Julia Moekkoenen
August 25, 2023
IT FDN 110 A
Assignment 07
https://juliamoe.github.io/IntroToProg-Python-Mod07/

Pickling and Error Handling in Python

Introduction

The goal of the assignment was to research and explore pickling and error handling in Python. An example script was created to accomplish both of the tasks. The code performed simple calculations and prompted the user to provide input values. After performing the calculations the user could write and/or read the data into a binary file.

```
juliamokkonen@Julias-MacBook-Pro Assignment07 % python3 Assignment07.py
Would you like to
 a) Add the values together?
 b) Substract the first value from the second value?a
Please enter the first value: 1
Please enter the second value: 2
The addition value is:  3

Would you like to write the data into a file? [y/n]y
Value is saved into the file
Would you like to read the data in file? [y/n]y
4 <class 'int'>
juliamokkonen@Julias-MacBook-Pro Assignment07 %
```

Figure 1.1a: Working code in the Mac Terminal

```
juliamokkonen@Julias-MacBook-Pro Assignment07 % python3 Assignment07.py
Would you like to
 a) Add the values together?
 b) Substract the first value from the second value?b
Please enter the first value: 1
Please enter the second value: t
The provide an integer.
juliamokkonen@Julias-MacBook-Pro Assignment07 %
```

Figure 1.1b: Working code in the Mac Terminal

Assignment07.py × Calculations.dat

```python
# Title: Assignment 07
# Dev:JMoekkoenen
# Date: 8/25/2023
# Exploring error handling on a simple calculator
#-------------------------------------------------------------#
import pickle

2 usages
class Processor:
    2 usages
    @staticmethod
    def WriteToFile(value):
        pass
        print()
        des2=input("Would you like to write the data into a file? [y/n]")
        if des2=='y':
            objFile =open("Calculations.dat","ab")
            pickle.dump(value,objFile)
            objFile.close()
            print("Value is saved into the file")
        if des2=='n':
            print()
        readFile = input("Would you like to read the data in file? [y/n]")
        if readFile=='y':
            objFile=open("Calculations.dat","rb")
            objFileData=pickle.load(objFile)
            objFileData += pickle.load(objFile)

            objFile.close()
            print(objFileData,type(objFileData)) #printing the


while(True):
    des1=str(input("Would you like to \n a) Add the values together? \n b) Substract the first value from the second value?"))
    try:
        val1=int(input("Please enter the first value: "))
    except Exception as e:
        print("The provide an integer.")
        break
    try:
        val2=int(input("Please enter the second value: "))
    except Exception as e:
        print("The provide an integer.")
        break

    if des1 == "a":
        add=val1+val2
        print("The addition value is: ",add)
        Processor.WriteToFile(add)
        break
    elif des1 == "b":
        subs = val2-val1
        print("The substraction value is:",subs)
        Processor.WriteToFile(subs)
        break
    else:
        print("Please provide a or b.")
        continue
```

Figure 1.1b: Script for the assignment

Pickling

Pickling allows the user to read, write, and append data to/from a .dat file. Instead of the common read = ‘r’, write = ‘w’, and append = ‘a’ commands, pickling used ‘rb’, ‘wb’, and ‘ab’ respectively. In append and write cases the file is opened in ‘ab’ or ‘wb’ modes, and pickle.dump() function is applied to add data into the file. When reading data the file is

opened in ‘rb’ mode and pickle.load() function is used to load the data. The advantages of pickling are that it keeps track of the objects it has already processed and won’t duplicate them, this way saving space on the machine (GeeksforGeeks). To use pickling an external pickle module has to be imported into the code by using ‘import pickle’ function.

```
des2=input("Would you like to write the data into a file? [y/n]")
if des2=='y':
    objFile =open("Calculations.dat","ab")
    pickle.dump(value,objFile)
    objFile.close()
    print("Value is saved into the file")
if des2=='n':
    print()
```

Figure 1.2a: Example of appending data into a binary file.

```
readFile = input("Would you like to read the data in file? [y/n]")
if readFile=='y':
    objFile=open("Calculations.dat","rb")
    objFileData=pickle.load(objFile)
    objFileData += pickle.load(objFile)

    objFile.close()
    print(objFileData,type(objFileData))
```

Figure 1.2b: Example of reading data from a binary file.

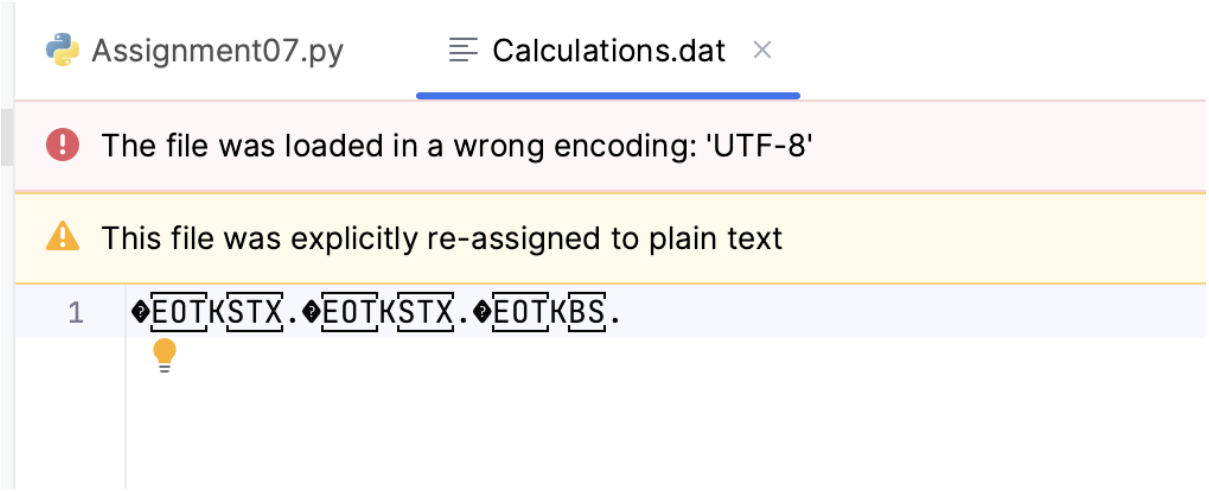

Figure 1.2c: Binary file in the folder

Error Handling

Error handling helps the developer to limit the type of answers the user could give. For example, in Figure 1.3 the questions are expecting the user to provide an integer values 1 and 2. If the user inputs an answer that isn’t an integer, the program will prompt the user to input an integer instead and break the loop. The error is caught by *try-except* clause, but if

no exceptions are detected the *except* clause will be skipped and the script moves on to the next line below (Handling Exceptions).

```
while(True):
    des1=str(input("Would you like to \n a) Add the values together? \n b) Substract the first value from the second value?"))
    try:
        val1=int(input("Please enter the first value: "))
    except Exception as e:
        print("The provide an integer.")
        break
    try:
        val2=int(input("Please enter the second value: "))
    except Exception as e:
        print("The provide an integer.")
        break
```

Figure 1.3: Example of error handling

Summary

The goal of the assignment was to research and explore pickling and error handling in Python. The script created for this accomplished both tasks by reading, and saving data into a binary file by using pickling, and error handling by checking if the values given for the value 1 and value 2 were actually integers.