Alejandro Wolf-Yadlin

Professor Randal Root

IT FDN 100 A

11/07/2016

Assignment 105

In this assignment, we worked with lists and dictionaries to make a to do list of household tasks and their priorities. We used functions to solve each requirement of our ToDo list. We incorporated them within a list management class and called them in a “runall” function to manage our to do list.

**We wrote the script “Assignement06.py” to perform this task. The script (including functions) is pasted below:**

1. #-------------------------------------------------#
2. # Title: Working with Functions
3. # Dev:   ayadlin
4. # Date:  November 7, 2016
5. # ChangeLog: (Who, When, What)
6. #   Based on RRoot, 11/02/2016 Assignment 5 solution
7. #-------------------------------------------------#
9. #-- Data --#
10. # declare variables and constants
11. # objFile = An object that represents a file
12. # strData = A row of text data from the file
13. # dicRow = A row of data separated into elements of a dictionary {Task,Priority, Number} # Number is added as function of priority
14. # lstTable = A dictionary that acts as a 'table' of rows
15. # strMenu = A menu of user options
16. # strChoice = Capture the user option selection
18. #-- Input/Output --#
19. # User can see a Menu (Step 2)
20. # User can see data (Step 3)
21. # User can insert or delete data(Step 4 and 5)
22. # User can save to file (Step 6)
24. #-- Processing --#
26. # Step 1
27. # When the program starts, load the any data you have
28. # in a text file called ToDo.txt into a python Dictionary.
30. # Step 2
31. # Display a menu of choices to the user
33. # Step 3
34. # Display all todo items to user
36. # Step 4
37. # Add a new item to the list/Table
39. # Step 5
40. # Remove a new item to the list/Table
42. # Step 6
43. # Save tasks to the ToDo.txt file
45. # Step 7
46. # Exit program
47. #-------------------------------
49. # Global Variables
51. FileName = "C:\\_PythonClass\ToDo.txt" # Default File to read and write to.
52. strData = ""
53. dicRow = {}
54. lstTable = []


58. # Prologue
59. # Function 0
60. # This function assigns a numerical value to the different levels of priority
61. # 1 for high, 2 for medium and 3 for low - All other options will be assigned 4
62. # We will use this function to sort our to do list by order of priority
64. **def** numerizepriority(x=dicRow):
65. **if** x["Priority"] == "high":
66. x["Number"] = 1
67. **elif** x["Priority"] == "medium":
68. x["Number"] = 2
69. **elif** x["Priority"] == "low":
70. x["Number"]=3
71. **else**:
72. x["Number"]=4

75. # First I will group all functions within a class I'll call list managmenet
77. **class** listmanagement():
78. """ This class contains methods for processing simple 2 column lists """

81. # Function 1
82. # Make a fucntion to load the task/priority list into a dictionary and then a list
84. **def** loadtodolist(objFileName = FileName):
85. """
86. This function loads a to do list from a txt file
87. :Input: file.txt
88. :Default input: C:\\_PythonClass\ToDo.txt
89. """
90. objFile = open(objFileName, "r")
91. **for** line **in** objFile:
92. strData = line.split(",") # readline() reads a line of the data into 2 elements
93. dicRow = {"Task":strData[0].strip(), "Priority":strData[1].strip()}
94. numerizepriority(dicRow)
95. lstTable.append(dicRow)
96. objFile.close()
98. # Here we wuse a lambda function to sort the to do list according to priority
99. lstTable.sort(key=**lambda** x:x["Number"])
101. # The function will return our list
102. **return**(lstTable)

105. # Function 2
106. # Make a fucntion to display current tasks
108. **def** currenttasks(task = lstTable):
109. """
110. This function prints the todo list liaded from the text file above
111. :Input: No input
112. """
113. **print**("\n\*\*\*\*\*\*\* The current items ToDo are: \*\*\*\*\*\*\*")
114. **for** row **in** lstTable:
115. **print**(row["Task"] + "(" + row["Priority"] + ")")
116. **print**("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")
117. #return(lstTable)

120. # Function 3
121. # Make a fucntion that adds a new item to the list/Table
123. **def** addtask():
124. """
125. This function adds tasks to our to do list
126. :Input: no input
127. """
128. strTask = str(input("What is the task? - ")).strip()
129. strPriority = str(input("What is the priority? [high|medium|low] - ")).strip()
130. dicRow = {"Task":strTask,"Priority":strPriority}
131. numerizepriority(dicRow)
132. lstTable.append(dicRow)
134. # Here we wuse a lambda function to sort the updated to do list according to priority
135. lstTable.sort(key=**lambda** x:x["Number"])
136. **return**(lstTable)
138. # Function 4
139. # Make a function that remove an item from the list/Table
140. @staticmethod           # Not sure if this is needed?
141. **def** removetask():
142. """
143. This function rmoves tasks to our to do list
144. :Input: no input
145. :Default input: C:\\_PythonClass\ToDo.txt
146. """
147. #4a-Allow user to indicate which row to delete
148. strKeyToRemove = input("Which TASK would you like removed? - ")
149. blnItemRemoved = False #Creating a boolean Flag
150. intRowNumber = 0
151. **while**(intRowNumber < len(lstTable)):
152. **if**(strKeyToRemove **in** str(lstTable[intRowNumber])):
153. **del** lstTable[intRowNumber]
154. blnItemRemoved = True
155. #end if
156. intRowNumber += 1
157. #end while loop
158. #4b-Update user on the status
159. #print(lstTable)
160. **if**(blnItemRemoved == True):
161. **print**("The task was removed.")
162. **else**:
163. **print**("I'm sorry, but I could not find that task.")
164. **return**(lstTable)

167. # Function 5
168. # Make a fucntion that save the tasks to the ToDo.txt file
170. **def** savetodolist(objFileName = FileName):
171. """
172. This function adds tasks to our to do list
173. :Input: file.txt where to do list will be saved
175. """
176. **if**("y" == str(input("Save this data to file? (y/n) - ")).strip().lower()):
177. objFile = open(objFileName, "w")
178. **for** dicRow **in** lstTable:
179. objFile.write(dicRow["Task"] + "," + dicRow["Priority"] + "\n")
180. objFile.close()
181. input("Data saved to file! Press the [Enter] key to return to menu.")
182. **else**:
183. input("New data was NOT Saved, but previous data still exists! Press the [Enter] key to return to menu.")


187. ###############################################################################################################################
188. # Now that functions are written develope the same code as for assignment 5,
189. # but using the 5 functions defined above

192. # Define a single funcion that run the other functions from listmanagement class together
193. **def** runall(objFileName = FileName):

196. # Use function 1 from listmanagement class to Load table from file
197. lstTable = listmanagement.loadtodolist()
199. # step 2 Provide menu of options and run functions
201. **while**(True):
202. **print** ("""
203. Menu of Options
204. 1) Show current data
205. 2) Add a new item.
206. 3) Remove an existing item.
207. 4) Save Data to File
208. 5) Exit Program
209. """)
210. strChoice = str(input("Which option would you like to perform? [1 to 5] - "))
211. **print**()#adding a new line
213. # Step 3 use function 2 from listmanagement class to show the current items in the table
214. **if** (strChoice.strip() == '1'):
215. listmanagement.currenttasks()
216. **continue** #to show the menu
218. # Step 4 use function 3 from listmanagement class to add a new item to the list/Table
219. **elif**(strChoice.strip() == '2'):
220. listmanagement.addtask()
221. listmanagement.currenttasks()
222. **continue** #to show the menu
224. # Step 5 use function 4 from listmanagement class to remove an item from the list/Table
225. **elif**(strChoice == '3'):
226. #removetask()
227. listmanagement.currenttasks()
228. listmanagement.removetask()
229. listmanagement.currenttasks()
230. **continue** #to show the menu
232. # Step 6 use function 5 from listmanagement class to save tasks to the ToDo.txt file
233. **elif**(strChoice == '4'):
234. listmanagement.currenttasks()
235. listmanagement.savetodolist()
236. **continue** #to show the menu
238. # Step 7 get out of program if user wants to exit
239. **elif** (strChoice == '5'):
240. **break** #and Exit the program
242. **else**:
243. **print**('please provide a valid option\n')
244. **continue**

247. # Call runall to view/modify the to do list
248. runall()


252. #############################################################################################################################################

255. ### Question About Function $
257. ### Function 4
258. ### Make a function that remove an item from the list/Table
259. ##    @staticmethod
260. ##    def removetask():
261. ##            #4a-Allow user to indicate which row to delete
262. ##            strKeyToRemove = input("Which TASK would you like removed? - ")
263. ##            blnItemRemoved = False #Creating a boolean Flag
264. ##            intRowNumber = 0
265. ##            while(intRowNumber < len(lstTable)):
266. ##                if(strKeyToRemove in str(lstTable[intRowNumber])):
267. ##
268. ##
269. #############################################################################################################################################
270. ## Question -  debugging I saw that when calling this adapted line from Randy's code sometimes Task would be column 1,                      #
271. ## sometime column 2, etc -  which made me write the code below - which works, but is cumbersome ( I didn't use it finally)                 #
272. ## why does the location of items in the list change with different runs of the program?                                                    #
273. ## If you look at the print values below some times it will go [ Task, priority, number], but other times [priority, task number] Why?      #
274. ## how can it be set so it is always for example [task, priority, number]                                                                   #
275. ##                                                                                                                                          #
276. ##                 print(str(list(dict(lstTable[intRowNumber]).values())))                                                                  #
277. ##                 if (strKeyToRemove == str(list(dict(lstTable[intRowNumber]).values())[1])) or \                                          #
278. ##                (strKeyToRemove == str(list(dict(lstTable[intRowNumber]).values())[1])) or \                                              #
279. ##                (strKeyToRemove == str(list(dict(lstTable[intRowNumber]).values())[2])): #the values function creates a list!             #
280. #############################################################################################################################################
281. ##
282. ##                    del lstTable[intRowNumber]
283. ##                    blnItemRemoved = True
284. ##                #end if
285. ##                intRowNumber += 1
286. ##            #end while loop
287. ##            #4b-Update user on the status
288. ##            #print(lstTable)
289. ##            if(blnItemRemoved == True):
290. ##                print("The task was removed.")
291. ##            else:
292. ##                print("I'm sorry, but I could not find that task.")
293. ##            return(lstTable)
294. ##

**The key functions and commands learned here are:**

* We learned to define a function using “def” and assign it return using, well, “return”
* We learned to use class command to group a set of functions within the same class and to call functions in the class using class.function
* While: this command allow us to repeat a set of code while a condition holds: *while x < 5*
* For loops: this command allow us to iterate through a variable, table, tuple string as long as we are within range or dimension: *for i in range x or for I in range(0,n)*
* If/elif/else: this command allow us to anticipate different scenarios an provide adequate code according to ouropearting conditions: *if a do b, elif c do d else do e*.
* Manipualtion of list and dictionaries: we learned to build to assign new lines to lists and values to dictionary keys, for example:

list.append([nt,np]) appends [nt,np] to a list of lists

list.remove(row) will remove row from list

dict[nt]=np gives the value np to the key nt

* Learn the mini function lambda

list.sort(key=lambda x:x[Number]) in this case lambda calls the sort function to sort list according to the column named “Number”

* We learned to extract all elements of a list into string to write the into file:

for line in list:

iterate through each sublist in list

task = ','.join(str(x) for x in line)

find all string in the sublist and join them ny ‘,’

myfile.write(task + '\n')

write the strings to file and move to next line

**We verified simple code worked by using one of the protocols from assignment 101:**

* 1. Open windows command window and run program from there:

Write code (verbatim):

start python.exe c:\\_PythonClass\todolist.py

* 1. Open c:\\_PythonClass folder and run program from there:
     1. Double click todolist.py file
     2. Right click edit with IDLE & choose RUN Module (F5 key)