David Zeng

August 9, 2023

IT FDN 110A

Assignment 5

Creating A Python Script for Managing Tasks as a To-Do List

Introduction

We write a Python script that acts as a “To-Do” list taker. We have a text file that stores information about tasks and their priority level and our script interacts with the user to update this “to-do” list. This script uses our previously learned knowledge about the Python programming language and introduces a new concept known as dictionaries. A Python dictionary is a collection that can be thought of as a set of “associations.” Like how a list contains elements, a dictionary also contains elements, but every “element” of a dictionary is an association (or “mapping”) of a “key” to a “value.” It may help to think about how a modern day dictionary contains a ”word” and its “definition.” Dictionaries are typically used by “looking up” a “value” associated with a given “key.”

Programming Our Script

Our script will edit a text file called “ToDoList.txt” that exists in the same directory as the script. Our script assumes that this file already exists, so make sure to create a text file named “ToDoList.txt” if it does not already exist in the same directory as the script.

Our script was developed after given starter code, so our tasks will be to:

- Load the data in the text file into the local collection (a list that acts as a “table”)
- Display the current context of the to-do list to output
- Code functionality that allows the user to add an entry to the to-do list
- Code functionality that allows the user to remove an entry from the to-do list
- Code functionality that that saves the tasks in the current script session to the text file

Loading the data

The general idea for loading the data from the text file into the local environment is as follows:

1. Create a file object for interacting with the text file
2. Iterate through each line of data in the text file:
    a. Create a dictionary collection storing the data form the text file
    b. Append the dictionary collection to the list collection (that represents the “table”)
3. Close the file object

Here is an example of what this code may look like:

```
fileObj = open(fileName, "r")
for line in fileObj:
    lineData = line.split(",")
    dicRow = {"Task":lineData[0].strip(), "Priority": lineData[1].strip()}
    lstTable.append(dicRow)
fileObj.close()
```

The most important part of this code is to note that we are using a dictionary collection, so it is important to understand and learn the syntax here. The "dicRow" variable is referring to a dictionary, and here we are creating a dictionary that maps the key "Task" to the value that is the first token in the line of text in the text file and that maps the key "Priority" to the value that is the second token in the line of text in the text file.

Displaying the data to output

This part of the code should be straightforward as it doesn't make use of any new knowledge. We simply output a "table" by printing a header and then iterate through every dictionary element in the list collection, printing out the contents in a formatted way.

```
print("Task|Priority")
for entry in lstTable:
    print(entry["Task"] + "|" + entry["Priority"])
```

Adding an entry

This section of the code will be similar to the section of code "Loading the data" because our goal is to update our local list collection with an entry representing the user's new task and priority. We will:

1. Get input from the user about the task and priority to be added
2. Update our local list collection with a dictionary collection storing this user data

```
taskInput = input("Enter the task: ").strip()
priorityInput = input("Enter the priority: ").strip()
dicRow = {"Task":taskInput, "Priority":priorityInput}
lstTable.append(dicRow)
print("New entry added")
```

Removing an entry

This section of the code will involve searching our list collection representing our to-do list for a given task and removing that task. The general idea here is to:

1. Get input from the user about the task to be removed
2. Iterate through our list collection
    a. If we find a dictionary entry that represents the target task, remove it

```
taskInput = input("Enter the task to be removed: ")
for entry in lstTable:
    if entry["Task"] == taskInput:
        lstTable.remove(entry)
        print("Entry removed")
```

Note that this code will remove all entries that match the user input. If there are duplicate tasks that match the user input, then they will all be removed.

Saving the current to-do list to file

This section will involve iterating through the list collection in our local environment and writing the contents of each dictionary entry to our text file. The general process:

1. Create a file object for the to-do list
2. Iterate through each dictionary object in the list object
    a. Write the contents of the dictionary object to the file

```
fileObj = open(fileName, "w")
for entry in lstTable:
    fileObj.write(entry["Task"] + "," + entry["Priority"] + "\n")
fileObj.close()
```

Summary

In summary, we have learned about the dictionary collection in Python to make a script that creates and edits a to-do list in the form of a text file. We have learned to create dictionary objects that map keys to values representing a task, remove dictionary objects by looking up a particular value associated with a key, and access values associated with keys in a dictionary.