Exceptions and Pickling

# Introduction

We have been asked to describe and demonstrate pickling and exception handling. I decided to work with the To Do List program from the module 6 assignment. That way the new techniques can be directly compared to the old. The program reads an existing list and can add, delete, and print the tasks and their priorities, or save the information back to the file. When adding tasks, the program asks for the name of a task, and then asks for its priority. Both pieces of data are stored in a dictionary, where each task and priority are a row of data. Each dictionary entry is then added to a list to create a table of data which can then be printed or saved. This is similar to the To Do List program developed for the Module 5 assignment, but uses classes and functions to organize the script.

# Exception Handling

I will start by looking at exception handling. What is an exception and why do we want to handle it?

According to the Tutorials Point website,

An exception is an event, which occurs during the execution of a program that disrupts the normal flow of the program's instructions. In general, when a Python script encounters a situation that it cannot cope with, it raises an exception. An exception is a Python object that represents an error. (<https://www.tutorialspoint.com/python/python_exceptions.htm>, 2020) (External Site)

In Python, when an error occurs, an exception is raised and the programmer can choose how to address it. Different kinds of errors result in different types of exceptions, some of which are easier to understand than others. “ZeroDivisionError” is pretty self-explanatory, but something like “KeyError” or “SyntaxError” will not be very helpful to most users.

In the module 6 code, the first step for the script is to load into memory any data from a text file, in this case “ToDoFile.txt.” The problem, of course, is that the text file may not exist, as was the case the first time I ran the script. Since the text file ToDoList.txt did not exist, the program gave the “FileNotFoundError” shown in Figure 1.

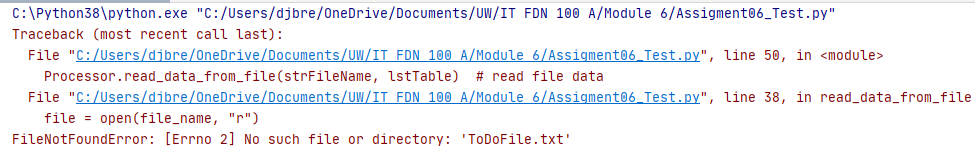


Figure : Missing file error

This particular error is pretty clear: No such file or directory exists. However, the user will not necessarily know what to do with that information. Try to create the file? Where does it go? What data should be in it? There needs to be something in the code to address the fact that the file may not exist without confusing the user.

In my case, rather than creating an initial text file, I had imported os.path and checked to see if the file existed as shown in Figure 2.

*# Data ---------------------------------------------------------------------- #  
# Declare variables and constants*from os import path

…

list\_of\_rows.clear() *# clear current data*if path.exists(file\_name):  
 file = open(file\_name, **"r"**)  
 for line in file:  
 task, priority = line.split(**","**)  
 row = {**"Task"**: task.strip(), **"Priority"**: priority.strip()}  
 list\_of\_rows.append(row)  
 file.close()  
return list\_of\_rows, **'Success!'**

Figure : Code to check if file exists

If it existed, the data was read from the file. If not, the function bypassed the file request and returned lstTable as empty.

A simpler method would have been to use Python’s exception handling functionality, specifically the try…except, or try…catch, blocks. I modified the code as shown in Figure 3.

list\_of\_rows.clear() *# clear current data*try:  
 file = open(file\_name, **"r"**)  
 for line in file:  
 task, priority = line.split(**","**)  
 row = {**"Task"**: task.strip(), **"Priority"**: priority.strip()}  
 list\_of\_rows.append(row)  
 file.close()  
except FileNotFoundError:  
 file = open(file\_name, **"w"**)  
 file.close()  
 print(**"Task list empty."**)  
return list\_of\_rows, **'Success!'**

Figure : Code for try..except error handling

Instead of importing the module ‘os’ and checking for the file with an if statement, the try block attempts to open the file. If the file is not found, the exception is caught and the file is created (see Figure 4).

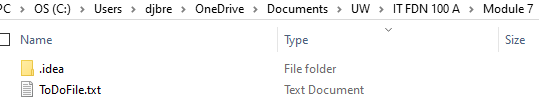


Figure : ToDoFile.txt created

The user is informed that the task list is empty as shown in Figure 5. Notice that the program continues and returns the empty list\_of\_rows, which is shown to the user as well.

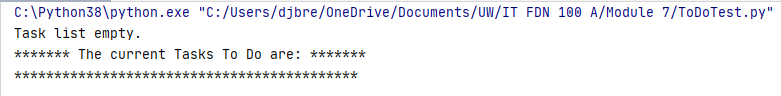


Figure : Result of except block

I could have used a generic except clause without the error name, but that would handle *any* type of error. I have learned to account for the text file possibly missing, but if something else is raising exceptions, I want to know what it is (spoiler alert: there is).

# What is Pickle and Why?

Pickle is a Python module that “allows you to take a complex object structure and transform it into a stream of bytes that can be saved to a disk or sent over a network.” (Real Python, <https://realpython.com/python-pickle-module/>, 2020) (External Site).

This process is known as serialization.

Data serialization is the process of converting structured data to a format that allows sharing or storage of the data in a form that allows recovery of its original structure. In some cases, the secondary intention of data serialization is to minimize the data’s size which then reduces disk space or bandwidth requirements. (The Hitchhiker’s Guide to Python, <https://docs.python-guide.org/scenarios/serialization/>, 2020) (External Site)

Reversing the process is known as deserialization or unpickling.

Again, I modified the code from Module 6 to create a To Do list. I imported the pickle module and changed the file name variable to “ToDoFile.dat”.

The .dat file extension can contain binary or text data, so it offers more flexibility for saving complex data. According to File Info,

All files can be categorized into one of two file formats — binary or text. The two file types may look the same on the surface, but they [encode](https://techterms.com/definition/encoding) data differently. While both [binary](https://techterms.com/definition/binary) and text files contain data stored as a series of bits (binary values of 1s and 0s), the [bits](https://techterms.com/definition/bit) in text files represent characters, while the bits in binary files represent custom [data](https://techterms.com/definition/data). (<https://fileinfo.com/help/binary_vs_text_files>, 2020) (External Site)

## Reading from the File (pickle.load)

I changed the ‘read\_data\_from\_file’ function to open the file as binary and to use pickle.load rather than reading each line (see Figure 6).

try:  
 file = open(file\_name, **"rb"**)  
 list\_of\_rows = pickle.load(file\_name)  
 file.close()  
except FileNotFoundError:  
 file = open(file\_name, **"wb"**)  
 file.close()  
 print(**"Task list empty."**)

Figure : Code to read data from file

On the first run, the file did not exist, so the except block created the file. Running the code again threw a “file must have ‘read’ and ‘readline’ attributes” error that was a little confusing (see Figure 7)

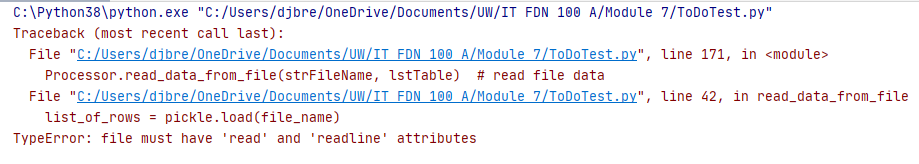


Figure : Unexpected error

Looking closer at the description for the pickle.load function (<https://docs.python.org/3/library/pickle.html>), it needs a file *object* rather than a file path or name like the open function. I exchanged the file name variable for the file object variable as shown in Figure 8.

try:  
 file = open(file\_name, **"rb"**)  
 list\_of\_rows = pickle.load(file)  
 file.close()

Figure : Code for pickle.load

This time, I discovered that empty files cause an error as shown in Figure 9.

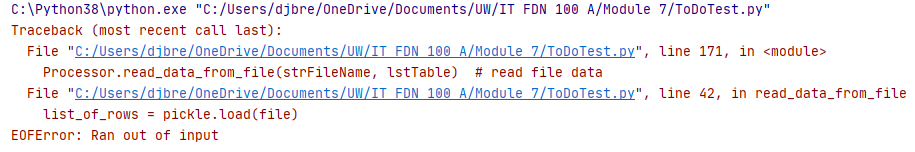


Figure : Another unexpected error

I added an except block for the EOFError and was able to run the first part of the program as shown in Figure 10.

try:  
 file = open(file\_name, **"rb"**)  
 list\_of\_rows = pickle.load(file)  
 file.close()  
except FileNotFoundError:  
 file = open(file\_name, **"wb"**)  
 file.close()  
 print(**"Task list empty."**)  
except EOFError:  
 print(**"No data in file."**)

Figure : Code to read from file

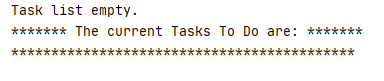


Figure : Result for FileNotFoundError

The message “Task list empty” in Figure 11 was triggered by the FileNotFoundError and tells me that the file had not existed before the script was run. The program continues running anyway and prints the empty list. Checking the folder, I can see that the ToDoFile.dat file has been created (see Figure 12).

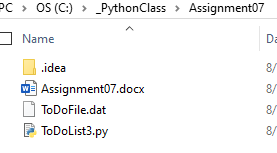


Figure : New file created by program

If I exit and re-run the program, or use the Reload Data from File option before saving any data to the file, I get the result shown in Figure 13.

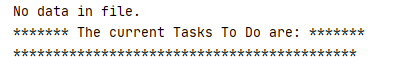


Figure : Result for EOFError

“No data in file” was triggered by the EOFError and tells me that no information has been saved to the file yet. I can verify this by opening the file in Python (see Figure 14).

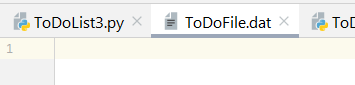


Figure : Empty .dat file

## Writing to the File (pickle.dump)

To write the data to the file, I changed the ‘write\_data\_to\_file’ function to use pickle.dump with the file object instead of file.write (see Figure 15)

file = open(file\_name, **"wb"**)  
pickle.dump(list\_of\_rows, file)  
file.close()

Figure : Code to write data to file

After running the script, I can open the file in Python and see that data exists. Because it has been encoded during the pickle process, it is not very readable, but the data is there (see Figure 16).

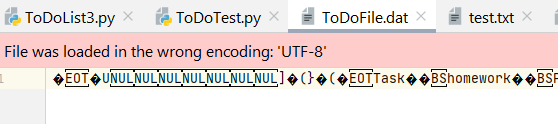


Figure : Data in file

# Running the script

To see whether the code functioned as intended, I first ran it in PyCharm. The data from ToDoFile.dat is printed along with the menu as shown in Figure 17.

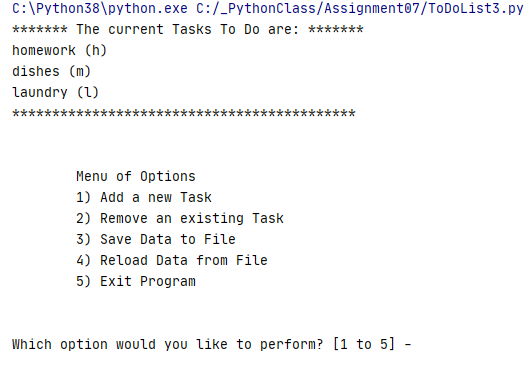


Figure : Show current data - PyCharm

For the most part, the program behaved just as it did in Module 6. When I chose option 2 and removed “dishes”, the program printed the updated list and returned to the menu as shown in Figure 18.

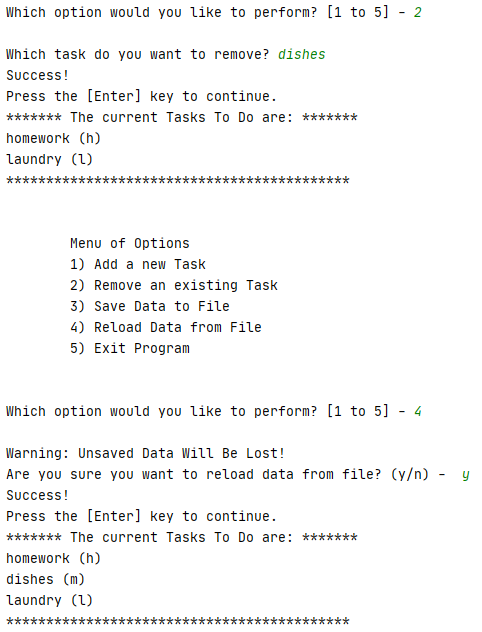


Figure : Removing a task - PyCharm

Next, I entered option 4 to reload the file data, then entered ‘y’ when asked if I wanted to reload the data. The reload function was called and the data unpickled to print the list of chores as shown in Figure 19.

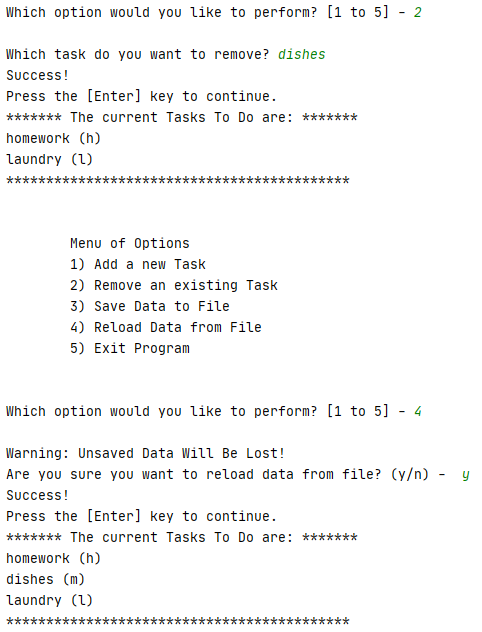


Figure : Reload data from file - PyCharm

Finally, I ran the program from the command window. I added a task and saved the data to the file using pickle.dump as shown in Figure 20.

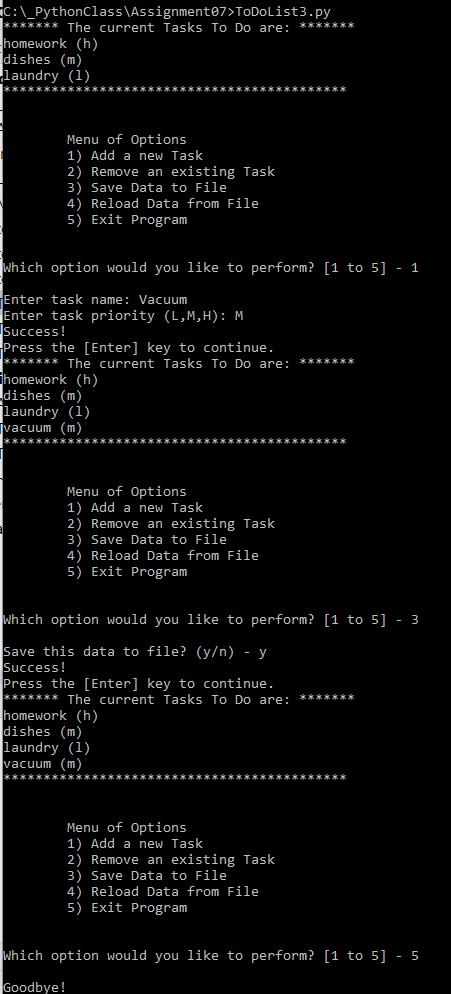


Figure : Run program – cmd

# Summary

Using the textbook, some additional websites, and the Module 7 documentation, I created the ‘To Do List3’ program using exception handling and pickling, and successfully ran it in PyCharm and the OS command window.

# Prior websites

<https://www.learnpython.org/en/>

Trevor Payne youtube

ComputerScienceUK youtube

<https://www.afterhoursprogramming.com/tutorial/>

<https://www.tutorialspoint.com/python/index.htm>

CoderJourney youtube

<https://www.youtube.com/playlist?list=PL6gx4Cwl9DGAcbMi1sH6oAMk4JHw91mC_>

thenewboston youtube

# Good Websites

<https://docs.python.org/3/library/pickle.html> – straight from the Python site

<https://wiki.python.org/moin/UsingPickle>

<https://realpython.com/python-pickle-module/>

compare pickle to marshal and JSON, protocol levels and how to identify