Matthew Tsao

November 29, 2022

Foundations Of Programming: Python

Assignment07

GitHub URL: <https://github.com/mrtsao1/IntroToProg-Python-Mod07>

GibHub Webpage: <https://github.com/mrtsao1/IntroToProg-Python-Mod07/blob/main/docs/index.md>

Create Script using Pickling and Error Handling

# **Introduction**

This week, I learned about pickling and error handling. I created a script to demonstrate these topics. The script consists of two parts: ***Example #1 – Pickling*** and ***Example #2 – Error Handling and Raising Custom Errors***. The following information is a breakdown on how I wrote this program step-by-step.

# **Creating and Running the Program**

Refer to the following subsections below.

## Script Header

Create the script header. A script header should contain the title of the script file, a description, and a change log. A script header is created using comments, which consists of either starting the line with “#” (single line comment) or multi-line comments (Figure 1).

Text

Description automatically generated

Figure 1. Script Header

## Import Pickle Module and Initialize Variables

Import pickle module and create variables with initialized values that will be used throughout the program (Figure 2). The key variables and their descriptions are shown in figure 2 below. The pickle module will be used to work with binary files.

Text

Description automatically generated

Figure 2. Import Pickle Module and Initialize Variables

## Functions

The script contains the following functions:

* save\_data\_to\_file (Figure 3) – Saves data to binary file using “dump” function from pickle module; used in Example #1
* read\_data\_from\_file (Figure 3) – Reads data from binary file using “load” function from pickle module; used in Example #1
* calculate\_quotient (Figure 4) – Calculate quotient, save to .txt file; used in Example #2
* output\_menu\_tasks (Figure 5) – Display a menu of choices to user; used in Example #2
* input\_menu\_choice (Figure 5) – Gets the menu choice from a user; used in Example #2

Text

Description automatically generated

Figure 3. Save and read data from file functions

Text

Description automatically generated

Figure 4. calculate quotient function

Text

Description automatically generated

Figure 5. display menu and get user menu choice functions

## Main Body of Script

### Example #1: Pickling

The first part of the script (Figure 6) demonstrates pickling. The user will be prompted to enter an ID number and name. The program will store this information in a list and call the save\_data\_to\_file function to save the data to the file “AppData.dat”. The user will need to press enter to continue (line 99). Then, the read\_data\_from\_file function is called to display the contents of “AppData.dat” in order to verify the data written to the file was correct.

Text

Description automatically generated

Figure 6. Example #1 - Pickling

### Example #2: Error Handling and Raising Custom Errors

The second part of the script (Figure 7) demonstrates error handling and custom errors. A while loop with a menu is used in order to demonstrate the various error handling situations that could be encountered. The script calls the output\_menu\_tasks and input\_menu\_choice functions to display the menu choices to the user and have the user make a selection. The menu options are: 1 – calculate quotient and save data to file, 2 – read data from file, and 3 – exit program. If the user inputs anything else, they will receive an invalid choice message and be prompted to choose again. Option 1 calls the calculate\_quotient function and option 2 calls the read\_data\_from\_file function. The “Verify Program Worked” section will go into the various error handling situations that the script implements.

A screenshot of a computer

Description automatically generated with medium confidence

Figure 7. Example #2 – Error Handling and Raising Custom Errors

## Verify Program Worked

The sample files “AppData.dat” and “test.txt” are in my working directory (Figure 8). I used these files to test my program in PyCharm and OS command/shell window.

Graphical user interface, text, application

Description automatically generated

Figure 8. Working Directory Files

### Run Program Using PyCharm

#### Example #1: Pickling

User enters an ID and name which are saved to “AppData.dat” (Figure 9). The user presses enter when ready to continue and the script reads the data that was saved to the file. The user presses enter when ready to continue again.

Text

Description automatically generated

Figure 9. Example #1 - Pickling

##### Verify Contents of Output File

Check to verify that the output file was created with data as expected (Figure 10).

Graphical user interface, text, application

Description automatically generated

Figure 10. Verification of Output File

#### Example #2: Error Handling and Raising Custom Errors

The user is presented with a small menu and prompted to make a selection (Figure 11).

Text

Description automatically generated

Figure 11. Example #2 - Error Handling and Raising Custom Errors

##### Error Handling – Division by Zero

The user selected option 1 (Figure 12). The user is prompted to input a value for numerator and denominator. The user entered zero for the denominator, evoking the division by zero handling error.

Text

Description automatically generated

Figure 12. Error Handling - Division by Zero

##### Error Handling – File Not Found

The user selected option 1 (Figure 13). The user is prompted to input a value for numerator and denominator. Then, the user is asked to enter a filename to save the entered data to. For simplicity, the script will evoke a file not found error if the entered filename is not in the working directory.

Text

Description automatically generated

Figure 13. Error Handling – File Not Found

##### Error Handling – Non-Specific Errors

The user selected option 1 (Figure 14). The user is prompted to input a value for the numerator. The user did NOT enter an integer value, evoking the non-specific error handling error.

Text

Description automatically generated

Figure 14. Error Handling – Non-Specific Error

##### Raised Custom Error – Numeric Filename Search

The user selected option 1 (Figure 15). The user is prompted to input a value for numerator and denominator. Then, the user is asked to enter a filename to save the entered data to. For simplicity, the script will evoke a file not found error if the entered filename is not in the working directory. Since the user tried to enter a number for the filename, the raised custom error that numbers can NOT be used for the filename was evoked. The script also evoked the non-specific error handling error.

Text

Description automatically generated

Figure 15. Raised Custom Error - Numeric Filename Search

### Run Program Using OS command/shell window

Snip of Assignment07.py running using shell window (Figure 16).

Text

Description automatically generated

Figure 16. Run Assignment07.py in shell window

# **Summary**

The Assignment07.py program demonstrated the following concepts:

* Exception handling
  + Division by zero
  + File doesn’t exist
  + Non-specific errors
* Pickling/working with binary files
* Raising custom errors
* Creating advanced GitHub pages