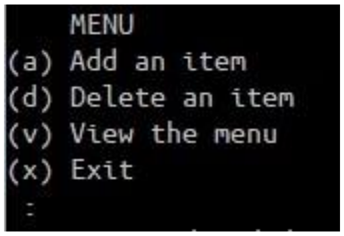
Python Course

Exercise 1– Restaurant Menu Manager

Description: We will allow the restaurant owner to manage the menu for his restaurant. He can view the menu, add an item, update an item and delete an item. The menu will be saved to a local file – it will be loaded from the file when the program starts, and written to the file before exiting. This way, when he comes back later his changes will still be there.

The program will be broken into two files – one which will deal with the UI (user interface), eg. showing the user menu, getting user input, etc. The other file will handle all the actual adding/removing of items from the menu, and the loading and saving of the data to a JSON file. This separation is important – the UI part should not 'know' anything about the menu file itself

1. Download the file menu as the simple menu to use for the restaurant. Make any changes you want.
2. Create a file called **menu_manager.py**. It should contain a class called **MenuManager**, with the following functions:
    1. **__init__()** - The function should attempt to read the menu from a specific **file path** (hardcoded inside the class), and store it in a variable (named 'menu') that will belong to the class object (what keyword do we use to refer to the class object?).
    2. **add_item(name, price, spice, gluten)** – this should add the given item to the menu, but not save it to the file yet.
    3. **remove_item(name)** – if the item named 'name' is in the menu, this should remove it from the menu (but not save to file yet), and return **True**. If the item was not in the menu, return **False**.
    4. **save_to_file()** - save the menu to the file. It was loaded from __init__.
3. Create a file called **menu_editor.py**, which will have the following functions:
    1. **load_manager**() - this will create a new object (instance) of the class MenuManager and return it.
    2. **show_user_menu**() - this will show the **program menu** (not the restaurant menu!), and ask the user to choose an item. Call the appropriate function that matches the user's input. Here's an example:



```
    MENU
(a) Add an item
(d) Delete an item
(v) View the menu
(x) Exit
 :
```

3. **add_item_to_menu**() - this will ask the user to input the item's name, price, spice level and gluten index. It will not interact with the menu itself, but simply call the appropriate function of the **MenuManager object.** After the item was added successfully, print a message to the user telling them that the item was added successfully and save it to the json file.
4. **remove_item_from_menu**() - this will ask the user to type the name of the item he wants to remove from the restaurant's menu. This function will not interact with the menu itself, but simply call the appropriate function of the **MenuManager object.**
   1. If the item was deleted successfully – print a message to let the user know this was completed. And update the file accordingly.
   2. If not – print an error message to the user.
5. **show_restaurant_menu**() - show the **restaurant's menu**, formatted in a user-friendly format.

4. When the user chooses to exit the program, first write the menu to the file, then show a message to tell the user that it was saved, then exit.
5. When you run the program after making changes to the restaurant menu, you should see your changes reflected in the menu (and also in the JSON file).

<u>BONUS: More functions and decorators</u>

1. Add a functionality to update an item in the menu: **update_item_from_menu()**
2. In the file **menu_editor.py**, use decorators to automatically save the new information to the json file, when the functions **add_item_to_menu**(), **update_item_from_menu()**, and **remove_item_from_menu**() are called.

<u>BONUS: Regular Expressions</u>

1. The Manager wants to do a special Valentine's night, but there are some rules. Create a new list of items special Valentine day inside the json file, and apply these rules with regular expressions. If these rules are not applied don't add or update the menu.
   a) Each part of the name should start with an uppercase letter. The first word needs to start with a **"V"** in uppercase. If the name contains connection words, they should start in lowercase.
   <u>Example:</u> **V**egetable **S**oup **o**f **t**he **D**ay
   b) It needs to contain two **"o"**, but no numbers
   c) The price needs to match this pattern: **XX,14**, where x are numbers. The two first numbers shouldn't be **1** or **4**.
2. Create an algorithm that displays a heart with ***** , when the menu is showed.