

Pickling and Error Handling

Introduction

This week's assignment was a little different from past week's. We had to first research websites explaining pickling and error handling then explain why those sites were noteworthy. Next we had to create a script that demonstrates how pickling and error handling work. To keep things simple I took the concept of creating and storing tasks from previous weeks and added pickling and error handling features to the code.

Body

Research Websites

The first part of this assignment was to research websites about pickling and error handling and explain why we thought those websites were good at explaining the subject. I first choose to research pickling. I went through several websites that looked somewhat complex and verbose before finding <https://www.tutorialspoint.com/python-pickling>. I like this website because it starts with a brief explanation of what pickling is without going into complex detail. It also provides simple examples of how to pickle and unpickle. Figure 1 and Figure 2

Pickle a simple list: Pickle_list1.py

```
import pickle
mylist = ['a', 'b', 'c', 'd']
with open('datafile.txt', 'wb') as fh:
    pickle.dump(mylist, fh)
```

(Figure 1: Simple Pickling Example)

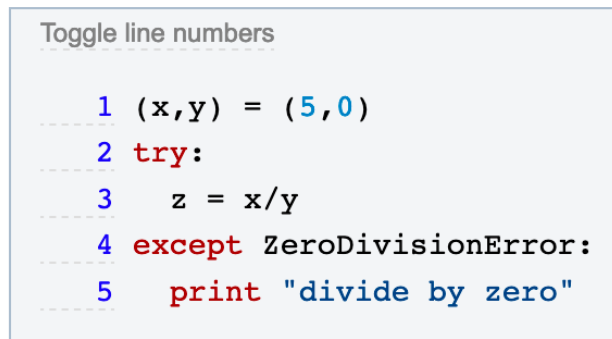
Unpickle a simple list: unpickle_list1.py

```
import pickle
pickle_off = open("datafile.txt", "rb")
emp = pickle.load(pickle_off)
print(emp)
```

(Figure 2: Simple Unpickling Example)

I then started researching websites on Error Handling. I was finding a similar problem I had when researching about pickling in that a lot of websites try to cover the topic in all it's complexity. That made it hard to just get a foothold of understanding. Eventually I found this website: <https://wiki.python.org/moin/HandlingExceptions> that I thought explained it well.

Similar to my example with pickling, I liked this website because they kept the explanation to a minimum and provided simple examples to get their point across. (Figure 3)



```
Toggle line numbers

1 (x,y) = (5,0)
2 try:
3     z = x/y
4 except ZeroDivisionError:
5     print "divide by zero"
```

(Figure 3: Simple Error Handling Example)

Writing Code

The next step in this assignment was to come up with our own code demonstrating how to use Pickling and Error Handling. For simplicity I continued using the theme of past assignments of working with tasks. I mostly copied previously used code and added in pickling and error handling where relevant.

I added pickling first. Pickling converts python objects into bytes. One advantage to doing this is it reduces the file size compared to saving as text. The task code I copied took a user's input (a task and priority) then stored it into a text file. Instead of storing into a text file I only had to make a few changes to pickle the data instead. To work with pickling data I had to import the "pickle" module. To pickle the data I had to use the "wb" attribute instead of "w" then use pickle method pickle.dump() instead of .write(). To unpickle the data I had to open the file with the "rb" attribute instead of "r" and use the pickle.load() method to read the file.

The next part was to add code to demonstrate error handling. The error handling technique we learned this week was try-except. When you run into an error Python will give the user its default code. However this code isn't always easy to understand, especially if you're not a programmer. By adding error handling to your code you can anticipate and manage human introduced error. Since we are working with files I thought it would be good to load a pickled file first then use try-except to check if file exists. I used the specific exception class "FileNotFoundError" to specifically check if file exists. Specific exception classes are used to check for specific errors. If it doesn't then a custom message is displayed letting the user know the file doesn't exist then moves on to the next code. If the file exists, it loads it then continues with the code. After inputting a task the user has the option to input another task by entering "next" or continue to the next part of the code by entering "exit" The second try-except I added was to "raise" an exception if the user didn't use one of these keywords. Using "raise" allows you to check for a condition then execute an exception.

Testing the Code

I first run the code without a file to demonstrate raising the exception when the file is not found. (Figure 4)

```
Try loading existing pickled data:  
Custom message: 'File not found'
```

(Figure 4: PyCharm File Not Found)

After inputting a task and priority I purposely input an incorrect text to force another exception (Figure 5)

```
What is the new task name?: task1  
What is the new task priority? (high/med/low): low  
Type 'exit' to exit or 'next' to input another task asdf  
Python message: Please use 'exit' or 'next' keywords  
Common base class for all non-exit exceptions.  
Python type: <class 'Exception'>
```

(Figure 5: PyCharm Wrong Keywords Exception)

I then continue with the code which shows what my data looks like pre-pickled, pickles and stores it, then shows what it looks like post-pickled. (Figure 6)

```
Here are your tasks pre-pickled:  
task1 | low  
task2 | high  
Pickling tasks...  
Here are your tasks unpickled:  
task1 | low  
task2 | high
```

(Figure 6: PyCharm Pickling data)

Here's what the byte file looks like (Figure 7)

```
Pickle.dat  
1  | 00@]0({0(0 Task00 task100Priority00 low0u}0(h 0 task20h 0 high0ue.  
.....
```

(Figure 7: PyCharm Byte File)

Now I have to run through the same process via the Terminal. I first open the file which automatically loads what's in the pickled file (Figure 8)

```
Try loading existing pickled data:  
task1 | low  
task2 | high  
Successfully loaded file
```

(Figure 8: Terminal Load file)

Next I enter a new task and purposely mistype "2" when it asks me what to do next to force an error. (Figure 9)

```
What is the new task name?: task3
What is the new task priority? (high/med/low): med
Type 'exit' to exit or 'next' to input another task 2
Python message: Please use 'exit' or 'next' keywords
Common base class for all non-exit exceptions.
Python type: <class 'Exception'>
```

(Figure 9: Terminal Wrong Keywords Exception)

I then went back through the program typing everything correctly this time. (Figure 10)

```
What is the new task name?: task3
What is the new task priority? (high/med/low): med
Type 'exit' to exit or 'next' to input another task exit
Here are your tasks pre-pickled:
task3 | med
Pickling tasks...
Here are your tasks unpickled:
task3 | med
```

(Figure 10: Terminal Pickling and Unpickling)

Conclusion

For this week's assignment we learned about pickling and error handling. The first part of the assignment we had to research each topic online and provide examples of website we thought explained those topics well. Then we had to write our own code demonstrating how pickling and error handling work. I used the task program framework we used in previous assignments. Instead of storing in a text file I stored in a pickled file. I then added in try-except error handling I places I thought made sense. Finally