

Jeff Nelson

May 18, 2020

IT FDN 100 A

Assignment #5

<https://github.com/jnelson22/IntroToProg-Python>

## Lists & Dictionaries

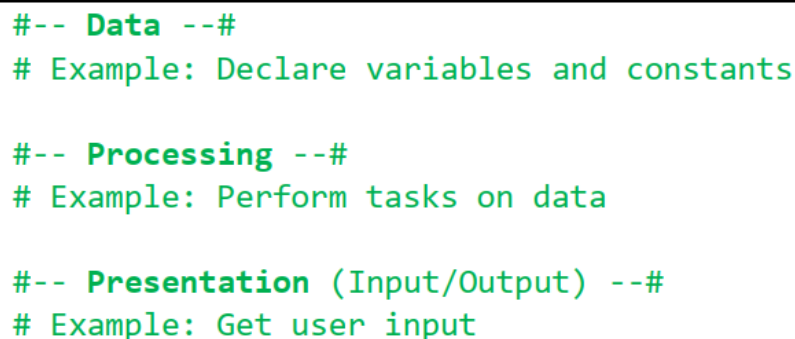
### Intro

In this paper, I will go over a few interesting and important items to remember that I learned from this week's lecture. Lastly, I will show a program that allows for the creation of a To-Do list.

### Week 4 Learnings

This week I learned about list objects, dictionaries, coding practices, functions, and GitHub. A list always holds data in the computer's memory. That means if you want to store the data in a list you'll have to write that data to a file before exiting the program. Dictionaries allow you to replace the index subscript in a list, tuple, or string with a key (character) subscript. This allows for a more natural language recall of data, similar to a spreadsheet. You use the {} to indicate the use of a dictionary.

One of the coding practices is to separate the code into sections. As you see in Figure 1 there is an example of code separated into data, processing, and presentation.



```
#-- Data --#  
# Example: Declare variables and constants  
  
#-- Processing --#  
# Example: Perform tasks on data  
  
#-- Presentation (Input/Output) --#  
# Example: Get user input
```

Figure 1: Screenshot of "separation of concerns"

Sometimes you can always order your code in the manner. A function helps with this process. A function is a named set of one or more statements. You can call these statements throughout your program.

Inside PyCharm allow you to create a script template. This allows for consistency across all your code and helps others to read it.

GitHub allows you to store your code or files in a repository. This allows for version control or sharing of your code through their online portal.

## Creating a Home Inventory Python Program

This program's problem statement is to create a ToDo list keeper. The program prompts the user with a menu that ask them to read the current tasks in the file, add a new task and priority, remove an existing task, and save all the task to a text file.

First, the program started with a provided structure similar to something that a senior level developer would give to a junior level developer, see Figure 2. This is a basic outline with “TODO” statement that PyCharm can understand and allows for earlier following.

```
# declare variables and constants
objFile = "ToDoList.txt" # An object that represents a file
strData = "" # A row of text data from the file
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

# -- 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)
# TODO: Add Code Here

# -- 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() # adding a new line for looks
    # Step 3 - Show the current items in the table
    if (strChoice.strip() == '1'):
        # TODO: Add Code Here
        continue
    # Step 4 - Add a new item to the list/Table
    elif (strChoice.strip() == '2'):
        # TODO: Add Code Here
        continue
    # Step 5 - Remove a new item from the list/Table
    elif (strChoice.strip() == '3'):
        # TODO: Add Code Here
        continue
    # Step 6 - Save tasks to the ToDoList.txt file
    elif (strChoice.strip() == '4'):
        # TODO: Add Code Here
        continue
    # Step 7 - Exit program
    elif (strChoice.strip() == '5'):
        # TODO: Add Code Here
        break # and Exit the program
```

Figure 2: Stater code

Step one was reading in the current “ToDoList” text file into a list. I did a little google searching to find the best way to make sure there was a file in the current directory. If there is no “ToDoList.txt” file located in the directory the *open* command is run there will be an error. So I found a library in Python *os.path.exists* that returns a true or false. See Figure 3 for the initial source code.

```

# -- 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)
# TODO: Add Code Here
if os.path.exists("ToDoList.txt"):
    objFile = open(objFile, "r")
    for rows in objFile:
        lstRow = rows.split(",")
        # print(lstRow)
        dicRow = {"Task": lstRow[0], "Priority": lstRow[1].strip()}
        lstTable.append(dicRow)
    objFile.close()

```

Figure 3: Step 1 beginning code

I ran into a little trouble when trying to run my code in the command line. The problem was with the file location of running python from the general prompt it would not ever find the file since the directory is different. I did a little google search for how to change the directory to the path location of the python file, see Figure 4. I also included that if the file was not found it would create a blank file.

```

# Change the absolute path of the python file location
abspath = os.path.abspath(__file__)
dname = os.path.dirname(abspath)
os.chdir(dname)

# Checks to see if there is in the same directory as the python file. If not it creates a blank file.
if os.path.exists("ToDoList.txt"):
    objFile = open(objFile, "r")
    for row in objFile:
        t, p = row.split(",")
        # print(t, p)
        dicRow = {"Task": t, "Priority": p.strip()}
        lstTable.append(dicRow)
    # print(lstTable)
    objFile.close()
else:
    objFile = open(objFile, "w")
    objFile.close()

```

Figure 4: Step 1 final

Step two was already provided as the menu. I only made a few minor changes to the text.

Step three was presenting the data that was now in the *lstTable*. Figure 5 shows the initial code of using a for loop to print each row. Figure 6 shows the final code where I added an if statement to capture when the file is blank and print that message to the user.

```
# Step 3 - Show the current items in the table
if (strChoice.strip() == '1'):
    # TODO: Add Code Here
    for row in lstTable:
        print(' ', row['Task'], ', ', row['Priority'], sep='')
    continue
```

Figure 5: Step 3 initial code

```
# Step 3 - Show the current items in the table
if (strChoice.strip() == '1'):
    # TODO: Add Code Here
    i = 0
    for row in lstTable:
        print(' ', row['Task'], ', ', row['Priority'], sep='')
        i = 1
    if (i == 0):
        print("No tasks in file. Ready to add tasks!")
    continue
```

Figure 6: Step 3 final code

Step four of choice “2” allowed the user to input new tasks and priorities. They are initially appended to the *lstTable* then they use is asked is they want to exit, see Figure 7.

```
# Step 4 - Add a new item to the list/Table
elif (strChoice.strip() == '2'):
    # TODO: Add Code Here
    while (True):
        strTask = input("What is the task: ")
        strPriority = input("What is the priority: ")
        lstTable.append({"Task": strTask.title(), "Priority": strPriority.title()})
        strChoice = input("Exit: ('y/n'): ")
        if strChoice.lower() == 'y':
            print(lstTable)
            break
    continue
```

Figure 7: Step 4 code

Step five or choice “3” allowed the user to remove items for the task list. Figure 8 shows the input that asks the user the loops through the list until it finds a match then removes it.

```

# Step 5 - Remove a new item from the list/Table
elif (strChoice.strip() == '3'):
    # TODO: Add Code Here
    while (True):
        strTask = input("Task to remove: ")
        for row in lstTable:
            if row['Task'].lower() == strTask.lower():
                lstTable.remove(row)
                print("Task removed from list")
            else:
                print("Task not found")
        strChoice = input("Exit? ('y/n'): ")
        if strChoice.lower() == 'y':
            break
    continue

```

Figure 8: Step 5 code

Step 6 or user choice “4” allows the user to save a copy of the data in the list they have been adding or removing. It simply loops through the *lstTable* and writes each element to a test file, see Figure 9.

```

# Step 6 - Save tasks to the ToDoList.txt file
elif (strChoice.strip() == '4'):
    # TODO: Add Code Here
    objFile = open("ToDoList.txt", "w")
    for row in lstTable:
        objFile.write(str(row["Task"]) + ", " + str(row["Priority"]) + "\n")
    objFile.close()
    print("Tasks saved to ToDoList.txt")
    continue

```

Figure 9: Step 6 code

Lastly, choice “5” allows the user to quit the program.

Figure 10 through Figure 13 are the screenshots of the final source code, it running in PyCharm and the command prompt.

```

## -- Input/Output -- #
## Step 2 - Display a menu of choices to the user
while (True):
    print("""
    Menu of Options

    1) Show current task list.
    2) Add a new task and priority.
    3) Remove an existing task.
    4) Save Data to ToDoList.txt
    5) Exit Program.

    """)
    strChoice = str(input("Which option would you like to perform? [1 to 5] - "))
    print() # adding a new line for looks
    # Step 3 - Show the current items in the table
    if (strChoice.strip() == '1'):
        # TODO: Add Code Here
        i = 0
        for row in lstTable:
            print(' ', row['Task'], ' ', ' ', row['Priority'], sep='')
            i = 1
        if (i == 0):
            print("No tasks in file. Ready to add tasks!")
        continue
    # Step 4 - Add a new item to the list/Table
    elif (strChoice.strip() == '2'):
        # TODO: Add Code Here
        while (True):
            strTask = input("What is the task: ")
            strPriority = input("What is the priority: ")
            # Clean up user inputs
            strTask = strTask.strip()
            strPriority = strPriority.strip()
            # Append inputs to the list
            lstTable.append({"Task": strTask.title(), "Priority": strPriority.title()})
            strChoice = input("Add another task? ('y/n'): ")
            if strChoice.lower() == 'n':
                # print(lstTable)
                break
            continue
    # Step 5 - Remove a new item from the list/Table
    elif (strChoice.strip() == '3'):
        # TODO: Add Code Here
        while (True):
            strTask = input("Task to remove: ")
            for row in lstTable:
                if row['Task'].lower() == strTask.lower():
                    lstTable.remove(row)
                    print("Task removed from list")
                else:
                    print("Task not found")
            strChoice = input("Remove another task? ('y/n'): ")
            if strChoice.lower() == 'n':
                break
            continue
    # Step 6 - Save tasks to the ToDoList.txt file
    elif (strChoice.strip() == '4'):
        # TODO: Add Code Here
        objFile = open("ToDoList.txt", "w")
        for row in lstTable:
            objFile.write(str(row["Task"]) + ", " + str(row["Priority"]) + "\n")
        objFile.close()
        print("Tasks saved to ToDoList.txt")
        continue
    # Step 7 - Exit program
    elif (strChoice.strip() == '5'):
        # TODO: Add Code Here
        break # and Exit the program
    else:
        print(" Only 1,2,3,4 or 5 are valid inputs. Please choose one.")

```

Figure 10: Final source code

```
Menu of Options
1) Show current task list.
2) Add a new task and priority.
3) Remove an existing task.
4) Save Data to ToDoList.txt
5) Exit Program.

Which option would you like to perform? [1 to 5] - 1

Clean Table, High
Build Table, High
```

```
Menu of Options
1) Show current task list.
2) Add a new task and priority.
3) Remove an existing task.
4) Save Data to ToDoList.txt
5) Exit Program.

Which option would you like to perform? [1 to 5] - 2

What is the task: weed garden
What is the priority: Low
Add another task: ('y/n'): y
What is the task: dishes
What is the priority: med
Add another task: ('y/n'): n
```

```
Which option would you like to perform? [1 to 5] - 1

Clean Table, High
Build Table, High
Weed Garden, Low
Dishes, Med
```

```
Which option would you like to perform? [1 to 5] - 3

Task to remove: weed garden
Task not found
Task not found
Task removed from list
Remove another task? ('y/n'): n
```

```
Which option would you like to perform? [1 to 5] - 1

Clean Table, High
Build Table, High
Dishes, Med
```

```
Which option would you like to perform? [1 to 5] - 4

Tasks saved to ToDoList.txt
```

Figure 11: Example of code running in PyCharm



ToDoList.txt - Notepad

File Edit Format View Help

Clean Table, High  
Build Table, High  
Dishes, Med

Figure 12: Text file output

```
Menu of Options
1) Show current task list.
2) Add a new task and priority.
3) Remove an existing task.
4) Save Data to ToDoList.txt
5) Exit Program.

Which option would you like to perform? [1 to 5] - 1

Clean Table, High
Build Table, High
Dishes, Med

Menu of Options
1) Show current task list.
2) Add a new task and priority.
3) Remove an existing task.
4) Save Data to ToDoList.txt
5) Exit Program.

Which option would you like to perform? [1 to 5] - 2

What is the task: water plants
What is the priority: high
Add another task: ('y/n'): n

Menu of Options
1) Show current task list.
2) Add a new task and priority.
3) Remove an existing task.
4) Save Data to ToDoList.txt
5) Exit Program.

Which option would you like to perform? [1 to 5] - 1

Clean Table, High
Build Table, High
Dishes, Med
Water Plants, High

Menu of Options
1) Show current task list.
2) Add a new task and priority.
3) Remove an existing task.
4) Save Data to ToDoList.txt
5) Exit Program.

Which option would you like to perform? [1 to 5] - _
```

Figure 13: Example of running in command prompt

## Summary

In summary, I learned about list objects, dictionaries, coding practices, functions, and GitHub. Then I went through the process of how I added to the To-Do list code to allow the user to view what was in the current list, add, remove from the list, then save to a text file and finally exit the program. I found this assignment a little more difficult than the previous ones but overall I'm happy with my code. I did run into a few issues but those were easily solved with a quick google search. One thing that I would have like to have added was a double check to make sure the user does not quite without saving. Overall, I'm happy with this assignment.