Nathaniel Smith

May 17, 2020

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

Assignment05

Python: Script with Lists and Dictionary

# Introduction

In this paper I discuss the Python code in the script called Assignment05\_Starter.py. In this assignment I was provided with a template and it was my job to modify the code so that the script would run as desired. The script reads a ToDoList.txt file. The data in the text file consists of a list of tasks and their corresponding priorities. The script allows the user to see this list and add to the list or remove items from the list. And when the user is done, the user has the option to save changes to the text file or exit without saving. This script reads and writes to a file named ToDoList.txt, but uses dictionaries and lists to process the data in memory before it is written back to the file. This script makes use of while loops, for loops, if-elif-else statements, try-except statement, lists and dictionaries. I will carefully explain the usage of each in my code throughout this paper.

# Script Header

As with writing all scripts, I started Assignment05\_Starter.py with a script header, see lines 1 through 10 in Figure 1. The purpose of the script header is to describe what the script does and when it was created. And just as I have done, it’s also a good place to store the change log for the script. From the change log you can see that on May 14 I added code to complete this assignment.

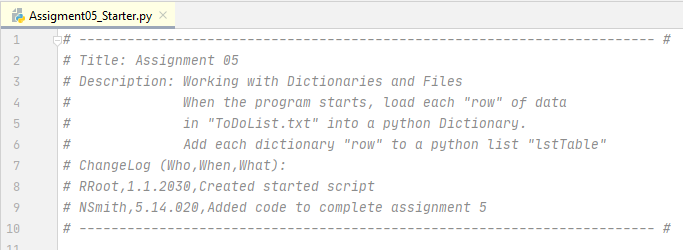


Figure 1. Assignment05\_Starter.py Script Header

# Pseudocode

Next, I wrote the pseudocode for my script (well not really, it was provided by the instructor, Randal Root). Pseudocode is a way of writing the code in text form. It is sort of an outline for the code. The pseudocode is not the final code, instead it is used to aid in writing the code. The pseudocode should be written so that it is easy to follow and include all the steps that are needed for the code to work. Without it, it can make writing the code hard to do. Good pseudocode can be used to write code in any programming language not just Python. Below is the pseudocode for this assignment.

*# Step 1 - When the program starts, load the data you have*

*# in a text file called ToDoList.txt into a python list of dictionaries rows*

*# Step 2 - Display a menu of choices to the user*

*# Step 3 - Show the current items in the table*

*# Step 4 - Add a new item to the list/Table*

*# Step 5 - Remove a new item from the list/Table*

*# Step 6 - Save tasks to the ToDoList.txt file*

*# Step 7 - Exit program*

# The Code

The code is too long to present in this paper. The code can be found in the file Assignment05\_Starter.py which is stored in the same folder as this paper.

# Separations of Concern

The template provided was organized into three sections following the design principle known as separation of concerns. As such it was divided into three sections: data, processing, and input/output.

# Data Section: Declare Variables

All variables used in the script were declared in the data section. Strings are declared with a pair of double quotation marks, lists are declared with left square bracket followed by right square bracket, dictionaries are declared by left curly bracket followed by right curly bracket, and integers are declared using any non-decimal number. The data section with comments is shown below.

# -- Data -- #

# declare variables and constants

strFile = "ToDoList.txt" #name of file

objFile = None # File handle, An object that represents a file

strData = "" # A row of text data from the file

lstRow = [] # Text data from file stored as a list

dicRow = {} # A row of data separated into elements of a dictionary {Task,Priority}

lstTable = [] # A list that acts as a 'table' of rows

strMenu = "" # A menu of user options

strChoice = "" # A Capture the user option selection

count = 0 # A number to count the rows in table

strMax = "" # A string to store the count as a string

intChoice = 0 # A number to store the strChoice as integer

# Processing Section

The processing section includes the first step of the pseudocode which is to load the data from the text file ToDoList.txt. into a dictionary object and then add the dictionary object to a list. The code is shown below.

objFile = open(strFile, "a")

objFile = open(strFile, "r")

for strData in objFile:

lstRow = strData.split(",")

dicRow = {"task":lstRow[0],"priority":lstRow[1].strip()}

lstTable.append(dicRow)

objFile.close()

I first opened the file with the open() function and the “a” mode i.e. append mode which opens a file for writing but will append any new data to the end of the file. This will also create the file if it does not exist. I could have left out this line of code, but it would give an error if the file did not exist. So this line is really only here in the event that the script is executed and the file ToDoList.txt does not yet exist.

The very next line of code is another open() function, but this time it uses the “r” mode which stands for read. I need to read the contents of the ToDoList.txt file. The append mode does not allow for this which is why I had to immediately open the file in read mode after opening it in append mode.

The for loop is reading each line of text in the file one line at a time. The line of text in the file includes a task like washing dishes followed by a comma followed by a priority like the number 7. So for each line of text in objFile the for loop will read that line and store it in variable strData.

The string method .split seen in line 4 above is splitting strData into two strings, the task before the comma and the priority after the comma and this is being stored into the list lstRow.

lstRow is then processed into a dictionary, dicRow, in the fifth line of code above. dicRow is assigned using the left curly brace, followed by a key “task” followed by the colon followed by lstRow[0] (list element at index position 0) followed by a comma, followed by a key “priority” followed by a colon followed by lstRow[1] (list element at index position 1) followed by .strip() method to remove any whitespace (in this case to remove a carriage return) followed by the right curly bracket.

dicRow is appended to the list lstTable using the list method .append().

After the for loop iterates through all lines of text in the data file, the file is closed with the .close() method. 6

# Display Menu to User

This is step 2 of the pseudocode and it belongs to the into the input/output section of code. In fact, the remaining steps, 3-7, all belong to the input/output section. In this step a menu is displayed to the user that let’s the user choose between 5 options, see the code below.

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 = input("Which option would you like to perform? [1 to 5]: ")

print()

The code above starts with an infinite while loop having a True condition. The block of the while loop contains all the input/output of the script and it contains steps 2-7 of the pseudocode (not all shown above, steps 3-7 will be discussed throughout the following sections of this paper). The display menu code of the while loop occurs in the first 10 lines of the while loop. The first 8 lines print to screen the menu options. Then the input() function is used to prompt the user to select an option by choosing a number 1 to 5. The user’s choice is stored in the string variable strChoice. And finally I used the print() function with empty parentheses to print a newline on screen for aesthetics.

# Show Items on the To Do List

This is step 3 of the pseudocode. This prints the to do list to screen if the user chose option 1 from the menu. The code for step 3 is shown below.

if (strChoice.strip() == '1'):

print("\t---ToDo List---")

print("\tTask | Priority")

for row in lstTable:

print("\t",row["task"]," | ",row["priority"],sep='')

The if statement checks if strChoice is equal to the string ‘1’. I used the .strip() method to remove any whitespace from strChoice in the event that the user hit the space bar for example after typing 1 and before hitting enter.

If the user entered ‘1’, the if expression evaluates to True, and the ToDo list is printed to screen. A title is printed “---ToDo List---” followed on a new line by column titles “Task | Priority”.

The for loop iterates over each element in lstTable, and each element of lstTable is a dictionary. For each dictionary element in lstTable a line of text prints the dictionary value for key “task” followed by “ | “, followed by the dictionary value for “priority”. Dictionary objects use key indexing instead of position indexing. This is why the key names are used in square brackets following the dictionary name, row. The optional argument sep=’’ is used in the parentheses of the print() function. This was done to prevent extra spaces from printing to screen because the default is to add a space between each value in the parentheses of the print() function.

# Add Items to the To Do List

This is pseudocode step 4. Each time this step runs a new to do item is added to the to do list. It only runs if the user picked option 2 from the menu. If so, it will ask the user to enter a task and a priority number for the task. This input from the user is saved as a dictionary object and then it is appended to the to do list. Let’s look at the code below.

elif (strChoice.strip() == '2'):

dicRow = {"task":input("Enter task: "),"priority":input("Enter Priority: ")}

lstTable.append(dicRow)

The elif expression checks if strChoice is equal to ‘2’. Again I used the .strip() method to remove whitespace. If this evaluates to True, the block statement is executed.

The user is asked to enter a task. The input from the user is stored in the key “task” of the dictionary dicRow. The user is then asked to enter a priority. The input is stored in the key “priority” of dicRow.

dicRow is then appended to the list lstTable.

# Remove an Item from the Table

This is pseudocode step 5. This will only execute if the user selects option 3 from the menu. If so, a numbered list representing the current to do list is printed to screen, and the user is asked to choose an item to remove from the list by choosing its corresponding number. The item is then removed from the to do list. If the to do list is empty the user is told that there is nothing to remove. If the user enters a number not on the list, they are told it is not an option and are returned to the main menu. If the user enters a non-integer value, they are told the input is not valid and again are returned to the main menu. Let’s take a look at the code below.

elif (strChoice.strip() == '3'):

count = 0

for row in lstTable:

count += 1

print("\t",count,": ",row["task"]," ",row["priority"],sep='')

if count == 0:

print("Nothing to remove. Try option 2 to add a task/priority.")

else:

strMax = str(count)

strChoice = input("\nWhich item should be removed?[1 - " + strMax +"]: ")

try:

intChoice = int(strChoice.strip())

if (intChoice >= 1 and intChoice <= count):

lstTable = lstTable[:intChoice-1] + lstTable[intChoice:]

else:

print("'",strChoice.strip(),"' is not an option! Let's start over.",sep='')

except:

print("'",strChoice.strip(),"' is not valid. Did you choose a number? Let's start over.",sep='')

The expression of the elif statement checks if strChoice is equal to the string ‘3’ (notice .strip() method is used again). If this evaluated to True the block statement executes.

count is reinitialized to zero. Count will be used to number the rows printed to screen.

The for statement iterates for each dictionary row of list lstTable. This will only execute if lstTable is not empty. Each iteration results in the count augmenting by 1. print() function is used to print the count followed by the value stored in the dictionary row corresponding to the key “task” followed by a space followed by the value stored in the dictionary row corresponding to the key “priority”.

Next the code checks if count is equal to 0 using an if statement. This will only be true if the previous for loop did not iterate and this would only occur if the list lstTable is empty. In the case that lstTable is empty i.e. the count is equal to 0, the message "Nothing to remove. Try option 2 to add a task/priority” is printed to screen.

If count is not equal to 0 i.e. lstTable is not empty, the else statement executes.

The first statement of the else block sets the strMax equal to the count converted to string. The second statement uses the input() function to prompt the user to choose one of the items to be removed from the items numbered from 1 to strMax. strMax is just the count (converted to string) of the number of dictionary elements in lstTable.

The user input is then checked by a try-except statement. The except statement will only run if the try statement errors. And the try statement will only error if the user does not enter an integer value.

Let’s look at the try statement first. The user input, strChoice, is converted to an integer and stored in the variable intChoice (again I used .strip() method to remove whitespace). An if statement checks to see if intChoice is less than or equal to 1 *and* less than or equal to the variable count i.e. the number of dictionary elements in lstTable. If the expression is True, lstTable is assigned a new value using slicing. The first slice is lstTable from index position 0 up to the value of intChoice subtracted by 1, and this is added to the slice of lstTable from index position intChoice to the last index position. The result is that the item represented by the number intChoice is removed from lstTable. If the expression is False, the else statement executes, a message prints to screen telling the user their choice was not an option, and the program returns to the main menu.

Now a look at the except statement. Again, this will only execute if the try statement errors. And the try statement only errors if the user did not enter an integer value when asked to choose an item to remove from the list. The error would occur during the conversion of strChoice to an integer with the int() function. The int() function returns an error if the string being converted is not a string numeral. If that is the case, the except statement executes, the message, “’’ is not valid. Did you choose a number? Let's start over” is printed to screen and the program returns to the main menu.

# Save To Do List to ToDoList.txt File

This is pseudocode step 6. It saves the current list of to do items to file if the user chose menu option 4. The code for step 6 is shown below.

elif (strChoice.strip() == '4'):

objFile = open(strFile, "w")

for row in lstTable:

objFile.write(row["task"] + "," + row["priority"] + "\n")

objFile.close()

The expression of the elif statement checks if strChoice is equal to the string ‘4’ (notice .strip() method is used again). If this evaluated to True the block statement executes.

The open() function is used to open the file with filename having the value stored in strFile, which is ToDoList.txt. Mode “w” is used to open the file in write mode and truncate the file if it already exists. This is assigned to the file object objFile.

The for loop iterates over each element, row, in the list lstTable. lstTable is a list of dictionary objects so each row is a dictionary object.

The .write() method writes data to the file. Within parentheses is a concatenated string that consists of the value of key “task” of dictionary row plus a comma plus the value of key “priority” plus a newline. This will write task name followed by a comma followed by the priority rating followed by a carriage return. So that each line written to file will include the two elements of a dictionary item separated by a comma. The result of this is shown in Figure 4.

The final line closes the object file with the .close() method.

# Exit the Program

This is pseudocode step 7. This only executes if the user chose option 5 from the menu. The code is below.

elif (strChoice.strip() == '5'):

print("Goodbye!")

break # and Exit the program

The expression of the elif statement will check if strChoice without whitespace is equal to the string ‘5’. If True, the message “Goodbye!” is printed to screen.

The break statement terminates the enclosing while loop and because there is no code after the while loop, the script terminates.

# Invalid Choice

If the user did not correctly enter an option from 1 to 5 a message is printed to the user showing what the user entered and to please choose a number 1 to 5. See the code below.

else:

print("You entered '%s'. Please choose a number 1 to 5." % strChoice)

If neither the if statement nor the elif statements of steps 2-7 evaluate to True, this else statement will execute. A message is printed to the screen telling the user to choose again. Within the print() function I used string substitution to make the statement reader friendly; strChoice will be substituted for %s. If the user were to enter 6, for example, the following message will appear: “You entered ‘6’. Please choose a number 1 to 5.” After the message is printed the script will loop back to the while loop statement and print the menu to the user. The user will have the option to choose again.

# Testing the Script

I first ran the script using the Interactive Development Environment (IDE) software from JetBrains called PyCharm. The result is shown in Figure 3. The code works as intended!

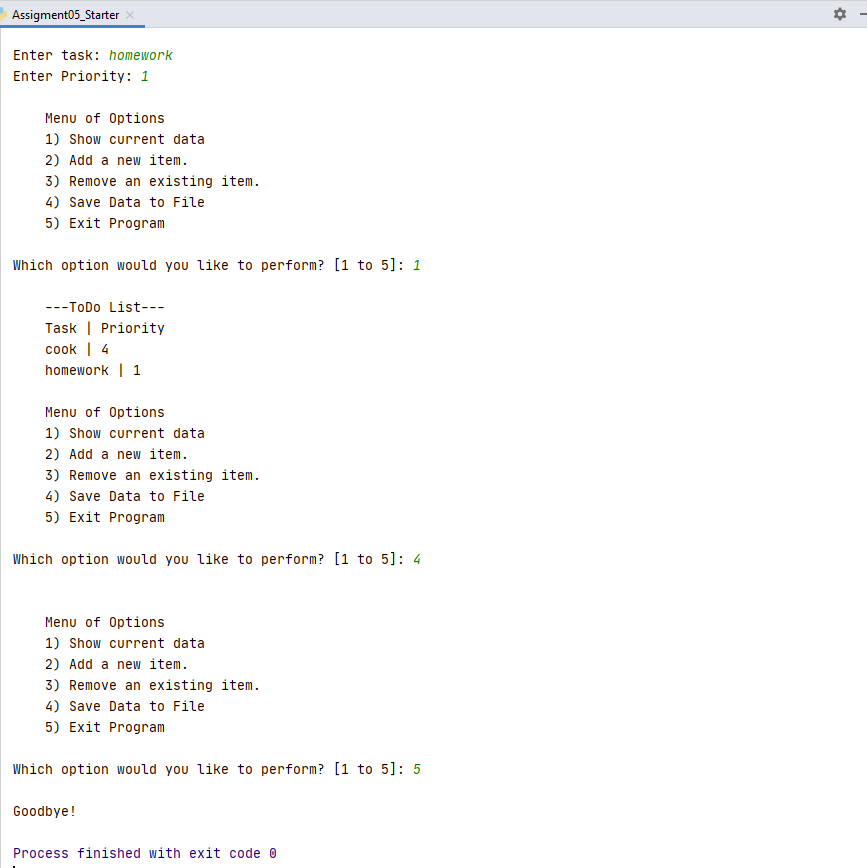


Figure 3. Running Assignment05\_Starter.py in PyCharm

I checked the text file after running the code in PyCharm and it did work. Two items were added to the ToDoList.txt see Figure 4.

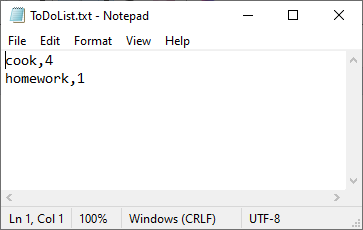


Figure 4. ToDoList.txt After Running Script in PyCharm

Now that I had it running in PyCharm, I also tested my script from Windows Command Prompt as shown in Figure 5. Success!

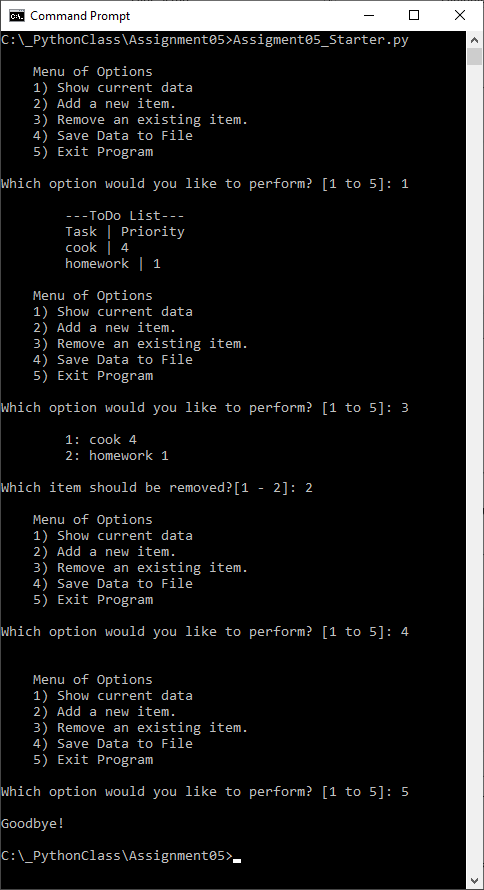


Figure 5. Running Assignment05\_Starter.py in Windows Command Prompt

After running the script in Command Prompt, I ended up with the text file shown in Figure 6. Complete success!

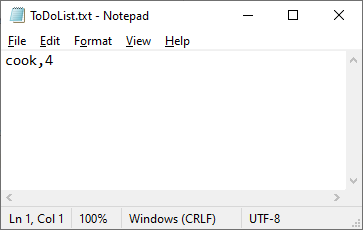


Figure 6. ToDoList.txt After Running Script in Command Prompt and Removing 1 Item

# Summary

In summary, I modified an existing script that had been written by another programmer, Randal Root. I was able to use his template and pseudocode to complete the script so that it reads a to do list from a file, lets the user add or remove items from the to do list, and then writes the changes back to the file if so desired by the user. This is also the first code I have written that uses a try-except statement to catch an error and lets the script continue running despite the error.