Jenn Wall

14 March, 2022

Foundations of Programming: Python

Assignment 07

Error Handling

Introduction

In this assignment we discussed how to introduce structured error handling in our code. Additionally, we covered the python exception class and how we can use it, adjust it, and add to it. We also investigated the use of binary files versus text files. Lastly, we touched on using markdown files with our code.

Benefits of Error Handling

Structured error handling adds great benefits to our program. It allows us to gracefully handle situations that do not meet the requirements of our code. When an error is encountered, we are able to let the user know and continue on with our program rather than crash. We can also customize our errors for the specific needs of our program.

Exception Class

The exception class is built into python and is used with error handling to help define the errors encountered. This allows us to collect and store data about different errors that we hit. Being able to capture information about different errors, then gives us the opportunity to further customize the errors returned to the user.

In python we can creating custom exception classes to fit the specific needs of our program. These can be defined as a function in our code, thus adding to the common exceptions. Because custom functions behave as child to the parent exception class, they inherit docstring content from the base class.

An example of using a custom, or derived, exception class is if we wanted to set an upper and lower limit to our math program. Say we want our user to enter an integer, but we don't want numbers less than zero or greater than 500. We can create new exception classes to implement this and provide the user with an error message if they enter a number outside of our set range.

Binary Files

The main difference between the text files we have been using and binary files is that text files are human readable, and binary files are not. As you would expect, binary files are written in binary. The upside to using binary files instead of text is that they take less memory to process. Saving to binary files is called pickling in python. The pickle module takes information from our program and stores it as binary data.

Markdown

A markdown file is a spec document for our program. It lets the reader know what our program does and how it functions. In githib, this is the readme.md file. This file can be added to and changed. When doing so the GitHub Flavored Markdown (GFM) method should be employed.

Update Our Script

For assignment 7 we were tasked with updating our script from last week to include error handling and binary file storage.

```
@staticmethod
def read_file(file_name, table):
    """Function to manage data ingestion from file to a list of dictionaries

    Reads the data from file identified by file_name into a 2D table
    (list of dicts) table one line in the file represents one dictionary row in table.

    Args:
        file_name (string): name of file used to read the data from

    Returns:
        table (list of dict): 2D data structure (list of dicts) that holds the data during runtime
    """
    table.clear()  # this clears existing data and allows to load data from file
    with open(file_name, 'rb') as objFile:
        while True:
            try:
                line = pickle.load(objFile)
                data = line.strip().split(',')
                dicRow = {'ID': int(data[0]), 'Title': data[1], 'Artist': data[2]}
                table.append(dicRow)
            except:
                break
    return table

@staticmethod
def write_file(file_name, table):
    """Function to write data from file to a list of dictionaries

    Writes the data to file identified by file_name into a 2D table
    (list of dicts) table one line in the file represents one dictionary row in table.

    Args:
        file_name (string): name of file used to read the data from

    Returns:
        None.
    """
    with open(file_name, 'wb') as objFile:
        for row in table:
            lstValues = list(row.values())
            lstValues[0] = str(lstValues[0])
            pickle.dump(','.join(lstValues) + '\n', objFile)
```

Figure 1. Converting to binary file

```python
@staticmethod
def CD_data_add():
    """Gathers CD data to add to memory

    Args:
        CD_data (list): ID, Title, Artist

    Returns:
        None
    """
    while True:
        try:
            strID = input('Enter ID: ').strip()
            intID = int(strID)
            strTitle = input('What is the CD\'s title? ').strip()
            strArtist = input('What is the Artist\'s name? ').strip()
            CD_data = [intID, strTitle, strArtist]
            return CD_data
        except:
            print('That is not a valid entry')
```

Figure 2. Exception handling

```
Which operation would you like to perform? [l, a, i, d, s or x]: a


Enter ID: no
That is not a valid entry
```

Figure 3. Running the program in spyder

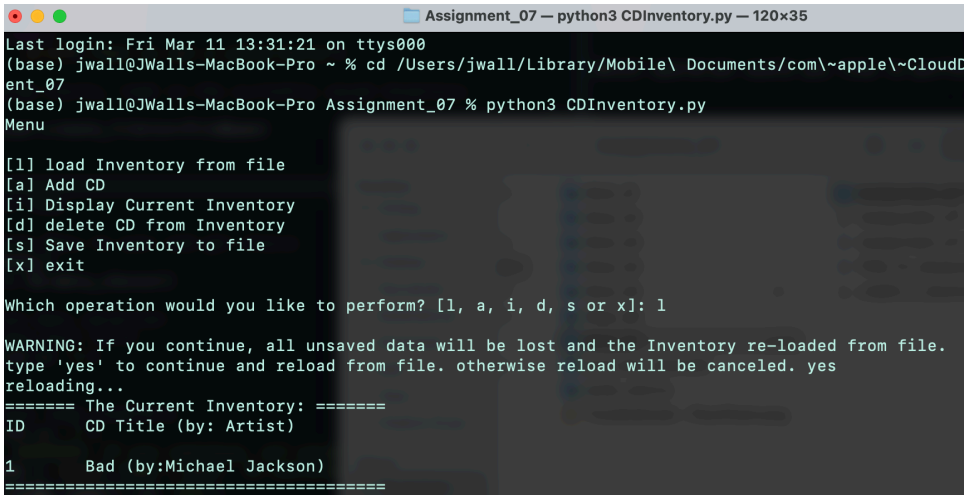

Figure 4. Running the program in terminal

https://github.com/jrwall704/Assignment_07

Summary

To summarize, this assignment was focused on integrating exception handling into our program, and using binary files instead of text files. I found the information at https://www.geeksforgeeks.org/python-exception-handling/ to be useful for cementing my knowledge about error handling.