**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).



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.

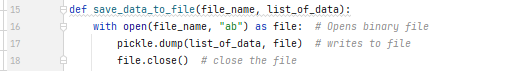


Figure 2: opening file and saving as binary

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).

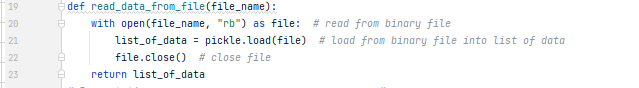


Figure 3: reading data from a binary file and saving to a list

# 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).

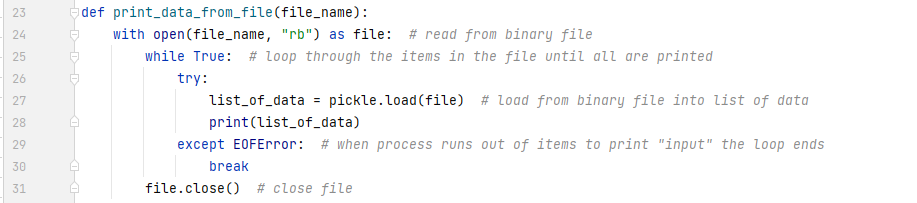


Figure 4: Updated function to include While and Try/Except.

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).

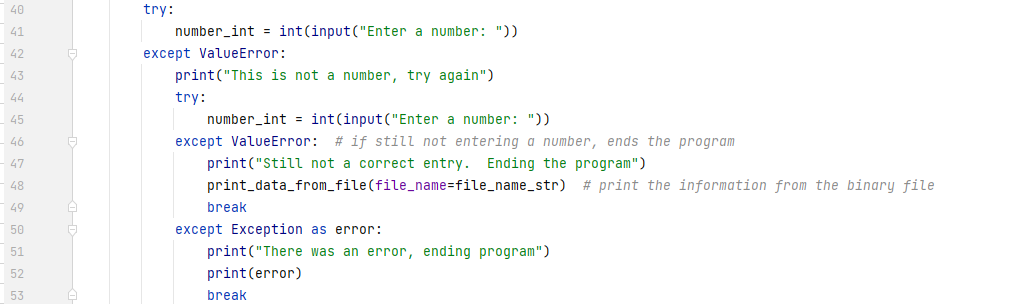


Figure 5: Try/Except, message to user

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)

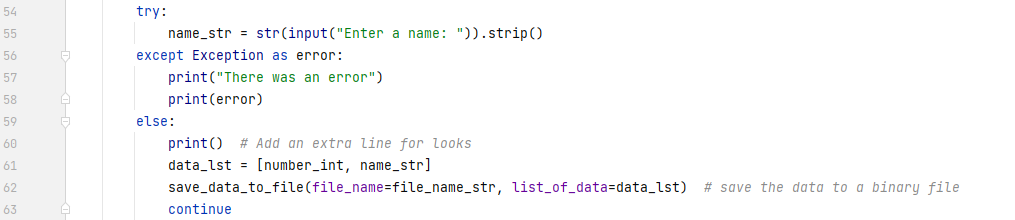


Figure 6: Try/Except/Else, generic message to user

# Completed Script

See (Figure 7) for the completed script.

# 



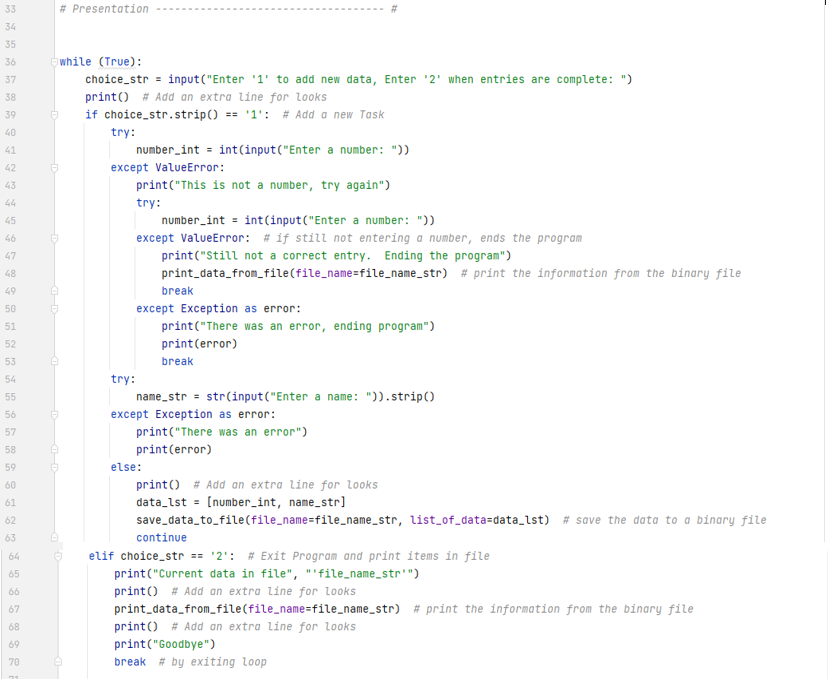


Figure 7: Completed Script

# Testing the Script

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

PyCharm

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

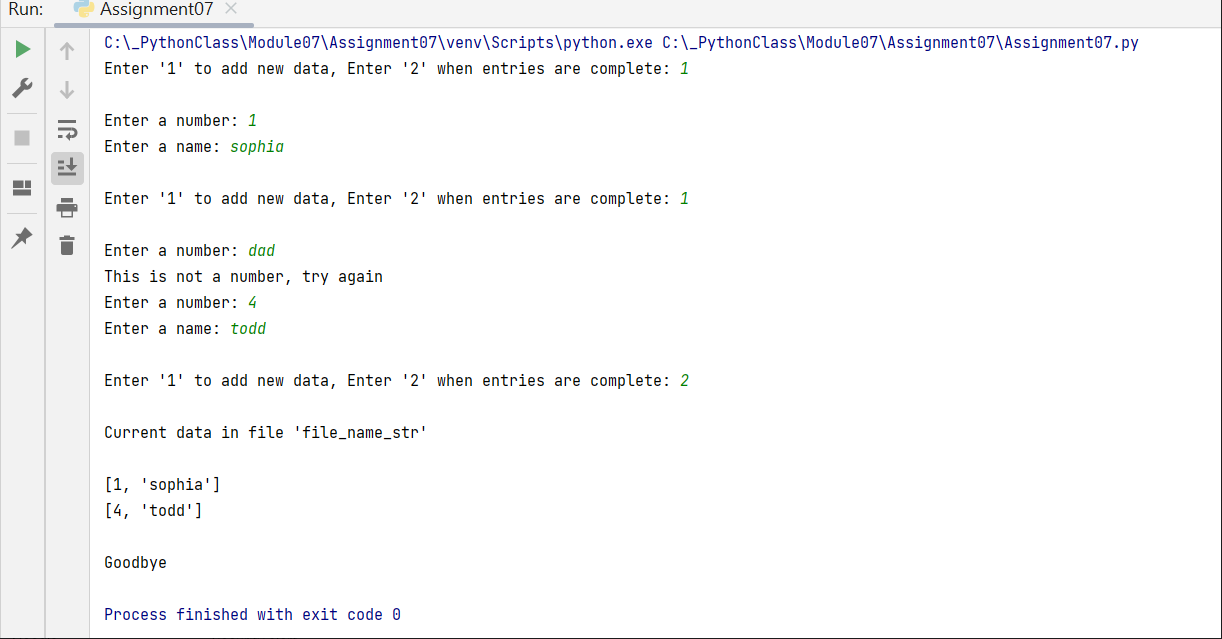


Figure 8a – running in Python



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).

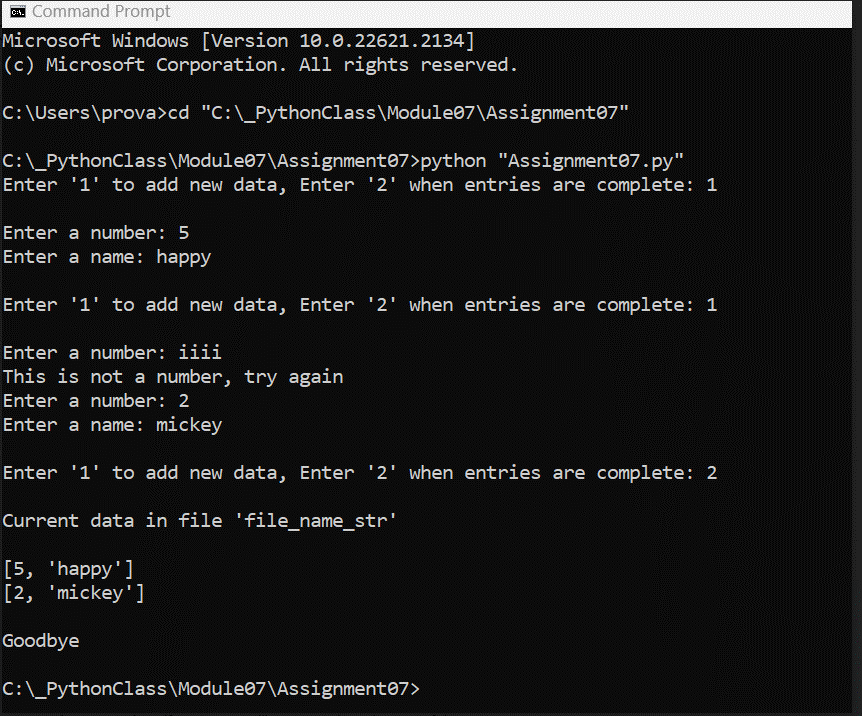


Figure 9: running script in Command Line

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

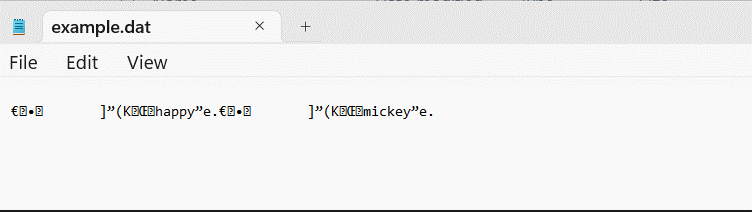


Figure 10: Testing in Command Line, results

# 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.