Jonah Jepson

March 1, 2023

IT FDN 110

Assignment07

https://jjepson3.github.io/IntroToProg-Python-Mod07/

Pickling and Structured Error Handling

Introduction

The purpose of this assignment is to create a script and an accompanying knowledge document that explains how to use the pickle module in python and demonstrates structured error handling. To do this we are going to look at some fictional examples that involve a student named Charlie Brown.

Pickling and Unpickling

Charlie Brown didn't do so well on the final exam, and he is afraid that he may not pass his class. While his teacher was out of the room, he decided to sneak a peak at the student grade file on her computer. To his dismay it all looked like a bunch of gibberish! That's because his teacher used the pickle module in python to save her grade file.

What is Pickle?

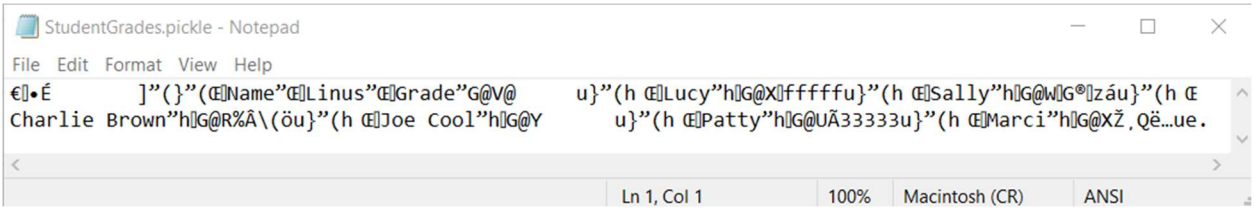
The python pickle module is used to serialize and de-serialize python objects. "Pickling" is the process that "serializes" or converts python objects into a byte stream (0's and 1's). Conversely, unpickling is the act of de-serializing byte streams back into python objects. This can be extremely useful because it allows for easier storage and transfer of data from system to system.

Pickling

To use the pickle module, it must be imported. At the beginning of the script type "import pickle" to load the pickle module. Next, to pickle your data and save it to a file you first open the file in binary write mode (using *open(file_name, 'wb')*)then you use the command *pickle.dump(your_data, file_name)* to pickle your data into a byte stream and write it to the file. For an example, see how Charlie Brown's teacher used python to pickle her grade book:

```python
import pickle
# Data ---------------------------------------------------------------------- #
# Declare variables and constants
GradeBook = [{'Name': 'Linus', 'Grade': 89.00},
             {'Name': 'Lucy', 'Grade': 96.35},
             {'Name': 'Sally', 'Grade': 92.27},
             {'Name': 'Charlie Brown', 'Grade': 67.59},
             {'Name': 'Snoopy', 'Grade': 100.00},
             {'Name': 'Patty', 'Grade': 87.05},
             {'Name': 'Marci', 'Grade': 98.23}]

GradeBookFile = 'StudentGrades.pickle'


# Processing  --------------------------------------------------------------- #
def write_grades(data, file):
    """  Serializes a msg in binary and uploads it to a file

    :param data: {dict} with student grades data:
    :param file: (string) containing name of the file:
    :return nothing
    """
    with open(file, 'wb') as file:
        pickle.dump(data, file)


# Presentation (Input/Output)  ---------------------------------------------- #
write_grades(GradeBook, GradeBookFile)

```

When we open the pickled 'StudentGrades.pickle' file in a text editor this is what we see:



Unpickling

Unpickling works similarly to pickling. At the top of the script you must import the pickle module then open the file in binary read mode (using *open(file_name, 'rb')*)and use the command *pickle.load(file_name,)* to read your data from the file and unpickle it from the byte stream. To unpickle his teachers grade file, Charlie Brown's Script needs to look like this:

```python
import pickle
# Data ---------------------------------------------------------------------- #
# Declare variables and constants
GradeBookFile = 'StudentGrades.pickle'


# Processing  --------------------------------------------------------------- #
def read_grades(file):
    """  Deserializes a msg in binary

    :param file: (string) containing name of the file:
    :return data: (list) of dictionary rows
    """
    with open(file, 'rb') as file:
        data = pickle.load(file)
    return data


# Presentation (Input/Output)  ---------------------------------------------- #
GradeBook = read_grades(GradeBookFile)
print(GradeBook)

```

This code will read the pickled data and output it in the form that it was input. See the output:

```
C:\_PythonClass\Assignment07\venv\Scripts\python.exe C:\_PythonClass\Assignment07\07-2.py
[{'Name': 'Linus', 'Grade': 89.0}, {'Name': 'Lucy', 'Grade': 96.35}, {'Name': 'Sally', 'Grade': 92.27}, {'Nam

Process finished with exit code 0
```

Structured Error Handling

You may be wondering what happens if you try to unpickle a file by opening it in read mode (using *open(file_name, **'r'**)*) instead of binary read mode (using *open(file_name, **'rb'**)*). An error will occur, and the program will crash. Errors like this can be handled using structured error handling.

What is structured error handling?

Structured error handling allows you to control what happens to your code if an error occurs by using a try-except block. This is what a try-except block looks like:

```
try:
    <some code>

except:
    <some other code>

```

Basically, this tells the program to try and run the first code. If that code causes an error, then run the second code instead. In the case of trying to read a pickled file we can handle the error like this:

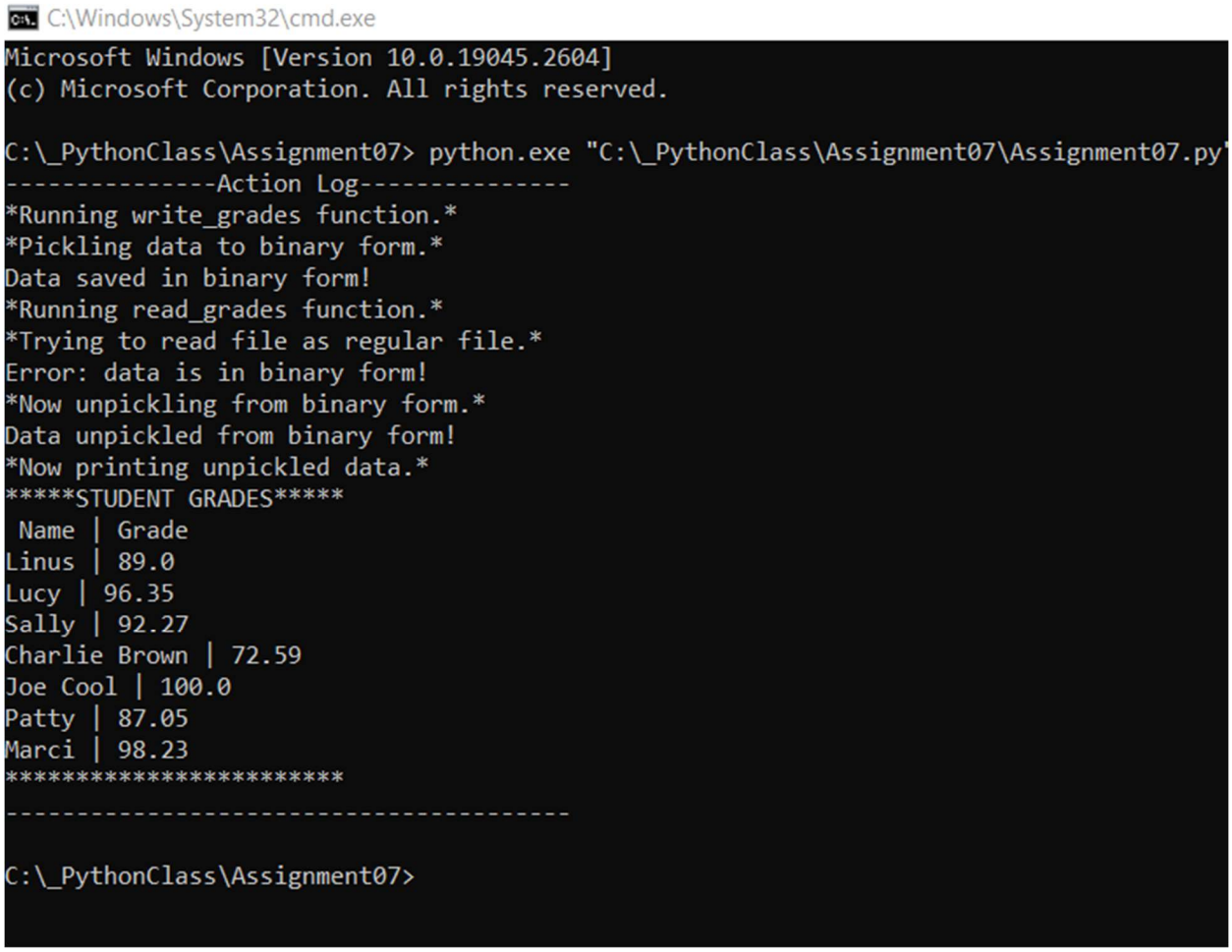
```
import pickle
# Data ---------------------------------------------------------------------- #
# Declare variables and constants
GradeBookFile = 'StudentGrades.pickle'


# Processing  --------------------------------------------------------------- #
def read_grades(file_name):
    """  unpickles a binary file.

    :param file_name: (string) containing name of the file:
    :return data: (list) of dictionary rows with student grades
    """
    try:
        print('*Trying to read the file as regular file.*')
        with open(file_name, 'r') as file:
            print(file.read())

    except:
        print('Error: data is in binary form!\n'
              '*Now unpickling from binary form!*')
        with open(file_name, 'rb') as file:
            data = pickle.load(file)

    return data


# Presentation (Input/Output)  ---------------------------------------------- #
GradeBook = read_grades(GradeBookFile)
print(GradeBook)
```

This will output the following:

```
C:\_PythonClass\Assignment07\venv\Scripts\python.exe C:\_PythonClass\Assignment07\07-3.py
*Trying to read the file as regular file.*
Error: data is in binary form!
*Now unpickling from binary form!*
[{'Name': 'Linus', 'Grade': 89.0}, {'Name': 'Lucy', 'Grade': 96.35}, {'Name': 'Sally', 'Grade

Process finished with exit code 0
```

Conclusion

With this assignment I have explained and demonstrated Pickling and Structured Error Handling. I've combined code used in the examples above into one script to demonstrate how pickling and structured error handling can be used in tandem. First the code pickles the data and writes it to a file. Then it tries to read the file as a regular file and when that errors, it reads the file as a binary file. Finally, it unpickles the file and formats it in a way that is easy to read. See the output below.



```
C:\_PythonClass\Assignment07\venv\Scripts\python.exe C:\_PythonClass\Assignment07\Assignment07.py
---------------Action Log---------------
*Running write_grades function.*
*Pickling data to binary form.*
Data saved in binary form!
*Running read_grades function.*
*Trying to read file as regular file.*
Error: data is in binary form!
*Now unpickling from binary form.*
Data unpickled from binary form!
*Now printing unpickled data.*
*****STUDENT GRADES*****
 Name | Grade
Linus | 89.0
Lucy | 96.35
Sally | 92.27
Charlie Brown | 72.59
Joe Cool | 100.0
Patty | 87.05
Marci | 98.23
**********
----------------------------------------

Process finished with exit code 0
```