**ENGR 102 – Programming Practice**
**Mini Project 1**
**Fall 2018**

*Tags: Tkinter, GUI Widgets, Layout, Class, Object, Method, Files, Exceptions*

All types of foods and drinks have certain amount of calories that affect our diets. For those people who want to keep a healthy lifestyle and so watch out what food they consume, they may have some difficulties in counting the calories in each dish or drink they are having. To serve for such a purpose, you are going to develop a software tool which keeps track of calories of the foods and drinks chosen by the user based on a specified calorie amount. Please read on more details.

**How should it look like?**

The graphical user interface of your tool will consist of an expendable window, which will expand two times, once after each operation (Pressing **Continue** button).

Initially, your GUI will look like as shown in Figure 1.

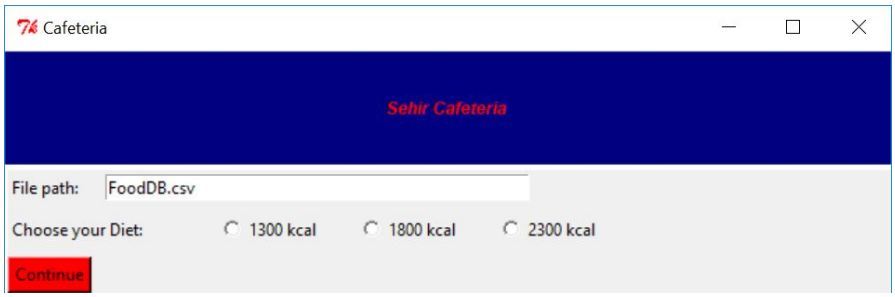


**Figure 1**. Main GUI window-default view

Once the user chooses a diet and presses **Continue** button, the GUI window will be expanded as shown in Figure 2.

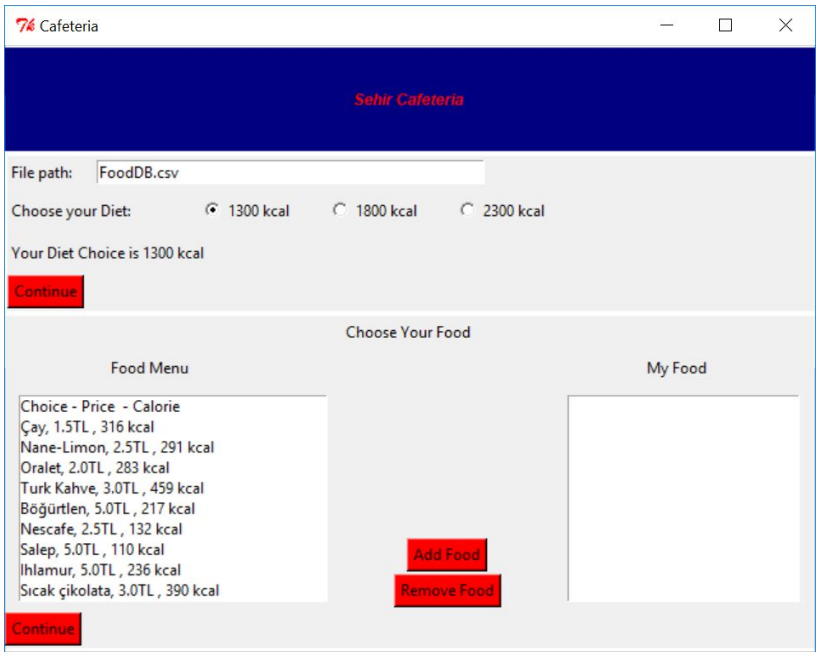


**Figure 2.** GUI expanded with a new frame after the user chooses the diet and presses **Continue** button.

After the user chooses their food from the Food Menu and add to "MyFood", once they are done with the selection and press **Continue** button; GUI window will expand again as shown in Figure 3.

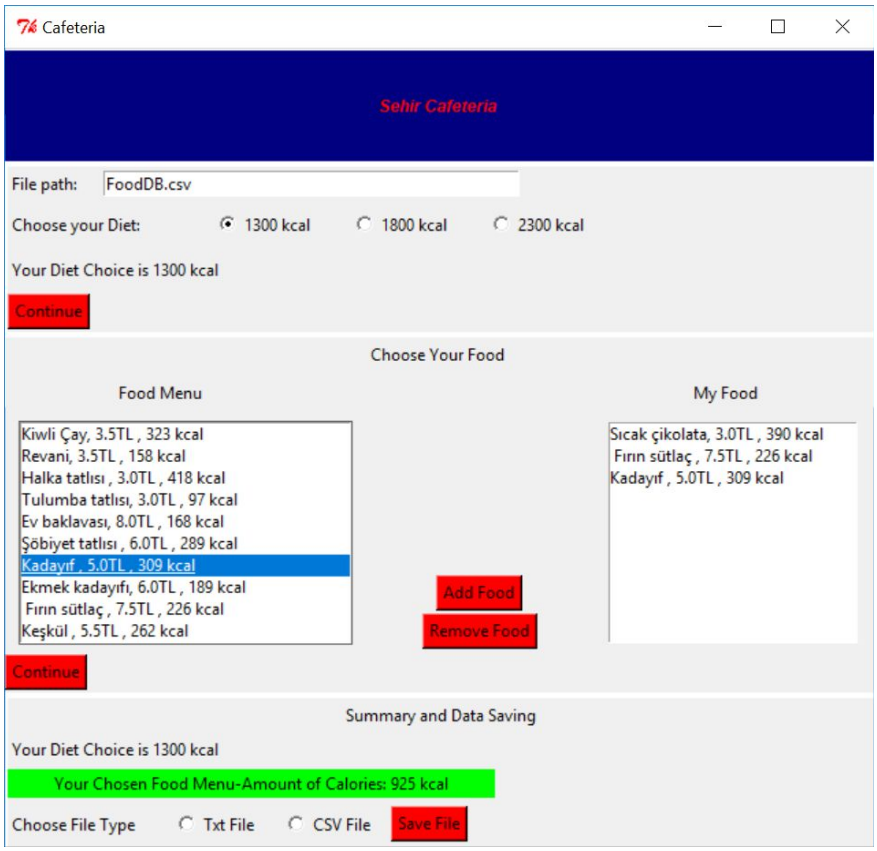

**Figure 3.** GUI window expanded with a new frame after the user creates their own menu ("My Food") from the Food Menu and presses **Continue** button.

**How should it work?**

1. First, the user should load the list of foods from the food menu file (FoodDB.csv). The program will read the file path from the "File path" entry which is at the top of the window. If the food menu file is in your current directory where your Python file is, it is sufficient to specify the file name as "File path" entry (Figure 4). The program should take a long path if the file is not in the same directory with the python file (Figure 5). A sample excel file for food menu (FoodDB.csv) is provided as part of the project documents to test your code.


**Figure 4**. File path entry if the csv file is in the same directory with the Python file.


**Figure 5**. File path entry if the csv file is NOT in the same directory with the Python file.

2. At the end of first step, the data of FoodDB file should have been loaded. Now the user can choose their diet in terms of specified amount of calories (Figure 1). Calories selections are provided by three radio buttons (Figure 1). The user can choose from one of three radio buttons which represent the diet type in terms of calories. After the diet type is selected, the user can press **Continue** button. Once the **Continue** button is pressed, the GUI window will expand as shown in Figure 2. The loaded data from FoodDB should now be visible in a listbox (Figure 2).

3. In this step, the user should be able to choose the foods and drinks that he would like to have from the **Food Menu** listbox and by pressing the **Add Food** button. Once the **Add Food** button is pressed, the food or drink selected should be added to the list box **My Food** on the right hand side of the frame.

   **Note:** the user should be able to choose as much as he/she wants from the Food Menu even if it's the same type of food or drink. Every selection should be appended to the previous selection, if not removed, and be visible in **My Food** listbox. The first row in the **Food Menu** is not addable since it's there to categorize the Food Menu.

4. As much as the user is able to **add food**, the user should also be able to **remove food** too. The **Remove Food** button should therefore delete anything the user selected from **My Food** listbox.

5. After the user is happy with their food/drink selection as specified in **My Food**, they can press **Continue**. Now the GUI window will expand again as shown in Figure 3 to show the **Summary and Data saving** portion of the window. In this part of the window, they should be able to see their selected diet type via **Your Diet Chose is ..** together with **Chosen Food Calorie count**. The idea here is that the user can see whether or not the menu he/she created is compatible with his/her selected diet in terms of amount of calories. If the calorie amount of My Food listbox is above the amount of calories specified by the chosen diet, the

background of **Chosen Food Calorie count** label should be red with a warning message **"xx calories above the daily limit"** (Figure 6). If the calories for the My Food menu is below the chosen diet calorie at first then the label's background should be green (Figure 7).

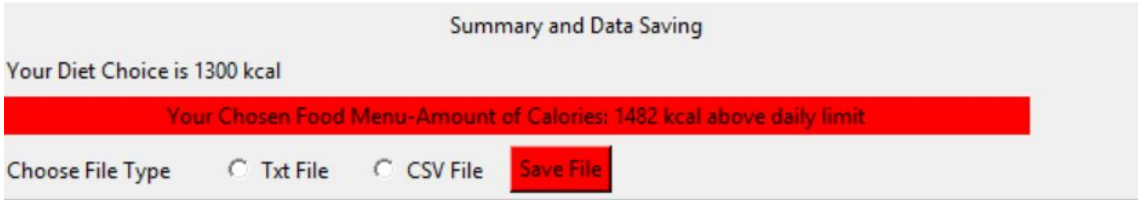


**Figure 6**. Format and content of the **Chosen Food Calories Count** label when the sum of selected food calories is above the chosen diet calorie.

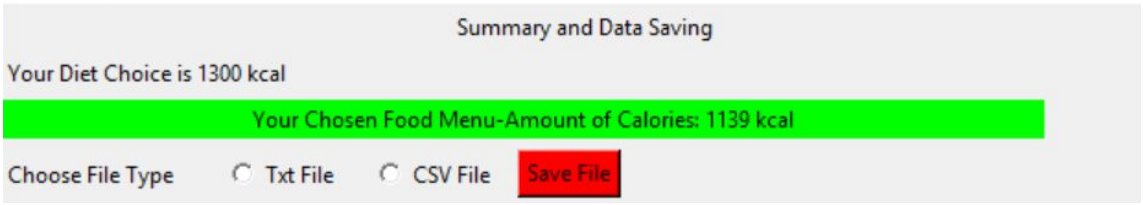


**Figure7**. Format and content of the **Chosen Food Calories Count** label when the sum of selected food calories is below the chosen diet calorie.

6. Finally, the user should be able to save his diet list **(My Food)** into a txt file or a csv by choosing between two radio buttons.

**Implementation Notes:**

- **You need to use at least 2 classes in this project. For instance, you may create one class that creates the file (txt and csv), another one for GUI. Please design your code accordingly.**
- **In order for you to understand better how the GUI should work, please check the GIF image that will be attached with the project documents.**
- For ListBox, please see the following reference:

  http://effbot.org/tkinterbook/listbox.htm

- For radio buttons, please check the following reference:

  https://www.tutorialspoint.com/python/tk_radiobutton.htm

- Apart from the above, the lecture slides are enough for most parts, still feel free to research and implement. Please provide references if you use outer sources. TutorialsPoint, Stack Overflow and Effbot might be good starting points, but **do not copy codes directly**.

**Warnings:**

- You **CANNOT** use place for geometry, only grid and pack are allowed.
- Do not talk to your classmates on project topics when you are implementing your projects. Do not show or email your code to others. If you need help, talk to your TAs or myself, not to your classmates. If somebody asks you for help, explain them the lecture slides, but do not explain any project related topic or solution. **Any similarity in your source codes will have serious consequences for both parties**.
- Carefully read the project document, and pay special attention to sentences that involve "should", "should not", "do not", and other underlined/bold font statements.
- If you use code from a resource (web site, book, etc.), make sure that you reference those resource at the top of your source code file in the form of comments. You should give details of which part of your code is from what resource. Failing to do so may result in **plagiarism** investigation. Last but not the least, you need to understand code pieces that you may get some other resources. This is one of the goals of the mini projects.
- Even if you work as a group of two students, each member of the team should know every line of the code well. Hence, it is important to understand all the details in your submitted code.

**How and when do I submit my project?**

- Projects may be done individually or as a small group of two students (doing it individually is **strongly** recommended for best learning experience). If you are doing it as a group, only **one** of the members should submit the project. File name will tell us group members (Please see the next item for file naming details).
- Submit your own code in a single Python file. Name it with your and your partner's first and last names. As an example, if your team members are Deniz Barış and Ahmet Çalışkan, then name your code file as deniz_baris_ahmet_caliskan.py (Do not use any Turkish characters in file name). If you are doing the project alone, then name it with your name and last name similar to the above naming scheme.
    - Those who do not follow the above naming conventions will **get** 10% **off** of their project grade.
- Submit it online on LMS by **17:00 on 6th November, 2018, Tuesday**.

**Late Submission Policy:**

- -10%: Submissions between 17:01 – 18:00 on the due date
- -20%: Submissions between 18:01 – midnight (00:00) on the due date
- -30%: Submissions after which are up-to 24 hours late.
- -50%: Submissions which are up-to 48 hours late.
- Submission more than 48 hours late will not be accepted.

**Grading Criteria?**

| GUI Design (25) | File Path and Diet Choice/ Calories Tracking (20) | Displaying Data from FoodDB in listbox and Add / Remove Meals From Listboxes (25) | Appropriate use of Error Handling And Exceptions (10) | Writing txt and CSV files (20) |
|---|---|---|---|---|

**Your code should be efficient, easy to follow and track.** Therefore, from your overall grade, we will deduct points by the specified percentage for the following items:

- Inappropriate/Cryptic variable names and method names (10%)
- Classes and objects are not used properly (30%)
- Insufficient commenting (10%).

**Have further questions?:**

If you need help with anything, please use the office hours of your TAs and the instructor to get help. **Do not walk in randomly (especially on the last day) into your TAs' or the instructor's offices. Make an appointment first. This is important. Your TAs have other responsibilities. Please respect their personal schedules.**

**IMPORTANT NOTES :**

**Note 1: Plagiarism**:

- Zero tolerance
- Cases will be referred to the Ethics Committee
- Both parties (provider and receiver) are responsible
- Process:
  - Automated computerized checks for pre-filtering
  - Human review for confirmation
  - Referral to the Ethics Committee if true positive

- **Note 2:** The calories of foods and drinks in FoodDB may not be their actual calories which is not an important issue at all as they are valid items for your program to execute.