

Dmitriy Alexandrov

November 20th, 2019

IT FDN 100 A

Assignment07

Python Error Handling and Pickling

Introduction

In this paper I am going to go through the steps that I took in creating a script that would demonstrate the pickle and error handling functions in Python.

Steps Taken

Per the assignment instructions, I first researched pickling and error handling in Python. Here are the following resources that I used for each:

Pickling:

<https://docs.python.org/2/library/pickle.html> - I looked in the Python documentation about pickling because I figured it would be the best way to understand the notion of it and how to use it.

<https://www.geeksforgeeks.org/understanding-python-pickling-example/> - I liked this resource because it gave code examples of how pickling works and what the outputs would look like. It also gave some advantages of using pickling as well which helped me further understand the usage of it.

Error Handling:

<https://www.pythonforbeginners.com/error-handling/> - I found this error handling for beginners' resource useful because it helped me understand how error handling worked in Python as well as how to use error handling in my own code in the future with examples.

<https://realpython.com/python-exceptions/> - Was another resource that I used and found very helpful because as it explained error handling in Python it gave examples and command outputs with each example to make it easier to understand and visually see what the outputs would be.

After I conducted my research on both topics, I went to create a script titled "Assignment07.py" in my `_PythonClass` directory to represent what I had learned. After creating the header and filling it out as needed, I started the creation of the "Pickle Rick 3000" application.

This application would allow the user to view their data, add more data, pickle data, unpickle data, and lastly exit. And as the user went about using the functions of the script, I would be incorporating both built-in Python exceptions as well as custom made ones.

The menu that the user would see looked as such:

```
Welcome to the Pickle Rick 3000:
Here are your options:
  1: View My Data
  2: Add Data
  3: Pickle My Data
  4: Unpickle My Data
  5: Exit
What option do you want to choose? [1 - 5]
```

The first option, would show the data that was stored in a list called `lstData` and depending on if that list was empty or not the user would either get a response of:

```
Your current data is:
empty
```

Or if they did have data, then the data would be displayed:

```
Your current data is:
{'Item': 'Test', 'Value': '12'}
```

The second (2) option that the user could choose would ask the user an item and a value to the item that they would like to add:

```
What option do you want to choose? [1 - 5] 2
What item would you like to add? Test
What value would you like to assign to that item? 12
```

This item would be added (appended) to the list (`lstData`) as a dictionary row (`dictRow`).

The third (3) option from the user menu would start off by asking the user which “.dat” file they would like to open to pickle their data to:

```
What option do you want to choose? [1 - 5] 3
What is the file name that you would like to open? *make sure it's a .dat file* |
```

If the user provided the script with a file name that did not include the “.dat” portion at the end then a custom error was generated:

```

What is the file name that you would like to open? *make sure it's a .dat file* Pickle
An error has occurred...
The Custom Error info:
File Extension is invalid, expecting .dat
Please try again.

```

The script then jumps back to the menu allowing the user to make another attempt at entering a correct file name with a “.dat” extension. If they did so, then the data in the list (lstData) was pickled and dumped into the file of their choice and the user was told that the pickling process was done and what file it was saved into.

***** During this process the data stored in the list (lstData) is taken and converted into a byte stream and saved in the .dat file that the user has specified in the same folder as the script is running in.***

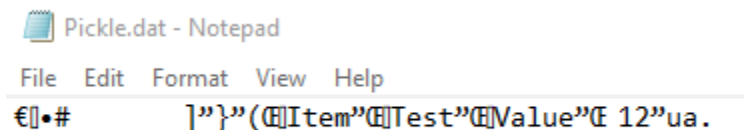
```

What option do you want to choose? [1 - 5] 3
What is the file name that you would like to open? *make sure it's a .dat file* Pickle.dat

Pickling your data...
Your data has been pickled in the Pickle.dat file.

```

This is what the output looked like at this point:



```

File Edit Format View Help
€[]•#      ]"}"(€Item"€Test"€Value"€ 12"ua.

```

Moving onto the fourth (4) option from the user menu, the script would again prompt the user for what file they wanted to open to pull pickled data from:

```

What option do you want to choose? [1 - 5] 4
What is the file name that you would like to open? *make sure it's a .dat file* |

```

Once again, if the user put in a file name that didn’t end in “.dat” the same exception was thrown as before. But also here if the user input a file name with a “.dat” that didn’t exist than a native Python exception of FileNotFoundError was thrown:

```

What option do you want to choose? [1 - 5] 4
What is the file name that you would like to open? *make sure it's a .dat file* test.dat
An error has occurred...
The Built-in Python Error info:
[Errno 2] No such file or directory: 'test.dat'
Please try again.

```

Allowing the user to once again try again in un-pickling their data by opening a file that actually exists. And if the user did input a valid “.dat” file, then the script goes through and unpickles the file and displays the content back to them.

```
What option do you want to choose? [1 - 5] 4
What is the file name that you would like to open? *make sure it's a .dat file* Pickle.dat

Your un-pickled data is:
[{'Item': 'Test', 'Value': '12'}]
```

Lastly, the fifth (5) option in the menu list just closes the script.

```
What option do you want to choose? [1 - 5] 5

Process finished with exit code 0
```

There was one more custom error that I included in the script, if the user input from the menu was not numeric then a custom NonNumericError handle was generated:

```
What option do you want to choose? [1 - 5] five
An error has occurred...
The Custom Error info:
The user input was not numeric, please try again.
```

Once again, letting the user know that an error had occurred and they would need to try another input.

Summary

Overall, the script achieved the showing of how Python both performs the pickling and unpickling of data and along side displayed how to properly handle errors that were either built-in to Python natively (FileNotFoundError) or a custom error (invalid File Extension or NonNumericInput error).

The final script looked as such:

```
# ----- #
# Title: Assignment07 - Python Error Handling and Pickling
# Description: Show how Python handles errors in code and
#               demonstrate how Pickling works in Python to
#               serialize/de-serialize a Python object structure
# ChangeLog: (Who, When, What)
# <Example Dev,01/01/2030,Created Script>
# DAAlexandrov, 11.20.2019, Created Script
# ----- #

import pickle

# -- Custom Exception -- #
```

```

class BinaryFileExtensionError(Exception): # custom exception if the file extension
is not .dat
    def __str__(self):
        return "File Extension is invalid, expecting .dat"

class NonNumericInputError(Exception): # custom exception if the user inputs a non-
numeric value
    def __str__(self):
        return "The user input was not numeric, please try again."

# -- Data -- #
lstData = []
dictRow = {}

# -- Presentation (I/O) -- #

def open_file(): # ask user for a file name
    file = input("What is the file name that you would like to open? *make sure it's
a .dat file* ")
    return file

# -- Main Program -- #

while(True):
    print("Welcome to the Pickle Rick 3000: \n Here are your options: ")
    print("\t1: View My Data")
    print("\t2: Add Data")
    print("\t3: Pickle My Data")
    print("\t4: Unpickle My Data")
    print("\t5: Exit")
    strUserInput = input("What option do you want to choose? [1 - 5] ")

    if (strUserInput.strip() == "1"): # show current data in the list
        print("\nYour current data is: ")
        if not lstData:
            print("empty \n")
        else:
            for row in lstData:
                print(row)
            print("\n")
            continue

        elif (strUserInput.strip() == "2"): # let the user add an item to the list using
a dictionary row
            strUserItem = input("What item would you like to add? ")
            strUserValue = input("What value would you like to assign to that item? ")
            print()
            dictRow = {"Item": strUserItem, "Value": strUserValue}
            lstData.append(dictRow)
            continue

        elif (strUserInput.strip() == "3"): # pickle the user data
            try:
                outputFile = open_file()

```

```

        if not outputFile.endswith(".dat"):
            raise BinaryFileExtensionError()
    except Exception as e: # Custom Exception thrown
        print("An error has occurred...")
        print("The Custom Error info:")
        print(e)
        print("Please try again. \n")
        continue
    else:
        print("\nPickling your data...")
        objF = open(outputFile, "wb")
        pickle.dump(lstData, objF)
        objF.close()
        print("Your data has been pickled in the " + outputFile + " file. \n")
        continue

elif (strUserInput.strip() == "4"): # unpickle the user data
    try:
        userFile = open_file()
        if not userFile.endswith(".dat"):
            raise BinaryFileExtensionError()
    except Exception as e: # Custom Exception thrown
        print("An error has occurred...")
        print("The Custom Error info:")
        print(e)
        print("Please try again. \n")
        continue
    else:
        try:
            objF = open(userFile, "rb")
        except FileNotFoundError as e: # throw an exception when the file cannot
be found
            print("An error has occurred...")
            print("The Built-in Python Error info:")
            print(e)
            print("Please try again. \n")
            continue
        else:
            output = pickle.load(objF)
            objF.close()
            lstData.append(output)
            print("\nYour un-pickled data is: ")
            print(output)
            print("\n")
            continue

elif (strUserInput.strip() == "5"): # exit per user request
    exit()
else:
    try:
        if not strUserInput.isnumeric():
            raise NonNumericInputError() # Throw custom non numeric input
exception
    except NonNumericInputError as e:
        print("An error has occurred...")
        print("The Custom Error info:")

```

```
        print(e)
    finally:
        continue
```