Developers Institute

Python Course

Week 3

Day 1

Exercises

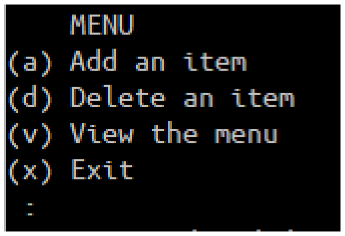
Exercise 1 – Random Sentence Generator
Description: We will create a program that will generate a random sentence and display it to the user. We will allow the user to choose how long the sentence will be.

1. Download this word list: http://norvig.com/ngrams/sowpods.txt
2. Save it in your development directory for this week.
3. Create a function called **get_words_from_file** that will read the file's contents and return them as a collection. What data type should you use?
4. Create another function called **get_random_sentence**. It should have a single parameter, '**length**', which will be used to determine how many words the sentence should have.
5. The function should get the words list, and choose random words from it, according to the amount specified by 'length'.
6. The words should be joined together into a string, which should be formatted in lower case (or title case) and returned.
7. Create a **main** function. It should:
   1. Print a message explaining what the program does.
   2. Ask the user how long the sentence should be. Acceptable values: an integer between 2 and 20. **Validate** your data, and test your validation!
   3. If the user inputs **incorrect** data, print an error message and end the program.
   4. If the user inputs **correct** data, user your functions to create a random sentence, and then display it proudly to the user.

Exercise 2 – 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, 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 (**encapsulation**)*

1. Download this file 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** (hard-coded 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)** – 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**. (Hint: use Python's **del** operator)
   4. **save_to_file()** - try to save the menu to the file it was loaded from in __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.
   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 and price. It will not interact with the menu itself, but simply call the appropriate function of the **MenuManager object.** If the item was added successfully to the MenuManager object (how do we know?), print a message to the user telling them that the item was added successfully.
   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.
      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).