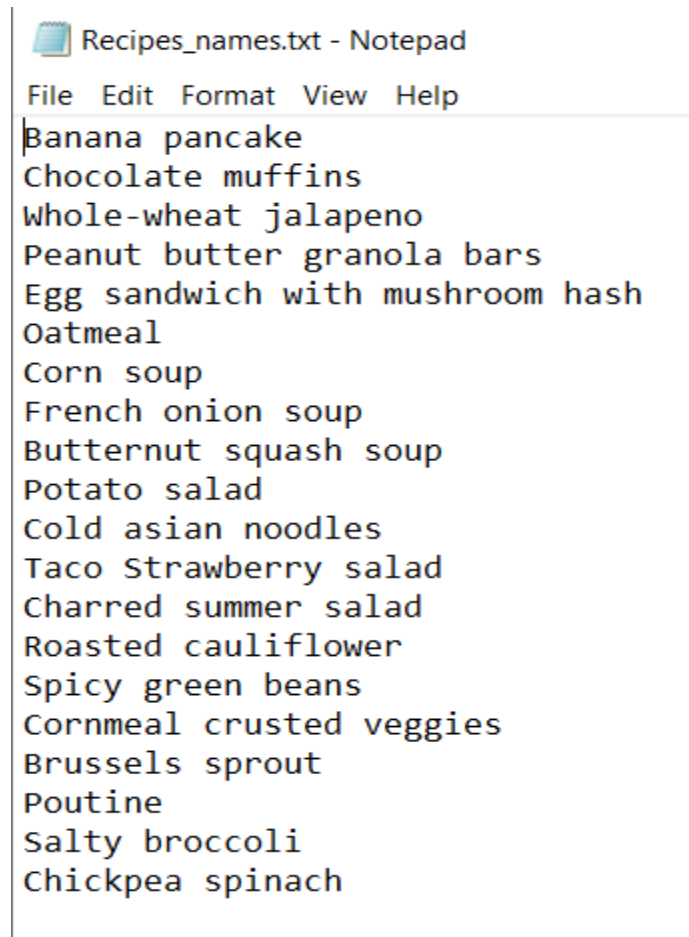
Recipe 1

ABC

If we do a split on this, my 3rd element will be my title, which is nothing but list[2]. The variable 'titles' stores the names of the recipes in proper sequential order starting with Recipe 1, 2, 3 and so on.

The recipe names are printed to a .txt file 'Recipe_names' which has all the names of the recipes sequentially.

Below is the screenshot of the output –



```
File Edit Format View Help
Banana pancake
Chocolate muffins
Whole-wheat jalapeno
Peanut butter granola bars
Egg sandwich with mushroom hash
Oatmeal
Corn soup
French onion soup
Butternut squash soup
Potato salad
Cold asian noodles
Taco Strawberry salad
Charred summer salad
Roasted cauliflower
Spicy green beans
Cornmeal crusted veggies
Brussels sprout
Poutine
Salty broccoli
Chickpea spinach
```

Finding the number of ingredients in each recipe:

The recipes are such that each ingredient is on a new line. Thus, we need to find the number of lines in each recipe and subtract it by 2 as the first line is Recipe number and the second is the title.

Note – As discussed with professor, I was told to use proper variable names in at least one function block. I have used proper variable names in this function.

Producing the output file with the Recipe name and number of ingredients:

The csv 'RecipeAndIngredients' stores the required result. It has 2 columns, Recipe name and the Number of Ingredients. This code is similar to the one used in the midterm.

Note – As discussed with professor, I was told to produce at least one sample output file. I have produced the csv file for this output.

Below is the screenshot of the output –

1	Recipe name	Number of ingredients
2	Banana pancake	10
3	Chocolate muffins	11
4	Whole-wheat jalapeno	7
5	Peanut butter granola bars	6
6	Egg sandwich with mushroom hash	8
7	Oatmeal	3
8	Corn soup	10
9	French onion soup	10
10	Butternut squash soup	12
11	Potato salad	5
12	Cold asian noodles	5
13	Taco Strawberry salad	7
14	Charred summer salad	6
15	Roasted cauliflower	6
16	Spicy green beans	6
17	Cornmeal crusted veggies	9
18	Brussels sprout	7
19	Poutine	5
20	Salty broccoli	6
21	Chickpea spinach	6

Finding the recipe with least ingredients:

The list 'lines' has the number of ingredients. The minimum number is the recipe with least ingredients. We fetch this using the min() function.

Below is the screenshot –

```
The recipe with least ingredients is : Recipe 6 - Oatmeal , which has 3 ingredients
```

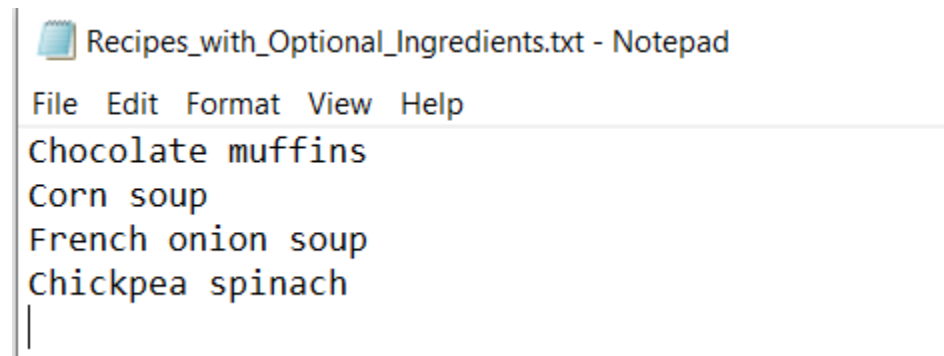
Finding the recipes with optional ingredients:

The list 'blanks' stores the positions of the recipes which have optional ingredients. The list 'mylist' has all the recipes. In a for loop, we go through each such recipe, we use split() method to search for the word 'optional'. If the word is found, that position is appended to the 'blanks' list. A recipe can have one or more optional ingredients. Even if it has 1 optional ingredient, it needs to be included. So, even if 'optional' word is found once, we need not go through the further ingredients of that recipe. That is why I have used 'break' here to terminate the search when 'optional' word is found even once in the given recipe.

The variable 'optrecipe' stores the titles of such recipes which have optional ingredients. A for loop is used to fetch the title names from the positions found.

The results that is the names of the recipes which have 'optional' ingredients are printed to a 'Recipes_with_Optional_Ingredients.txt' file that has the recipe names as required.

Below is the screenshot of the output –



Finding the recipes which have optional ingredients and finding those ingredient details as well:

'blanks' has the positions of such recipes. Adding 1 to it fetches the required recipe number. If we split on a new line character, we get each line of a recipe. We loop through each such line, within each line, we again split and loop through each word. If the word 'optional' is found, the entire line is printed. Thus, we print the recipe number, the recipe name, and the optional ingredient details as required.

Below is the screenshot of the output –

Optional ingredient(s) in Recipe 2 - Chocolate muffins are :

1 tsp cinnamon optional

1/2 cup dark chocolate chips optional

Optional ingredient(s) in Recipe 7 - Corn soup are :

1 chili pepper, finely chopped optional

Optional ingredient(s) in Recipe 8 - French onion soup are :

1 tsp vinegar, any type optional

Optional ingredient(s) in Recipe 20 - Chickpea spinach are :

1 smoked paprika optional

Finding the recipes which have fruits:

The list 'fruits' has the names of the fruits. The list 'frt' stores the positions of the recipes which have fruits. In a for loop, we go through each word of a recipe. If the word is found in the 'fruits' list, append the position to the 'frt' list.

The list 'fruitrecipe' should have the recipe names of such recipes which have fruits. Using 'frt' which has the positions, the title is fetched.

Below is the screenshot of the output –

The recipes which have fruits are -

['Banana pancake', 'Taco Strawberry salad ']

Finding the recipes which use potato:

The list 'potato' is used to store the position of the recipes which use potato. In each recipe, we loop through the words. If the word 'potato' is found, the position is appended to 'potato'. The titles of these recipes are fetched using these positions. The list 'potatorecipe' has the recipe names.

Below is the screenshot of the output –

```
The recipes that use potato are -  
['Egg sandwich with mushroom hash', 'Corn soup', 'Potato salad', 'Poutine']
```

Finding the total quantity of salt used in all the recipes:

The list 'salt' is used to store the lines in the recipes which have salt. The loop is written such that, in each recipe, we look for each line. In each line, we look for each word. If the word 'salt' is found, the entire line is appended to the 'salt' list.

The variable 'totalsalt' is used to store the total salt quantity in all the recipes. Each line in 'salt' begins with the quantity. So, we fetch the 0th element. The fractional quantity is converted to decimal and float. We add up these quantities and finally print the total quantity of salt used in all the recipes.

Below is the screenshot of the output –

```
The total quantity of salt used in all the recipes is 13.75 table spoons
```

Approach –

I used some of the recipes from an online cookbook -

<https://cookbooks.leannebrown.com/good-and-cheap.pdf>.

When I started working on this project, I was almost clueless and very much worried about writing such a big program for the first time in my life. I decided that I will divide it into smaller steps. The proposal and the checkin definitely helped me as I started to think about the project long before I actually started to work on it. While working, I was stuck at the initial step itself wherein I was supposed to read multiple files from a folder. Professor helped me in this portion. After this, I started working with my data and thought about how I am going to manipulate and use it for my project. When writing each block of code, I used to read the code in my mind (as in, ok so now I read the line, now I have to read each word, I have to find one word among this and so on). That is why, I have used short variable names in my code as it was easier for me to process in my mind while I was working on it. I discussed this with Professor, wherein Professor told me that I can go ahead with this and use proper variable names in at least one block of code. As mentioned before, I have use proper variable names for finding the number of ingredients in each recipe. While writing the loops, I printed each step to see if I was going in the correct direction. I also printed all intermediate steps to ensure my output results. As discussed with Professor, I have printed the results in the Python program itself using print() statement, and for some I printed in csv or txt files.

Challenges faced –

I faced a lot of challenges while working on this project. The first one was to read the multiple files. Then at each step, I was lost so many times while working on the data. I divided into smaller parts and printed the results while working. It took me a lot of time to understand and tackle the questions that I had decided to work on in this project. I also had never used GitHub before this. I really enjoyed a lot working on this project and I feel confident that I was able to write most portion of the code by myself.

GitHub –

Below is the link for the project's GitHub page –

https://github.com/tejalmahore/tmahore2_FinalProject

Deliverables –

Below is the list of deliverables that I have attached with the submission –

- The .py Python code file with significant comments - tmahore2_FinalProject_code.py
- The 20 .txt recipe files (I have attached a zip folder) - recipes.zip
- A readme file that will describe how to run the code and give an overview of how it works - tmahore2_FinalProject_readme file.pdf
- Narrative files – tmahore2_FinalProject_Narrative.pdf
- A manifest file describing the various files used and the folder structure within the project – tmahore2_FinalProject_manifest file.pdf
- The output files produced – Recipes_names.txt, RecipeAndIngredients.csv, Recipes_with_Optional_Ingredients.txt
- The project's GitHub page - https://github.com/tejalmahore/tmahore2_FinalProject