Jeff Wichmann

May 17, 2020

IT FDN 100 A Sp 20: Foundations of Programming: Python

Assignment 5

Python: Create and Manage a To Do List

# Introduction

This paper outlines the steps and process to create a programming script in Python to create a To Do List. The To Do List will include two series of data, task and priority. That is, the task that needs to be accomplished and the priority of that task. In addition to inputting tasks, the program will provide a menu of choices, allowing the user to 1) display current data, 2) add a new item to the list, 3) remove an item from the list, 4) save the data to a text file, and 5) exit the program. My goal is to create a document that a Python user can utilize to complete this coding exercise. Additionally, I can use this as a reference to refresh my memory in the future, if necessary. A subsequent draft of this assignment will be completed with the corrected script. I am using Mac OS.

# Lists

Like tuples, lists are another way to work with sequences of data. Lists do everything that a tuple does but more. Lists are mutable or able to be changed (Dawson M., Python Programming, Third Edition, Course Technology, a part of Cengage Learning, 2010). A simple list, as provided by Dawson, is below (Figure 1).

A screenshot of a cell phone

Description automatically generated

**Figure 1: Example Python code created by Michael Dawson**

The program in figure 1 creates a list, assigns it the name inventory, and prints each element of the list (Figure 2).

A screenshot of a cell phone

Description automatically generated

**Figure 2: Output in PyCharm**

# Dictionaries

Dictionaries are another way to work with data. While lists work with sequences of data, dictionaries work with pairs of data and allow the user to identify columns, or otherwise known as keys. Like a dictionary on a bookshelf, Python dictionaries associate one key with one value. Each dictionary contains one or more items (Dawson M., Python Programming, Third Edition, Course Technology, a part of Cengage Learning, 2010). Each item consists of a key, a colon, and the key's value. Commas separate each item if the dictionary has more than one item.

While items in a list are surrounded by brackets, items in a dictionary are surrounded by curly brackets or curly braces. Michael Dawson prefers “curly brackets”. Randal Root prefers “curly braces”. An example of an item from the dictionary shown below is "Link Rot" : "the process by which web pages links become obsolete." This example dictionary has six items (Figure 3).

A screenshot of a cell phone

Description automatically generated

**Figure 3: Example Python code created by Michael Dawson**

# Writing data to a file

Normally, the data entered into a Python program is lost when the program is closed. It's possible to save the data by writing data to a file. The file can be a simple text file or a database. In this example, the following code (Figure 4) will save the data to a text file. Note that objFile.close() is necessary to close the file. Not including this line would be “sloppy coding", according to Randal Root (Root R., Assignment03 IT FDN 100 A Sp 20: Foundations of Programming: Python, University of Washington, 2020).



**Figure 4: Example script in IDLE**

# Reading data from a text file into a List

Python can also read data from a file such as a text file and then put the data into a List. Once read, Python then puts the information into memory where it can be manipulated and put into the List. In this example, the following code (Figure 5) reads the data from a text file called HomeInventory.txt. This example code was created by Randal Root (Root R., Assignment05 IT FDN 100 A Sp 20: Foundations of Programming: Python, University of Washington, 2020).

A screen shot of a smart phone

Description automatically generated

**Figure 5: Example code created by Randal Root**

Figure 6 shows the output in PyCharm.

A close up of a logo

Description automatically generated

**Figure 6: Example output in PyCharm**

Figure 7 shows the HomeInventory.txt file.

A screenshot of a cell phone

Description automatically generated

**Figure 7: Example list within a text file**

# Reading data from a text file into a Dictionary

Python can also read data from HomeInventory.txt (Figure 7) and put the data into a Dictionary. The code in figure 8 was created by Randal Root (Root R., Assignment05 IT FDN 100 A Sp 20: Foundations of Programming: Python, University of Washington, 2020).

# A screenshot of a cell phone Description automatically generated

**Figure 8: Example code created by Randal Root**

Figure 9 shows the output in PyCharm.

A picture containing drawing

Description automatically generated

**Figure 9: Output in PyCharm**

# Removing items from the list

This program also has the ability to remove items from the To Do List. The code in figure 10, also written by Randal Root, shows how to do this (Root R., Assignment03 IT FDN 100 A Sp 20: Foundations of Programming: Python, University of Washington, 2020). It’s important to clarify that this code does not remove the data from the text file. The data is removed from the list in the computer’s memory. The item will be removed from the text file when the entire list is written to the text file. The old list, which contained the deleted item, is overwritten by the new list.

A screen shot of a social media post

Description automatically generated

**Figure 10: Example code written by Randal Root.**

# The assignment – saving a to do list

The script assignment for Module 5 is to modify the file Assigment05\_Starter.py where prompted. The prompt was marked in the program # TODO: Add Code Here. The first step (Step 1) was to add code that loaded existing data from the ToDoList.txt file. The next step (Step 3) was to create code that showed the To Do List items that were saved in memory. Step 4 called for code that would add a new item. Step 5 required code to remove an item from the To Do List in memory. Step 6 asked for code that saved the list of tasks and priorities to the ToDoList.txt file. Finally, Step 7 code would exit the program. (Root R., Assignment05 IT FDN 100 A Sp 20: Foundations of Programming: Python, University of Washington, 2020). This script follows in figure 11.

# ------------------------------------------------------------------------ #  
# Title: Assignment 05  
# Description: Working with Dictionaries and Files  
# When the program starts, load each "row" of data  
# in "ToDoToDoList.txt" into a python Dictionary.  
# Add the each dictionary "row" to a python list "table"  
# ChangeLog (Who,When,What):  
# RRoot,1.1.2030,Created started script  
# JWichmann, 5/17/2020, Added code to complete assignment 5  
# ------------------------------------------------------------------------ #  
  
# -- Data -- #  
# declare variables and constants  
objFile = "ToDoList.txt" # An object that represents a file  
strTask = '' # The name of the task  
strPriority = '' # The value or priority of the task  
strChoice = "" # A Capture the user option selection  
dicRow = {} # The Task and Priority will not go into a list but into a dictionary. A row of data separated into elements of a dictionary {Task,Priority}  
lstTable = [] # A list that acts as a 'table' of rows. This incudes multiple rows of data.  
strData = "" # A row of text data from the file  
strMenu = "" # A menu of user options  
Row = [] # A row of text data from the file  
Table = [] # A table of rows  
  
  
# -- Processing -- #  
# Step 1 - When the program starts, load the any data you have  
# in a text file called ToDoList.txt into a python list of dictionaries rows (like Lab 5-2)  
  
objFile = open("ToDoList.txt", "r") # David B confirmed that we should manually create a text file before starting.  
for row in objFile:  
 lstRow = row.split(",")  
 dicRow = {"Task": lstRow[0], "Priority": lstRow[1].strip()}  
 lstTable.append(dicRow)  
objFile.close()  
  
# reminders:  
# -when you read from the text file and display the data on the screen, you want to remove the carriage return (so use the strip function)  
# -when you write to the text file, you want to include the carriage return which is + '\n\  
  
  
# -- Input/Output -- #  
# Step 2 - Display a menu of choices to the user  
while (True):  
 print("""  
 Menu of Options  
 1) Show current data  
 2) Add a new item.  
 3) Remove an existing item.  
 4) Save Data to File  
 5) Exit Program  
 """)  
 strChoice = str(input("Which option would you like to perform? [1 to 5] - "))  
 print()  
 # Step 3 - Show the current items in the table  
 if (strChoice.strip() == '1'):  
  
 print("Your Current Data Is")  
 for objRow in lstTable:  
 print(objRow)  
  
 continue  
 # Step 4 - Add a new item to the list/Table  
  
 elif (strChoice.strip() == '2'):  
  
 print(" Type in a Task and Priority")  
 strTask = str(input(" Enter a Task: "))  
 strPriority = str(input(" Enter the Priority: "))  
 lstTable.append({"Task": strTask, "Priority": strPriority})  
   
  
  
 continue  
 # Step 5 - Remove a new item from the list/Table  
 elif (strChoice.strip() == '3'):  
  
 strTask = input("Task to Remove: ")  
 for row in lstTable:  
 if row["Task"].lower() == strTask.lower():  
 lstTable.remove(row)  
 print("row removed")  
 print(lstTable, '<< List with Dictionary objects')  
 else:  
 print("row not found")  
 print(lstTable, '<< List with Dictionary objects')  
 continue  
 # Step 6 - Save tasks to the ToDoList.txt file  
 elif (strChoice.strip() == '4'):  
 objFile = open("ToDoList.txt", "w")  
 for row in lstTable:  
 objFile.write(str(row["Task"]) + ',' + str(row["Priority"] + '\n') )  
 objFile.close()  
 print(" Data was saved!")  
  
  
 continue  
 # Step 7 - Exit program  
 elif (strChoice.strip() == '5'):  
 break  
 else:  
 print(" Please choose only 1, 2, 3, 4, or 5! ")

**Figure 11: Example Python Script**

# Images showing the script working

Since the program is so long, the functional code is broken into multiple pieces. Note that I start with the ToDoList.txt shown in figure 12.

A screenshot of a cell phone

Description automatically generated

**Figure 12: Example text file**

# 1) Show current data

The image below shows current data being displayed (Figure 13). This data was read from ToDoList.txt file when the program began.

# A screenshot of a cell phone Description automatically generated

**Figure 13: Output in PyCharm**

# 2) Add a new item

Figure 14 shows an item being added (paint; medium) and the new list.

A screenshot of a cell phone

Description automatically generated

**Figure 14: Output in PyCharm**

# 3) Remove an existing item

Figure 15 shows the item weed removed.

A close up of text on a black background

Description automatically generated

**Figure 15: Output in PyCharm**

# 4) Save Data to File

Figure 16 shows the output in PyCharm and the new text file (Figure 17).

A close up of text on a black background

Description automatically generated

**Figure 16: Output in PyCharm**

The newly saved ToDoList.txt file (Figure 17) which includes the new item (paint; medium) and does not include the removed item (weed; low).

A screenshot of a cell phone

Description automatically generated

**Figure 17: Example text file**

# 5) Exit Program

As show in figure 18, the program exited properly.

A screenshot of a cell phone

Description automatically generated

**Figure 18: Output in PyCharm**

# Running the script in OS command

Figure 19 shows the first half of the script running in OS command.

A screenshot of text

Description automatically generated

**Figure19: Output in OS command**

Figure 20 shows the second half of the script running in OS command.

**A screenshot of a social media post

Description automatically generated**

**Figure 20: Output in OS command**

# Summary

In summary, this document outlines the steps and process to create a programming script. The goal was to create a To Do List that included two series of data, task and priority. Another goal was to make the file dynamic, allowing the user to 1) show data, 2) add an item, 3) remove an item, 4) save the data to a text file, and 5) exit the program. The document also discussed Dictionaries which is a new way to work with data. This document acts as guide to a Python user or for me, if I need a refresher at some point in the future.