Name: Gail Provancha

Date: August 22, 2023

Course: IT FDN 110 A Foundations of Programming: Python

Assignment: Assignment07

GitHub: https://github.com/provgl1/IntroToProg-Python-Mod07

Pickling and Exception Handling

Introduction

This document will go over the steps needed to write a Python Script that provides a choice for the user to add to a binary file or exit the program. This script builds upon prior lessons, introducing Pickling and Exception handling.

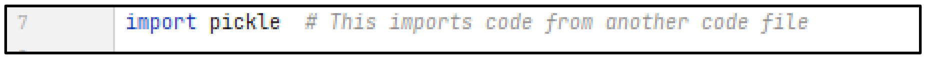
Sections of the Script

I simplified sections of the script from Assignment06 that involved a user making a selection from a menu. The script is divided into different sections including "Data"- declaring of variables, "Processing" – where functions are defined, and "Presentation" – where the menu is displayed and the user makes inputs.

Pickling

Pickling means to serialize an object or to be able to save complex data in a single line of code that has all the necessary information which can then be unpickled and used in other Python scripts.

First, I imported the pickle module which is include with Python (Figure 1).



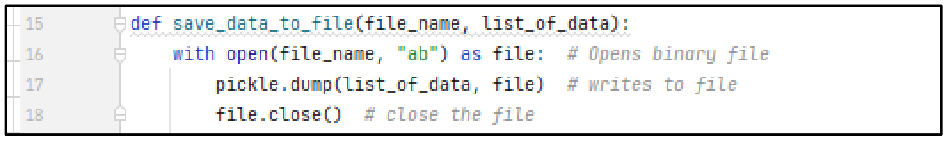
```
import pickle  # This imports code from another code file
```

Figure 1: import pickle

Function: save data to a file

Next, I created a function to save to a binary file. The function "open()" is used with the name of the file and mode "ab". The mode "ab" means to append to a binary file, and if the file does not exist, it will be created.

The function "pickle.dump()" is then used to write a data list to the binary file. And then the file is closed, "close()" and saved.

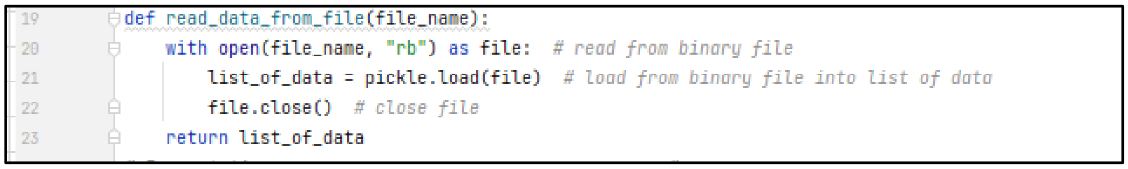


```
def save_data_to_file(file_name, list_of_data):
    with open(file_name, "ab") as file:  # Opens binary file
        pickle.dump(list_of_data, file)  # writes to file
        file.close()  # close the file
```

Figure 2: opening file and saving as binary

Function: read data from file

Then, I created a function to read data from a binary file. The function "open()" is once again used, but this time with the mode "rb" which means to read from a binary file. The function "pickle.load()" is then used to add one entry to a list. And then the file is closed, "close()" (Figure 3).



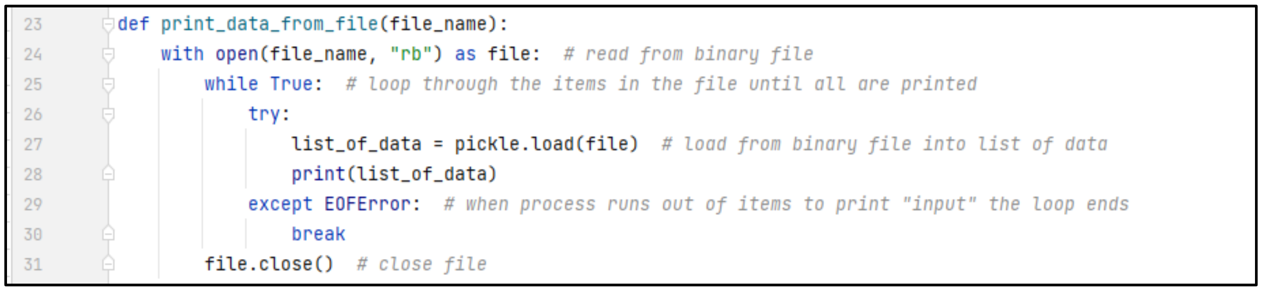
```
def read_data_from_file(file_name):
    with open(file_name, "rb") as file:  # read from binary file
        list_of_data = pickle.load(file)  # load from binary file into list of data
        file.close()  # close file
    return list_of_data
```

__Figure 3: reading data from a binary file and saving to a list__

Exception Handling

Scripts don't always run smoothly and various errors can occur. To make a script more robust, exception handling can be added.

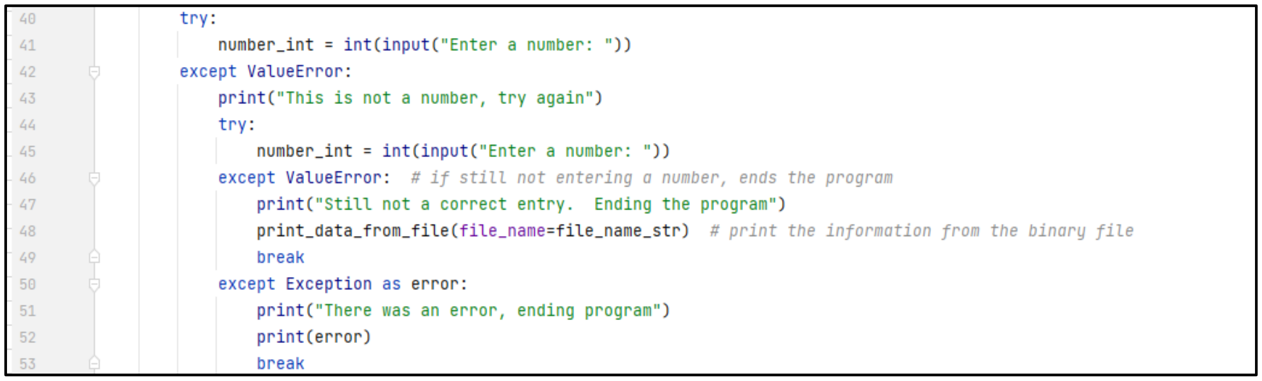
One method is the Try/Except. For my script, I decided to use this method to accomplish something a bit different, not to capture a possible error, but to gain the ability to add more then 1 entry in my list when reading (unpickling) my binary file. I decided to use a While loop and "try" reading the lines from the binary file until no further lines exist. Once that occurred, the script would jump to the "except" and the loop would end (Figure 4).



```
def print_data_from_file(file_name):
    with open(file_name, "rb") as file:  # read from binary file
        while True:  # loop through the items in the file until all are printed
            try:
                list_of_data = pickle.load(file)  # load from binary file into list of data
                print(list_of_data)
            except EOFError:  # when process runs out of items to print "input" the loop ends
                break
        file.close()  # close file
```

__Figure 4: Updated function to include While and Try/Except.__

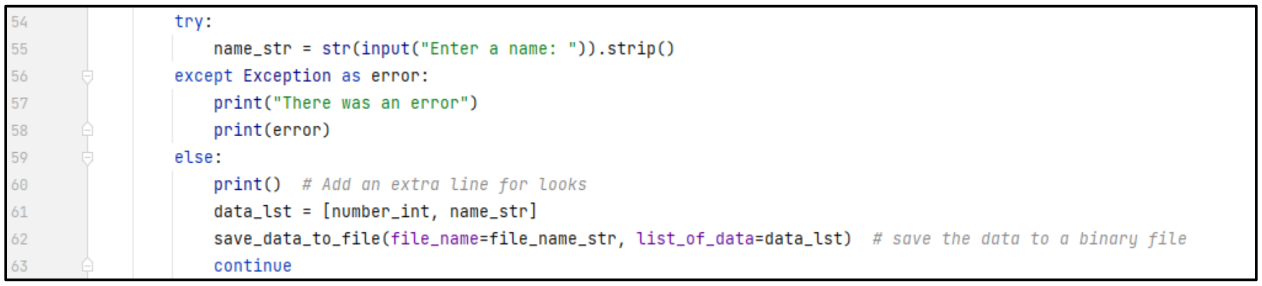
The next section I added Try/Except was with user inputs. If the user inputted something besides an integer, they were provided a message "This is not a number, try again." If they continued to enter an incorrect value, the message changed to "Still not a correct entry. Ending the program". This was to avoid the program continuously running when the user was not able to provide an acceptable value (Figure 5).



```
        try:
            number_int = int(input("Enter a number: "))
        except ValueError:
            print("This is not a number, try again")
            try:
                number_int = int(input("Enter a number: "))
            except ValueError:  # if still not entering a number, ends the program
                print("Still not a correct entry.  Ending the program")
                print_data_from_file(file_name=file_name_str)  # print the information from the binary file
                break
            except Exception as error:
                print("There was an error, ending program")
                print(error)
                break
```

__Figure 5: Try/Except, message to user__

Then I added Try/Except for the string input. This time, the message was more generic and I added the Else to run a block of code if no exception is found (Figure 6)

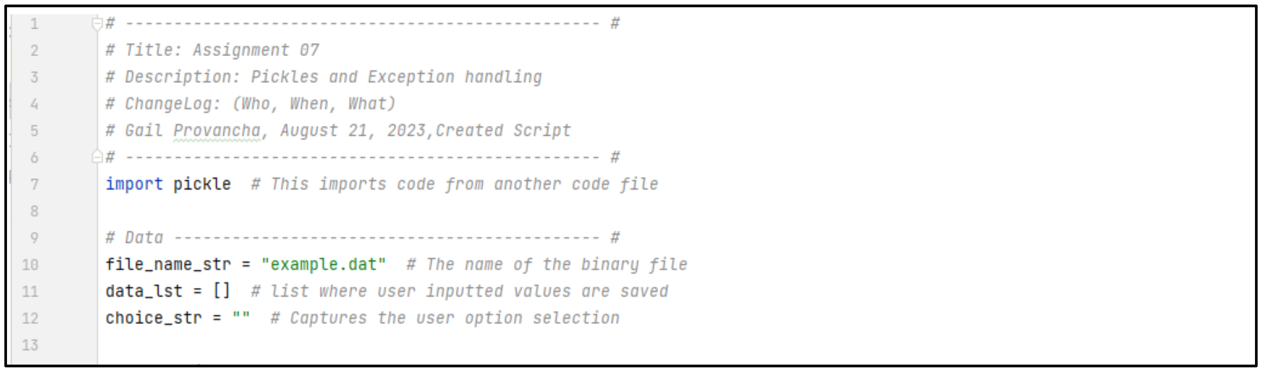


```python
        try:
            name_str = str(input("Enter a name: ")).strip()
        except Exception as error:
            print("There was an error")
            print(error)
        else:
            print()  # Add an extra line for looks
            data_lst = [number_int, name_str]
            save_data_to_file(file_name=file_name_str, list_of_data=data_lst)  # save the data to a binary file
            continue
```

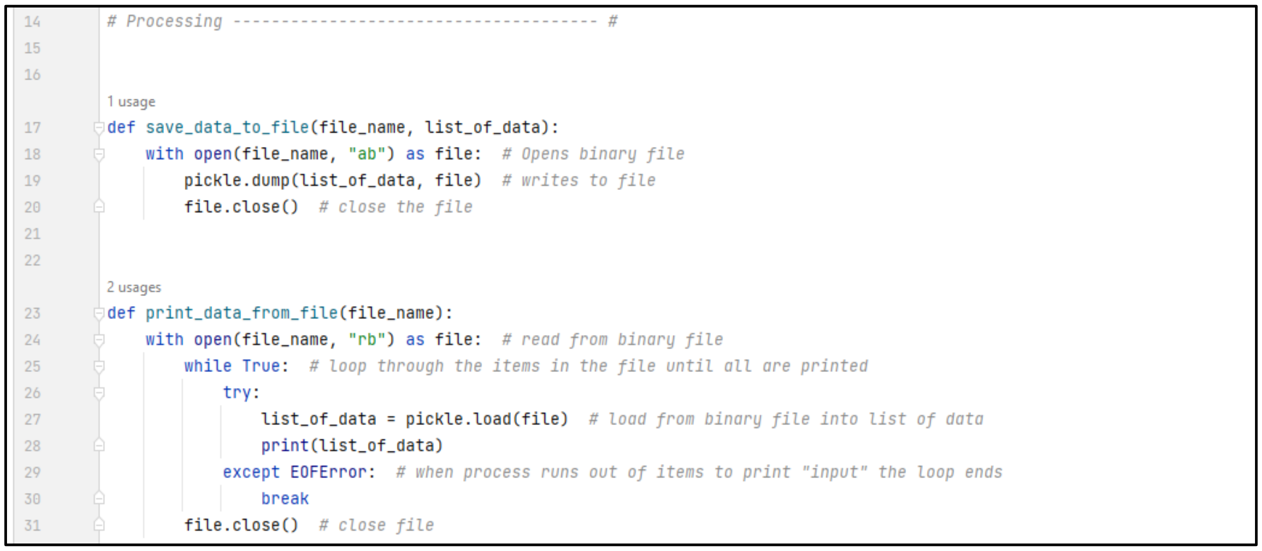
Figure 6: Try/Except/Else, generic message to user

Completed Script

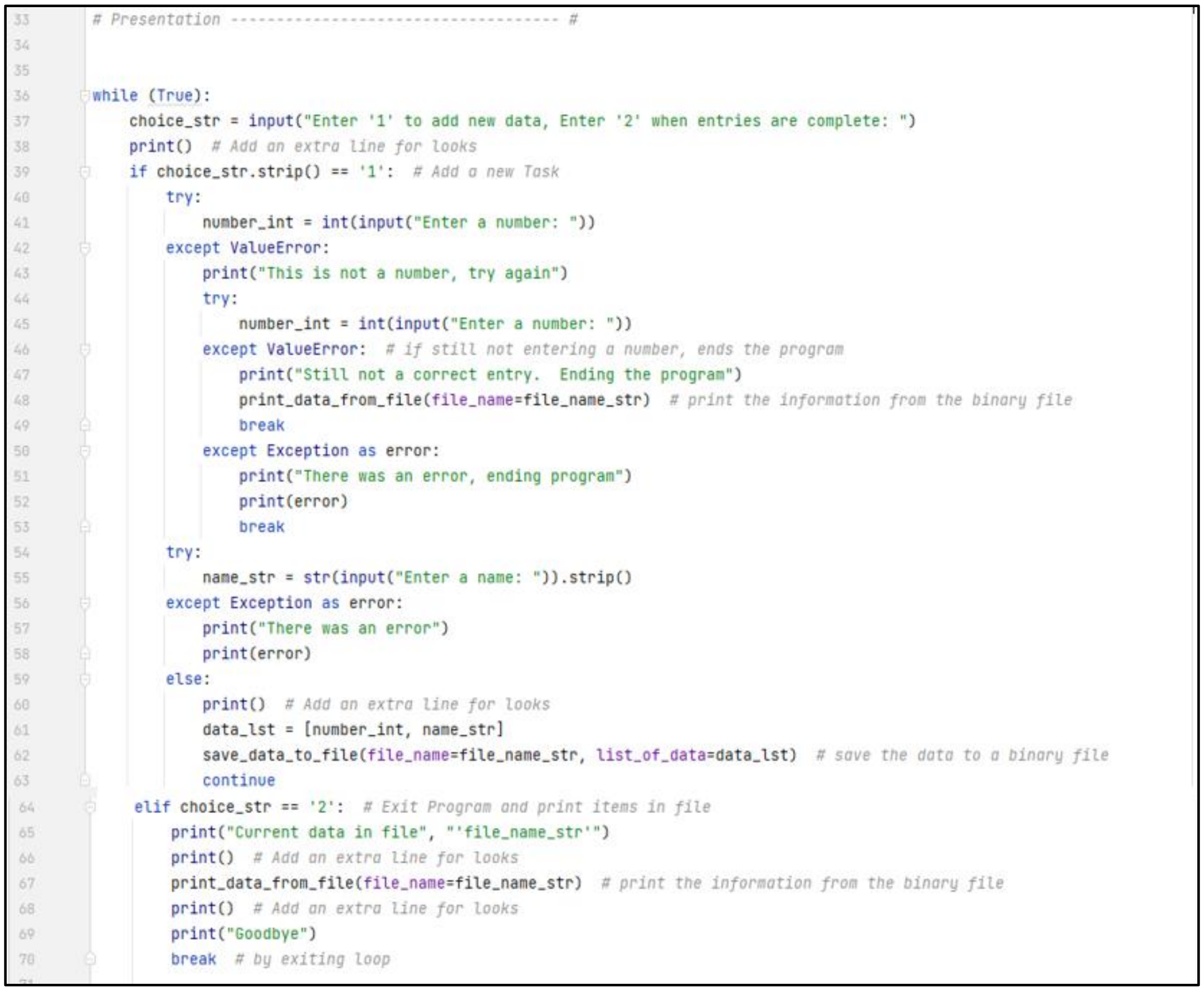
See (Figure 7) for the completed script.



```python
# -------------------------------------------------- #
# Title: Assignment 07
# Description: Pickles and Exception handling
# ChangeLog: (Who, When, What)
# Gail Provancha, August 21, 2023,Created Script
# -------------------------------------------------- #
import pickle  # This imports code from another code file

# Data -------------------------------------------- #
file_name_str = "example.dat"  # The name of the binary file
data_lst = []  # list where user inputted values are saved
choice_str = ""  # Captures the user option selection

```



```python
# Processing ------------------------------------- #


1 usage
def save_data_to_file(file_name, list_of_data):
    with open(file_name, "ab") as file:  # Opens binary file
        pickle.dump(list_of_data, file)  # writes to file
        file.close()  # close the file


2 usages
def print_data_from_file(file_name):
    with open(file_name, "rb") as file:  # read from binary file
        while True:  # loop through the items in the file until all are printed
            try:
                list_of_data = pickle.load(file)  # load from binary file into list of data
                print(list_of_data)
            except EOFError:  # when process runs out of items to print "input" the loop ends
                break
        file.close()  # close file
```



```
# Presentation ----------------------------------- #


while (True):
    choice_str = input("Enter '1' to add new data, Enter '2' when entries are complete: ")
    print()  # Add an extra line for looks
    if choice_str.strip() == '1':  # Add a new Task
        try:
            number_int = int(input("Enter a number: "))
        except ValueError:
            print("This is not a number, try again")
            try:
                number_int = int(input("Enter a number: "))
            except ValueError:  # if still not entering a number, ends the program
                print("Still not a correct entry.  Ending the program")
                print_data_from_file(file_name=file_name_str)  # print the information from the binary file
                break
            except Exception as error:
                print("There was an error, ending program")
                print(error)
                break
        try:
            name_str = str(input("Enter a name: ")).strip()
        except Exception as error:
            print("There was an error")
            print(error)
        else:
            print()  # Add an extra line for looks
            data_lst = [number_int, name_str]
            save_data_to_file(file_name=file_name_str, list_of_data=data_lst)  # save the data to a binary file
            continue
    elif choice_str == '2':  # Exit Program and print items in file
        print("Current data in file", "'file_name_str'")
        print()  # Add an extra line for looks
        print_data_from_file(file_name=file_name_str)  # print the information from the binary file
        print()  # Add an extra line for looks
        print("Goodbye")
        break  # by exiting loop
```

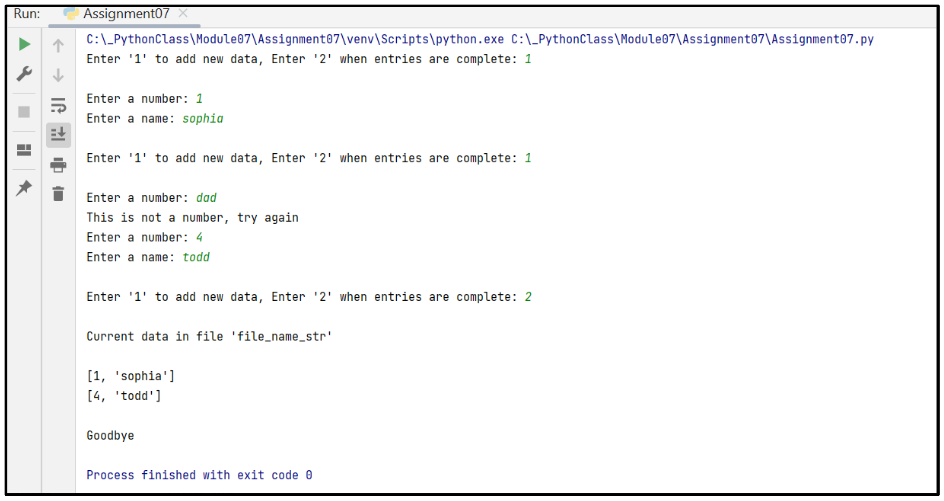
Figure 7: Completed Script

Testing the Script

We will be testing the script in both PyCharm and Command Prompt.

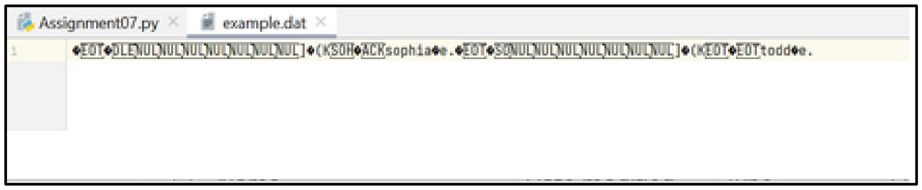
<u>PyCharm</u>

First, let’s test the script by running it in PyCharm (Figure 8).



```
C:\_PythonClass\Module07\Assignment07\venv\Scripts\python.exe C:\_PythonClass\Module07\Assignment07\Assignment07.py
Enter '1' to add new data, Enter '2' when entries are complete: 1

Enter a number: 1
Enter a name: sophia

Enter '1' to add new data, Enter '2' when entries are complete: 1

Enter a number: dad
This is not a number, try again
Enter a number: 4
Enter a name: todd

Enter '1' to add new data, Enter '2' when entries are complete: 2

Current data in file 'file_name_str'

[1, 'sophia']
[4, 'todd']

Goodbye

Process finished with exit code 0
```

Figure 8a – running in Python

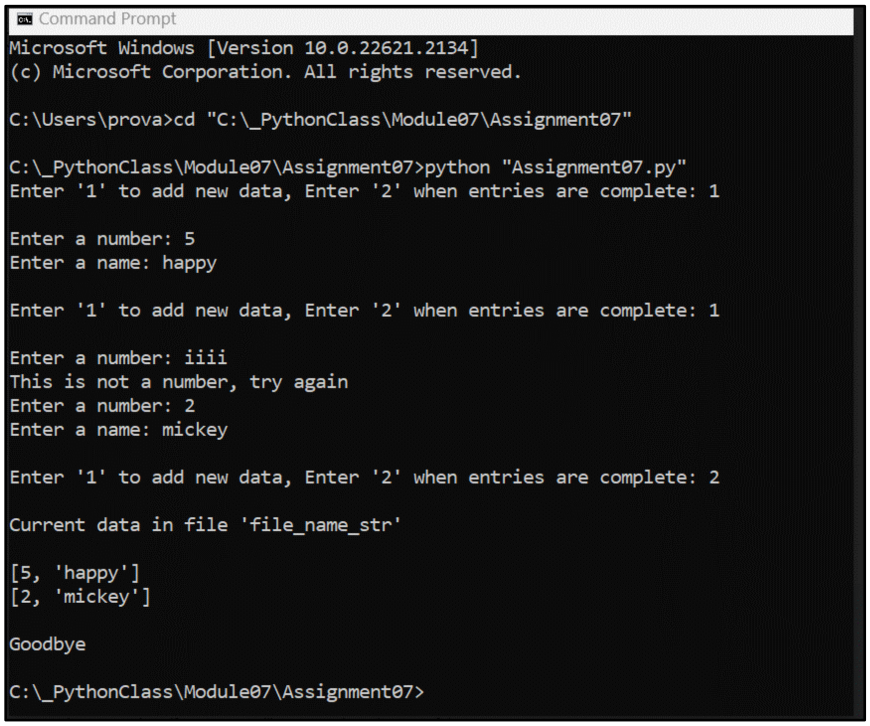


```
�EOT�DLENULNULNULNULNULNULNUL]�(KSOH�ACKsophia��.�EOT�SONULNULNULNULNULNULNUL]�(KEOT�EOTtodd��.
```

Figure 8b: binary file

CMD line

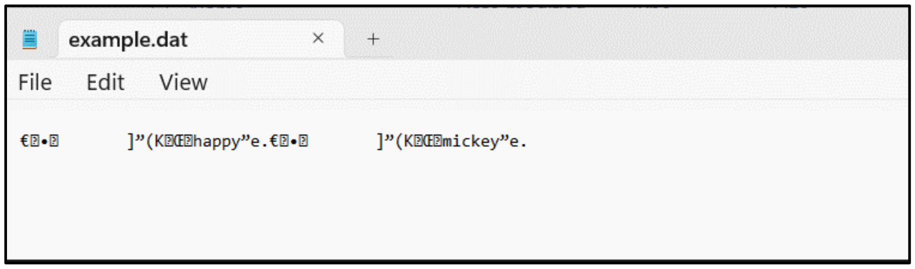
Now the script will be run from Command Prompt to make sure it also runs as expected. Open a command prompt window "CMD". Type in "cd" and then the pathway to the directory where the python script is saved "C:_PythonClass\Module07\Assignment07". Click Enter. Next type "python" followed by the name of the python script "Assignment07.py". Click "Enter" again to see the script run. Provide the necessary inputs as if you are the user and verify the result is what you expected (Figure 9).



```
Microsoft Windows [Version 10.0.22621.2134]
(c) Microsoft Corporation. All rights reserved.

C:\Users\prova>cd "C:\_PythonClass\Module07\Assignment07"

C:\_PythonClass\Module07\Assignment07>python "Assignment07.py"
Enter '1' to add new data, Enter '2' when entries are complete: 1

Enter a number: 5
Enter a name: happy

Enter '1' to add new data, Enter '2' when entries are complete: 1

Enter a number: iiii
This is not a number, try again
Enter a number: 2
Enter a name: mickey

Enter '1' to add new data, Enter '2' when entries are complete: 2

Current data in file 'file_name_str'

[5, 'happy']
[2, 'mickey']

Goodbye

C:\_PythonClass\Module07\Assignment07>
```

Figure 9: running script in Command Line

Open up the “example.dat” file to verify the results have changed based on the new inputs are what are expected (Figure 10).



```
€•      ]”(KŒhappy”e.€•      ]”(KŒmickey”e.
```

Figure 10: Testing in Command Line, results

Summary

In summary, this document goes over the steps needed to write a Python script that provides a menu that allows the user to add to a binary file or read from a binary file. The script introduced the use of pickling and exception handling.