Josh Embury
12/20/2019
Foundations of Programming: Python
Assignment07

Working with Pickling and Error Handling

Introduction

Before diving into the scripting, it was important to review the resources recommended by Prof. Root. This included his Module 7 material video and class notes. Additionally, he provided links to external webpages and YouTube videos detailing the same information but presented in a different way. This variety of material was useful in understanding the use of error handling within a script and how to use pickling.

The goal of this assignment is to provide a demonstration of how pickling works and how to implement error handling within a script. The following demonstration is an example which incorporates the use of pickling and error handling. It can be used as a starting point for someone wanting to create a budgeting tool when determining how much to spend for people on a holiday shopping list.

Try-Except

The first portion of the code is shown if Figure 1 and is the try block.

```
shopping_list = []
FileName = "AppData.dat"

# Get info from user and save to list. Display input.
print("How much money can you spend on Christmas gifts?")
try:
    # Get info from user
    name_str = str(input("Enter a name on your shopping list: "))
    budget_int = int(input("Enter a budget for this person($): "))

    # Save to list and print back to user
    shopping_list = [name_str, budget_int]
    print(shopping_list)

    # Store Data to binary file, begin pickling
    objFile = open(FileName, "ab")
    pickle.dump(shopping_list, objFile)
    objFile.close()

    # Read data from binary file
    objFile = open(FileName, "rb")
    objFileData = pickle.load(objFile)
    objFile.close()
    print(objFileData)
```

Figure 1: Try block

When using structured error handling, a try-except block is used. When an error is thrown in the code contained within the try section, exceptions can be "caught" in the except section. It is up to the programmer to anticipate what types of errors the user may introduce when running the program and to implement the try-except block accordingly. By having this foresight, the program will be more useable by the customer of the program by showing them what has gone wrong in the program by printing custom, coherent error messages, designed by the programmer, instead of the Python built-in messages which are typically seen when an error occurs.

In the example shown in Figure 1, there is one error in particular which may be introduced by the user. If the user decides to be cheeky when entering the dollar value for the budget input, an error will occur. That is, if the user enters "fifty" instead of the integer 50, a ValueError exception will be thrown. This can be designed into the code with an exception block, which is shown in Figure 2.

```
except ValueError as e:
    print("Value entered for budget must be an integer!")
    print("Built-In Python error info: ")
    print(e, e.__doc__, type(e), sep='\n')
except Exception as e:
    print("There was a non-specific error!")
    print("Built-In Python error info: ")
    print(e, e.__doc__, type(e), sep='\n')
```

Figure 2: Exception Block

To illustrate what running the program and breaking it when there is no error handling implemented, see Figure 3. Notice that the errors shown are not particularly readable by the user and one may not be able to glean what went wrong when entering the "fifty."

```
C:\_PythonClass\Assignment07>python Assignment07.py
How much money can you spend on Christmas gifts?
Enter a name on your shopping list: Elizabeth
Enter a budget for this person($): fifty
Traceback (most recent call last):
  File "Assignment07.py", line 18, in <module>
    budget_int = int(input("Enter a budget for this person($): "))
ValueError: invalid literal for int() with base 10: 'fifty'
```

Figure 3: Broken program without error handling

```
C:\_PythonClass\Assignment07>python Assignment07.py
How much money can you spend on Christmas gifts?
Enter a name on your shopping list: Elizabeth
Enter a budget for this person($): fifty
Value entered for budget must be an integer!
Built-In Python error info:
invalid literal for int() with base 10: 'fifty'
Inappropriate argument value (of correct type).
<class 'ValueError'>
```

Figure 4: Broken program with error handling

After the ValueError exception block, there is a catch-all exception block which is used in the case that an unexpected error (that is, unforeseen by the programmer) occurs while the user runs the program. It is good practice to provide the user with coherent error messages when errors occur instead of letting Python provide its built-in error messages.

Pickling

Notice that at the beginning of the code, and shown in Figure 5, that there is something called pickling occurring in the script. Pickling is a way to store and retrieve data to/from binary (.dat) files. Binary files are similar to .txt files, but by using this type of file, higher efficiency can be achieved since this a way to reduce file size.

```
# ------------------------------------------------
# Title: Assignment 07
# Description: Demo of pickling and error handling
# ChangeLog (Who,When,What):
# JEmbury, 12/1/19,Created started script
# ------------------------------------------------

import pickle
```

Figure 5: Importing pickle, code from external source

In the example given, the data input by the user is “dumped” into a binary file. The data is then retrieved from the binary file and printed to the user.

```
C:\_PythonClass\Assignment07>python Assignment07.py
How much money can you spend on Christmas gifts?
Enter a name on your shopping list: Elizabeth
Enter a budget for this person($): 50
['Elizabeth', 50]
['Elizabeth', 50]
```

Figure 6: Program ran without any errors

In Figure 7, the contents of the binary file are shown. Note that the information contained in this file are less readable to a human as compared to utilizing a .txt file for storing information.

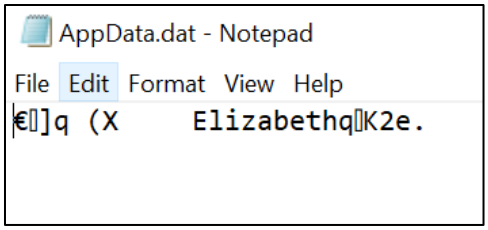

Figure 7: Contents of resultant binary file.

Summary

This document showed a demonstration of how error handling works in Python and how pickling is used to store and retrieve data in a binary file. Error handling with a try-exception block is a useful tool for improving the usability of a program by letting the program designer anticipate potential errors which could be introduced by the user during use, providing an opportunity to design-in customized error messages to handle the errors accordingly. By using pickling to store and retrieve data, a smaller amount of memory is used, as opposed to using a .txt file, to store/retrieve data. Pickling is an efficient means for saving data when running a Python program. Below is a link to the github webpage showing this document using markdown:

https://github.com/emburyj/IntroToProg-Python-Mod07/blob/master/docs/index.md