Lauren Parker

November 29, 2020

IT FDN 110 A Au 20: Foundations Of Programming: Python

Assignment 07

Exception Handling & Pickling

# Introduction

This week’s assignment was to create a script that showcased both exception handling and pickling. For my assignment, I chose to create a script that asked its user for their name and birth year. The program would save the user’s input to a list and ask the user if they would like to save their information to a binary file. The user can then choose yes or no. Finally, the program asks its user if they would like to read their data from the binary file. Once the user answers, the program ends. You will see that exception handling and pickling were both used in order to make this code run properly.

# Step 1: Create a New File and Adding Headers

The first thing I did was create a new file in PyCharm and added a header with the title of my program, description, and change log. I then added some preliminary headers to keep my separation of concerns. These are denoted with the pound symbol. You can see an example of this is figure 1.

Graphical user interface, text

Description automatically generated

# Step 2: Pickle Module & Defining Variables

Pickling is a way of converting an in-memory object into a character string. Knowing that I was going to use pickling in my program, the first thing I did was call the pickle module as you can see in figure 2. This is mandatory if you are planning on using pickling in your code. Next, I added and defined my global variables that I would be using. You can see an example of this is figure 2.

Text

Description automatically generated

***Figure 2: Calling the pickle method and defining global variables.***

# Step 3: Creating Functions

I knew that I would need to create at least two separate functions for my program. One that would create and append the user’s data to a .dat file and another function that would open the .dat file and read the data back to the user. Both of these functions would use pickling to save the data to a binary file and then read the data back. Both functions are listed under the processing section of my code. You can see an example of them in figure 3.

***Text

Description automatically generated***

***Figure 3: Defining functions.***

# Step 4: Input/Output and Error Handling

The next portion of the code was listed under the presentation (I/O) section. This code would retrieve the users name and birth year, then add them to the list, lst\_data. As the input function normally returns a string, I didn’t throw an exception. However, for the second input, I knew I wanted the user to input an integer and there was a possibility the user could enter something else. Therefore, I used a try-except block. The try-except block is a helpful way to troubleshoot possible errors or bugs. You can tell the program what to do if a specific error arises which can be helpful to anyone using or working with your code. I first have the program try an input statement. If the user returns something that is not an integer, the program will either return the statement “Error: Please enter a number for your birth year!” or “Something went wrong”. It will then ask the user to re-input their birth year. You can see an example of this in figure 4.

Text

Description automatically generated

***Figure 4: Example of a try-except block.***

# Step 5: Calling Functions in Main Body of Script

The next portion of code would be used to call the previously defined functions in order to add the user’s input to a binary file. I used an if-elif block to ask the user if they wanted to save their data to a file. If the user input ‘y’ for yes, the program would call the function save\_data\_to\_file(). If the user chose ‘n’ the program would continue on to the next if-elif block. The user would then be asked if they would like to load their data from the file. If the user answered ‘y’, the program would then call the function read\_data\_fm\_file(). You can see an example of this in figure 5.

Text

Description automatically generated

***Figure 5: Example of if-elif blocks using functions.***

# Summary

In conclusion, we have discussed how to call the pickle module, in order to convert in-memory objects into character strings using binary files. We’ve covered error handling and how it can be beneficial when troubleshooting your code. Lastly, we’ve seen how both of these methods can be incorporated into a Python script to enhance your coding.