Chanel M. Leung

June 02, 2020

Foundations of Programming: Python

Assignment 07

https://github.com/cml-python/ITFnd100-Mod07

Pickling & Error Handling

Introduction

This module allowed me to explore ideas of how to use different functions and techniques within one script that I would create without the guidelines that we had used before. I learned about pickling, which helps serialize the data in a file, and error handling using try-except that prevents bugs from affecting the code that way the script will try and put the data through the process, otherwise if it does not work, it'll use the except block to catch errors that can be corrected by the user through the script.

Deciding Script to Use

Deciding what type of script to create that would include these techniques took some time to decide, but since error handling within our module dealt with dividing numbers and how to deal with numerical errors, I decided to use a script that would divide numbers. I wanted to use pickling as well, so I decided to save the data of the numbers divided onto a data file. I remembered that we went through a math processor lab in the previous module, so I thought that I would utilize parts of that within the script to exemplify the pickling and error handling (Figure 1).

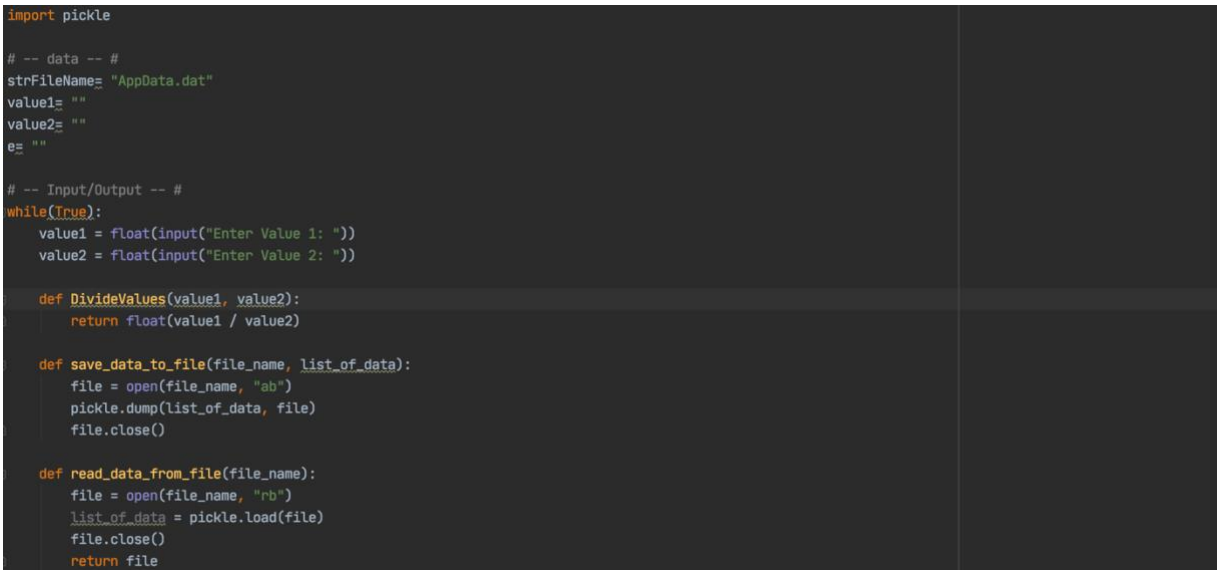
```
# -- Input/Output -- #
while(True):
    value1 = float(input("Enter Value 1: "))
    value2 = float(input("Enter Value 2: "))

    def DivideValues(value1, value2):
        return float(value1 / value2)
```

Figure 1

Adding Pickling

I was able to use pickling when adding the solved division problem to the file by first using import pickle, the being able to save and read the pickle to binary language (Figure 2). I learned more about pickling through this website: https://docs.python.org/3/library/pickle.html through this webpage I was able to learn about different serializations such as marshal and json in Java. It mentioned as well what was mentioned in the videos about pickle.dump and pickle.load but also expanded into more detail of what each does, for example, dump meant to write the pickled representation of obj to open the file object (Figure 2). I also learned about different things that are able to be pickled and unpickled which in my script I was able to with integers and strings of words. In the end I was able to do it where after I would enter the two values and then print out The Quotient of x and y is x/y which would read that it was saved to the file by outputting <_io.BufferedReader name='AppData.dat'>. I through learning how to utilize pickling for data to be stored can in the future help more with tons of data.



```
import pickle

# -- data -- #
strFileName= "AppData.dat"
value1= ""
value2= ""
e= ""

# -- Input/Output -- #
while(True):
    value1 = float(input("Enter Value 1: "))
    value2 = float(input("Enter Value 2: "))

    def DivideValues(value1, value2):
        return float(value1 / value2)

    def save_data_to_file(file_name, list_of_data):
        file = open(file_name, "ab")
        pickle.dump(list_of_data, file)
        file.close()

    def read_data_from_file(file_name):
        file = open(file_name, "rb")
        list_of_data = pickle.load(file)
        file.close()
        return file
```

Figure 2

Adding Error Handling

Error handling confused me a bit at first when talking about it, but after going through some examples I was able to look at the website https://docs.python.org/3/tutorial/errors.html to research more into the try-except block to use it in my script. It allowed me to go through

the process which would try to do the function I wanted it to do, otherwise after it was tried, it would go through the except blocks in order to allow the user to understand why the script did not work. In the script I was able to use the try and except to divide the numbers, otherwise if not able, it would print whether the user entered an invalid character, can't be divided by zero otherwise it would print the results of the divided values (Figure 3).

```
try:
    divide= float(value1/value2)
except ValueError:
    print('Invalid character. Please enter a number.')
except ZeroDivisionError:
    print('Divisor cannot equal zero. Please enter a valid divisor.')
except Exception as e:
    print('Something went wrong.')
    print("Python Built-In error message:", (e))
else:
    print("The Quotient of %.2f and %.2f is %.2f" % (value1, value2, DivideValues(value1, value2)))
    break
```

Figure 3

Summary

Through this module I was able to make a script that would divide two numbers that were entered by the user. The two numbers would divide, unless the quotient was 0, a divisor was 0, or if there was an invalid character. The equation would also be added to a data text file in binary to serialize the data which may reduce the files size. I thought it was interesting to look up how others would explain pickling and different error handling ways other than just resources that we are given.