Thomas Farmer

3-1-2022

Foundations Of Programming

Assignment07

https://github.com/MtnWolf82/IntroToProg-Python-Mod07

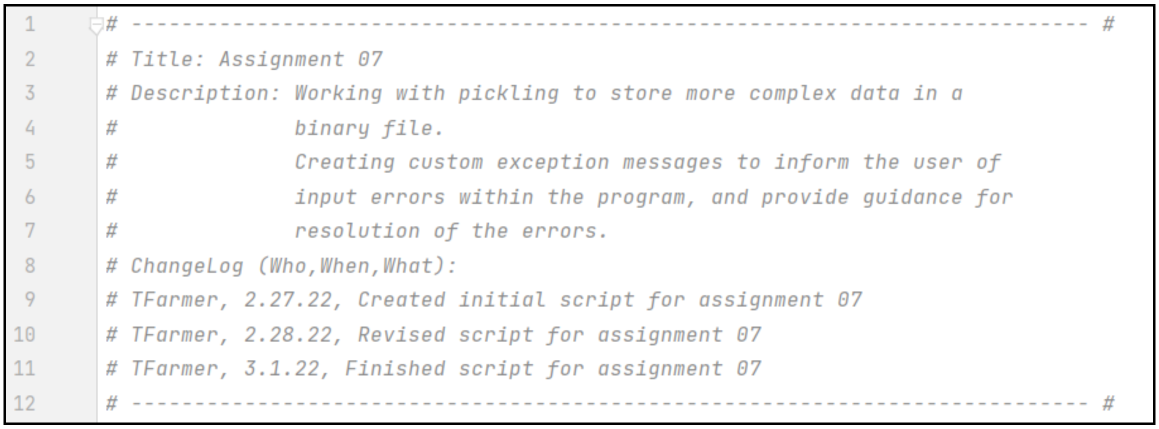
Pickling Data & Exception Handling

INTRODUCTION

There were two different topics covered this week – Pickling Data and Exception Handling. Let's start with pickling data. It can best be thought of as a way to store more complex data in a more efficient format/file for recall at a later time. Exeption handling is a way for the programmer to provide more user-centric feedback to the user when they make an error – typically in the form of data entry.

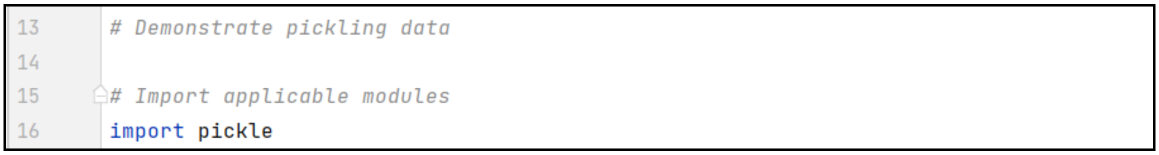
THE PROCESS – PICKLING DATA

1. Open a new PyCharm file and enter your program’s header comments. An example of this assignment's comments can be seen below.



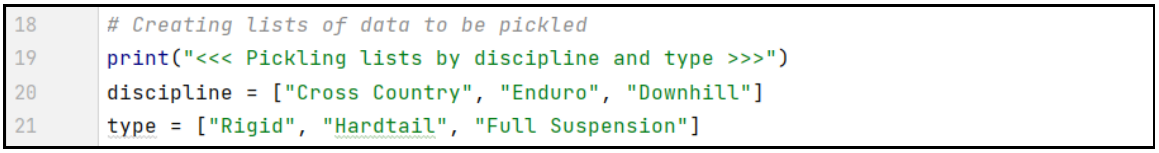
```
# ---------------------------------------------------------------------------- #
# Title: Assignment 07
# Description: Working with pickling to store more complex data in a
#              binary file.
#              Creating custom exception messages to inform the user of
#              input errors within the program, and provide guidance for
#              resolution of the errors.
# ChangeLog (Who,When,What):
# TFarmer, 2.27.22, Created initial script for assignment 07
# TFarmer, 2.28.22, Revised script for assignment 07
# TFarmer, 3.1.22, Finished script for assignment 07
# ---------------------------------------------------------------------------- #
```

2. We'll be working with the 'pickling' module, so let's be sure to import it into our program. Enter the code shown below.



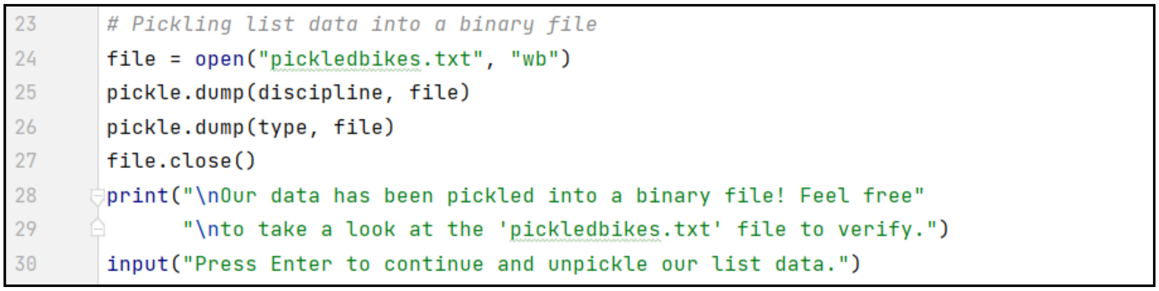
```
# Demonstrate pickling data

# Import applicable modules
import pickle
```

3. Now let's create a couple lists of data to be pickled. For the sake of something a little different, I've used a couple bike related lists. Enter the code shown below.



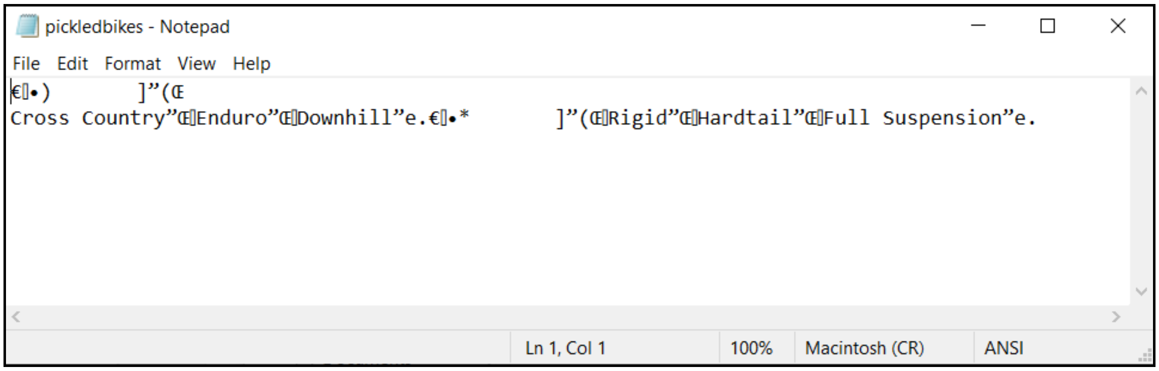
```python
# Creating lists of data to be pickled
print("<<< Pickling lists by discipline and type >>>")
discipline = ["Cross Country", "Enduro", "Downhill"]
type = ["Rigid", "Hardtail", "Full Suspension"]
```

4. With our lists created, we can now pickle our data. By using the “wb” file access mode, we can let the program create our text file for us. In this case, the text file “pickledbikes” will be created in the same directory folder as our program. Each list is then 'dumped' into our text file and the file then closed. Enter the code shown below.



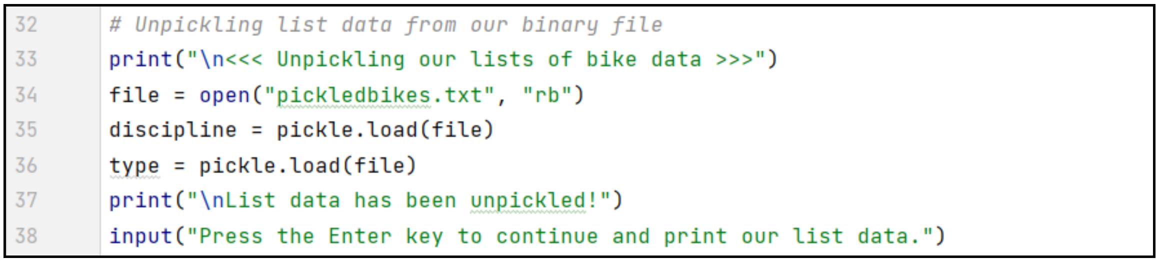
```python
# Pickling list data into a binary file
file = open("pickledbikes.txt", "wb")
pickle.dump(discipline, file)
pickle.dump(type, file)
file.close()
print("\nOur data has been pickled into a binary file! Feel free"
      "\nto take a look at the 'pickledbikes.txt' file to verify.")
input("Press Enter to continue and unpickle our list data.")
```

5. Here's an example of the type of information you should see when viewing your “pickledbikes” text file. Note that much of the data is jumbled do to its binary storage.



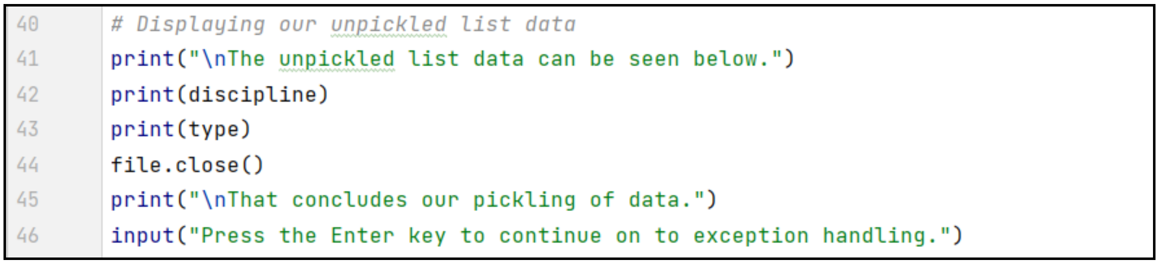
```
€▯•)        ]”(Œ
Cross Country”Œ▯Enduro”Œ▯Downhill”e.€▯•*        ]”(Œ▯Rigid”Œ▯Hardtail”Œ▯Full Suspension”e.
```

6. It is now time to unpickle our data. We do this by against opening our text file, but we'll use the “rb” file access mode to simply read from the file. Each list is then loaded back into our program from the text file. Enter the code shown below.



```python
# Unpickling list data from our binary file
print("\n<<< Unpickling our lists of bike data >>>")
file = open("pickledbikes.txt", "rb")
discipline = pickle.load(file)
type = pickle.load(file)
print("\nList data has been unpickled!")
input("Press the Enter key to continue and print our list data.")
```

7. We will now display our unpickled list data loaded in the previous step. Once printed, we close the file as it no longer required to be open. Enter the code shown below.



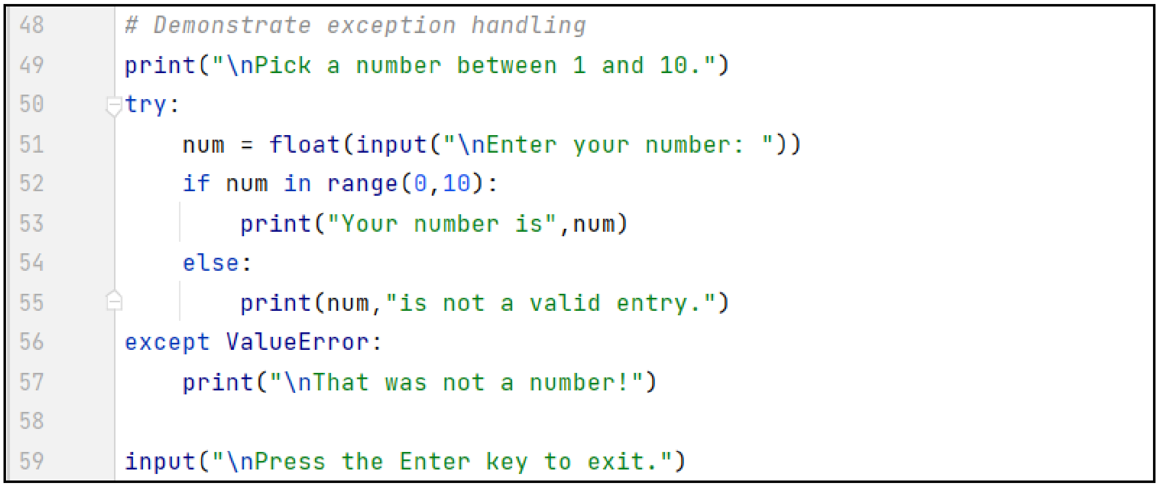
```
# Displaying our unpickled list data
print("\nThe unpickled list data can be seen below.")
print(discipline)
print(type)
file.close()
print("\nThat concludes our pickling of data.")
input("Press the Enter key to continue on to exception handling.")
```

Add'l resources:

- https://realpython.com/python-pickle-module/ – User friendly breakdown of the pickle module and its usage. Gives plenty of examples and does a good job of describing its intent and limitations.
- https://docs.python.org/3/library/pickle.html – More in-depth technical description of the pickle module and its associated applications.

THE PROCESS – EXCEPTION HANDLING

1. Next, we'll look at an example of exception handling. In the example below, we query the user for a number between 1 and 10. If a number outside of the range is received, the user will receive the message stating the entry is invalid. However, if they enter something other than a number, such as text, they'll be met with a statement that their entry was not a number, rather than the standard error exception message for 'ValueError'. Enter the code shown below.



```
# Demonstrate exception handling
print("\nPick a number between 1 and 10.")
try:
    num = float(input("\nEnter your number: "))
    if num in range(0,10):
        print("Your number is",num)
    else:
        print(num,"is not a valid entry.")
except ValueError:
    print("\nThat was not a number!")

input("\nPress the Enter key to exit.")
```

Add'l resources:

- https://realpython.com/python-exceptions/ – User friendly breakdown of exceptions and exception handling. Gives plenty of examples and does a good job of describing its intent and limitations. There's also a related course!
- https://www.w3schools.com/python/python_try_except.asp – Provides clear definitions of each function and incremental examples. You can also step into the example code and make tweaks to the code to see how it affects the program.

- https://docs.python.org/3/tutorial/errors.html – More in-depth technical description of exception handling and errors. Also goes into more advanced exception handling technics.

SUMMARY

The pickle module sounds like it's pretty useful for storing larger amounts of data for recall at a later time. I did manage to find some pretty useful links providing additional guidance with its applications and usage. The exception handling is a very handle tool. While general exceptions can be figured out with a bit of familiarity, having the ability to convey more user-centric feedback is handy for making sure the user if providing the data necessary to utilize the program. I need to do a bit more exploring with this particular topic.