Cameron Kline
August 13, 2020
IT FDN 110B
Assignment 5

Introduction to Python | Module 5

Introduction

Module 5 discusses the following topics. First, the difference between lists and dictionaries are discussed. Following this, we look at the difference between an index and a key. Next, students learn to read data from a file into a list. Then students learn to read data from a file into a dictionary. Going further we discuss why it makes sense to organize data into 2-dimensional structures.

As we expand our knowledge base we look at the programming pattern, "separation of concerns," and then discuss how we might use functions to organize our code. We learn about preparing a script template, and why it might be useful and discuss the usefulness of try-except error handling. Lastly we learn to create repositories on GitHub and about Octocat, the GitHub mascot.

Topic 1 | { } or [ ] - the Difference Between Lists and Dicts.

As we now know, lists are a convenient way to hold collections of objects. Dictionaries are similar to lists, and other sequence types in that they can hold objects, but they differ in that they do not use the index to reference data, but instead use key:value pairs. Both lists and dictionaries are mutable.

Lists use an index to identify an elements position within the list, but dictionaries do not. Elements within a dictionary are all subject to key:value pairing, which is to say that for every key in a dictionary there is a value, even if it hasn't been assigned yet.

To read data from a file into a list we must first open the file with the proper permissions, and then process the data line by line. Before closing the document. It's important to note that most data corruption occurs while files are open and therefore we want to keep it open for as little time as possible. An example of loading data from a file can be seen below.

```python
objFile = open(strFileName, 'r') # Opens the txt file with permission to write
for row in objFile: # Works through each row in the file
    lstRow = row.strip().split(',') # Split each row with a comma
    print(*lstRow) # Print the unpacked row
    lstTbl.append(lstRow) # Append the list of list with the information assigned to lstRow from file
objFile.close() # Closes the file
print(lstTbl)
```

Figure 1, loading data from a file into a list.

Reading data into a file from a dictionary is done quite similarly. The main difference is that we need to assign the data being read into the script a key so that it can become a key:value pair.

```python
objFile = open(strFileName, 'r')
for row in objFile:
    lstRow = row.strip().split(',')
    dictRow = {'id': int(lstRow[0]), 'name': lstRow[1], 'email': lstRow[2]}
    lstTbl.append(dictRow)
objFile.close()
```

Figure 2, loading data from file into dictionary

It's interesting to note that in the example above we can see that data from a list is being loaded into a dictionary, and that the list data is being assigned to a key based on its index position. For example, lstRow position 0, is being assigned the key of ‘id’ as coded: ‘id’: int(lstRow[0]). The structure and flexibility of the 2-D data structure make it appealing to use.

Topic 2 | Separation of Concerns

The term separation of concerns refers to a design principle for computing programming which separates a program into distinct sections. We introduced separation of concerns into our Spyder template file in module 5 which allows us to follow the same structured path with all future programs. Users could also opt to do this with Pseudocode if they choose.

Functions were discussed briefly in module 5 and are a good way to further section off code. Statements can be held in a function as declarations and then called with the function name.

Updating our Spyder template provided us with the opportunity to start with this best practices method of breaking up our programs early in our programming career. The thought being that learning this early on would make it a good life-long habit.

Topic 3 | Error Handling

My own personal script was so problematic for assignment 5 that I didn’t get to take an in-depth look at error handling, but try-except error handling is one way we can keep python from crashing when an error is encountered. Believe me when I say I need to know all of the ways to keep Python from crashing at this point, so this is a good tool to have in my toolbox.

The try-except method for error handling asks Python to first ‘try’ an operation, and if it cannot perform that operation to perform the except statement instead of crashing. It looks like the example below in figure 3.

```
def divide():
    intDiv='error'

    intA = int(input('This is A:'))
    intB = int(input('This is b:'))
    try:
        intDiv = intA /intB
    except:
        print('No can do')
    return intDiv
print(divide())
```

Figure 3, try-error handling

Topic 4 | GitHub and the Octocat

GitHub is a version control system which offers us the flexibility to turn back the clock, so to speak. Versions of code are stored on GitHub to allow for the tracking of changes, something I wish I had done when starting assignment 5.

In addition to allowing the tracking of changes, GitHub allows for teams and individuals to collaborate on projects and track the changes that are bing made to a project.

The mascot for GitHub is the octocat[1], meow. Or blub, blub. Whatever noise an octocat makes.

Topic 5 | Lab05-A

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
#-------------------------------------------#
# Author: cameronkline
# Description: LAB -5-A
# Created on Thu Aug  6 20:33:24 2020
# Change Log: CKLINE, 8/6/2020, File Created
#-------------------------------------------#

# Declare Variables

```

[1] https://medium.com/@smhatre59/the-untold-story-of-github-132840f72f56 8/13/20

```python
strChoice = '' # User Input
lstTbl = [] # List of lists to hold data
lstRow = [] # List of row data
strFileName = 'CDInventory.txt' # Date storage file
objFile = None # file object

# Get User Input
print('Write or Read file data.')
while True:
    # This line is printing the menu
    print('\n[a] add data to list\n[w] to write data to file\n[r] ro read data from file')
    # This line prints the last two menu items
    print('[d] display data\n[exit] to quit')
    #This accepts the user input
    strChoice = input('a,w,r,d or exit: ').lower() # Convert choice to lower case at time of input
    print('\n\n')

    # If the user types exit the script ends.
    if strChoice == 'exit':
        break
    if strChoice == 'a': # No elif necessary, as this code is only reached if strChoice is not 'exit'
    # Add data to list in memory
    # TODO ask user to input and store it in the in-memory list

        pass
    elif strChoice == 'w':
        # List to File
        # TODO add code here to write from in-memory list to file
        pass
    elif strChoice == 'r':
        # File to print
        # TODO read the file line by line into in-memory list
            # This should go into lstRow or lstTbl?
            # print(*lstRow, sep = ', ')
        pass
    elif strChoice == 'd':
        # Display data
        # TODO display the data to the user.
        pass
    else:
        print('Please choose a, w, r or exit!')
```

Figure 4, Lab05-a

Topic 6 | Lab05-B

```
#------------------------------------------#
# Title: Lab05_B.py
# Desc: Lab05-B starter script
# Change Log: (CKline, 8/10/20, Attempted to complete lab)
# DBiesinger, 2030-Jan-01, Created File
#------------------------------------------#

# Declare variables

strChoice = '' # User input
lstTbl = []  # list of lists to hold data
dicRow = {}  # dictionary containing artist and album
strFileName = 'CDInventory.txt'  # data storage file
objFile = None  # file object

# Get user Input
print('Write or Read file data.')
while True:
    print('\n[a] add data to list\n[w] to write data to file\n[r] to read data from file')
    print('[d] display data\n[exit] to quit')
    strChoice = input('a, w, r, d, or exit: ').lower()  # convert choice to lower case at time of input
    print()

    if strChoice == 'exit':
        break
    if strChoice == 'a':  # no elif necessary, as this code is only reached if strChoice is not 'exit'
        # Add data to list in memory
        strArtist = input('Enter artist name: ')
        strAlbum= input('Enter album name: ')
        # TODO ask user to input data and store it in the in-memory list
        dicRow = {'artist': strArtist, 'album': strAlbum}
        lstTbl.append(dicRow)
        # pass
    elif strChoice == 'w':
        # List to File
        # TODO add code here to write from in-memory list to file
        objFile = open(strFileName, 'w')
        for row in lstTbl:
            strRow = ''
            for item in row.values():
                strRow += str(item) + ', '
            strRow = strRow[:-1] + '\n'
            objFile.write(strRow)
        objFile.close()
    elif strChoice == 'r':
        # TODO read the file line by line into in-memory list.
        lstTbl.clear()
        objFile = open(strFileName, 'r')
        for row in objFile:
            lstRow = row.strip().split(',')
            dicRow = {'artist': lstRow[0], 'title': lstRow[1]}
            lstTbl.append(dicRow)
        objFile.close()
    elif strChoice == 'd':
        print('Artist,  Album')
        print(dicRow)
        for row in lstTbl:
            print(*row.values(), sep = ', ')

    else:
        print('Please choose either a, w, r or exit!')
```

Figure 5, Lab05-B

Topic 7 | Assignment 05

I'll resist writing a tome about assignment 05, and just say that I found it very difficult, even with the starter script. When I step back from the script I feel like the materials make sense and as soon as I have to implement the skills that I've learned I'm overwhelmed with choices and stall out.

Even though I haven't been able to fully complete the script per the assignment I have still learned a lot, and I found a tool which helped me to visualize what was happening in my code. When searching for an actual Python tutor I found this site, http://www.pythontutor.com[2], which shows the execution of code in a visual manner which for me was helpful. In particular, it was helpful to see the structure when a list became a list of dictionaries.

Topic 8 | Summary

In module 5 discusses we discussed the difference between lists and dictionaries. Of particular importance is the usage of keys instead of indices. We learned to read data from a file into a list, and the importance of 2D data structures.

We look at the programming pattern, "separation of concerns," and discussed how we might use functions to organize our code. In addition we prepared a script template, and discussed the usefulness of try-except error handling. Lastly we learned to create repositories on GitHub and about Octocat, the GitHub mascot.

[2] http://www.pythontutor.com 8/13/20