Kasey Clarke

2/28/2023

Foundations of Programming (Python)

Assignment07

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

Assignment 7

Introduction

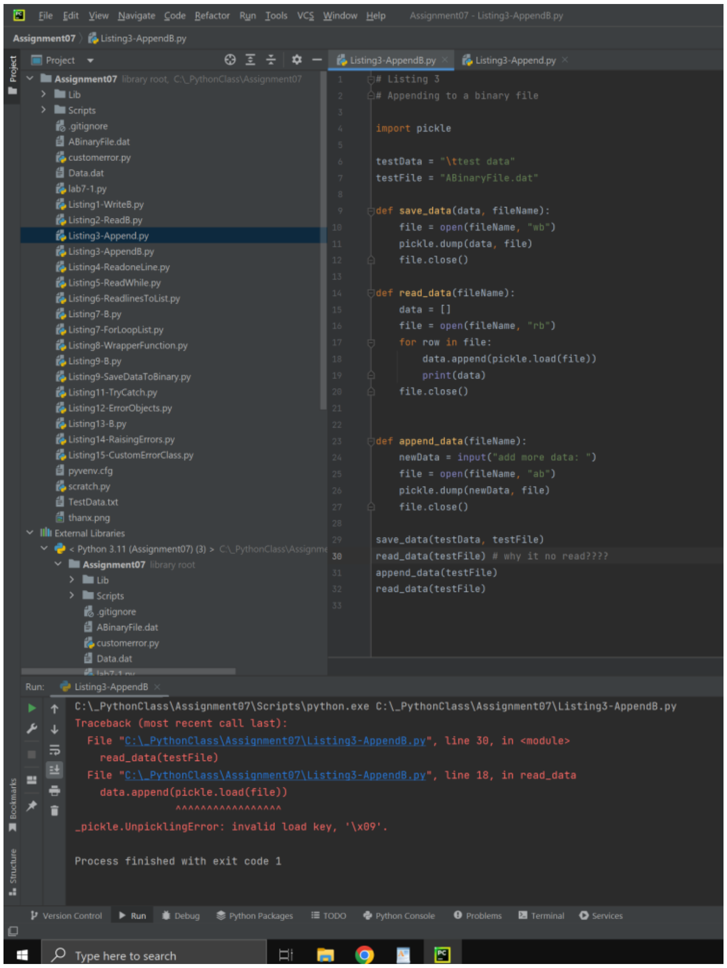
This document outlines the steps I took to learn about how Python deals with Binary Files and Exceptions.

Previewing the Class Material

I started by watching the Mod07 YouTube video, reading the corresponding chapter from the assigned book and watching the Panopto recording for the week. I go through the material in one shot to get an overview of the topics covered in the assignment for the week. After Getting an idea of the topics I download the class materials and prepared to dive in.

Diving In

I wrote in my planner a checklist of the listings in the Panopto recording with the intention of writing them with binary files instead of text files. The first two listings were simple to write with binary instead of text files. Attempting the third listing I ran into a problem because the function to read objects from a binary file only reads one line at a time and the third listing required more than one line to be read from a file. I felt this a good place to start researching to see what solutions could be found on the internet for my problem with listing 3.



```
1  # Listing 3
2  # Appending to a binary file
3
4  import pickle
5
6  testData = "\ttest data"
7  testFile = "ABinaryFile.dat"
8
9  def save_data(data, fileName):
10     file = open(fileName, "wb")
11     pickle.dump(data, file)
12     file.close()
13
14 def read_data(fileName):
15     data = []
16     file = open(fileName, "rb")
17     for row in file:
18         data.append(pickle.load(file))
19         print(data)
20     file.close()
21
22
23 def append_data(fileName):
24     newData = input("add more data: ")
25     file = open(fileName, "ab")
26     pickle.dump(newData, file)
27     file.close()
28
29 save_data(testData, testFile)
30 read_data(testFile) # why it no read????
31 append_data(testFile)
32 read_data(testFile)
33
```

```
C:\_PythonClass\Assignment07\Scripts\python.exe C:\_PythonClass\Assignment07\Listing3-AppendB.py
Traceback (most recent call last):
  File "C:\_PythonClass\Assignment07\Listing3-AppendB.py", line 30, in <module>
    read_data(testFile)
  File "C:\_PythonClass\Assignment07\Listing3-AppendB.py", line 18, in read_data
    data.append(pickle.load(file))
                ^^^^^^^^^^^^^^^^^
_pickle.UnpicklingError: invalid load key, '\x09'.

Process finished with exit code 1
```

Figure 1. My first attempt to read multiple lines of a binary file (line 30) feels like it *should* work with a for-loop and the keyword “in” (line 17) but it simply does not.

Researching

I began searching the internet for terms that I was using in my code. I was looking for keywords and phrases such as “pickling”, “reading a binary file in python”, “pickle.load()” and “for-loop” in various combinations for hours. I found material that was at best tangential to what I needed, and I was burning through time and becoming stressed.

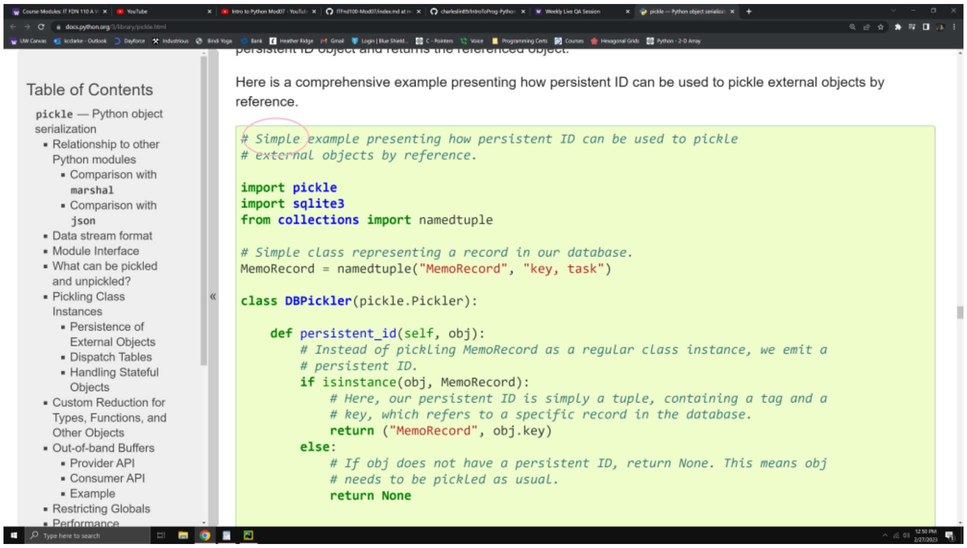


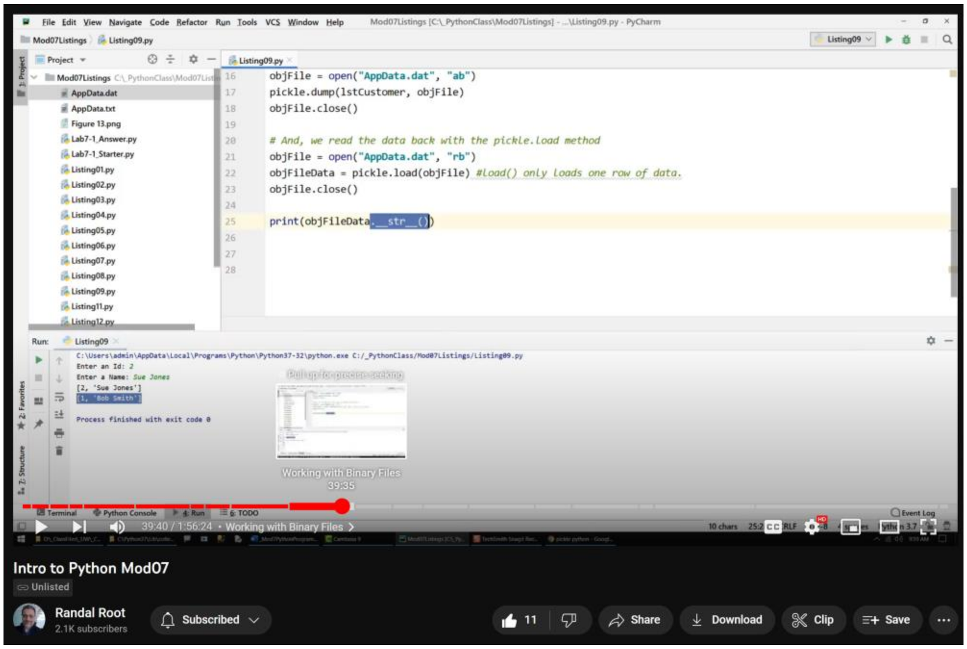
Table of Contents
pickle — Python object serialization
Relationship to other Python modules
Comparison with marshal
Comparison with json
Data stream format
Module Interface
What can be pickled and unpickled?
Pickling Class Instances
Persistent of External Objects
Dispatch Tables
Handling Stateful Objects
Custom Reduction for Types, Functions, and Other Objects
Out-of-band Buffers
Provider API
Consumer API
Example
Restricting Globals
Performance

persistent ID object and returns the referenced object.

Here is a comprehensive example presenting how persistent ID can be used to pickle external objects by reference.

```
# Simple example presenting how persistent ID can be used to pickle
# external objects by reference.

import pickle
import sqlite3
from collections import namedtuple

# Simple class representing a record in our database.
MemoRecord = namedtuple("MemoRecord", "key, task")

class DBPickler(pickle.Pickler):

    def persistent_id(self, obj):
        # Instead of pickling MemoRecord as a regular class instance, we emit a
        # persistent ID.
        if isinstance(obj, MemoRecord):
            # Here, our persistent ID is simply a tuple, containing a tag and a
            # key, which refers to a specific record in the database.
            return ("MemoRecord", obj.key)
        else:
            # If obj does not have a persistent ID, return None. This means obj
            # needs to be pickled as usual.
            return None
```

Figure 2. Most* of the examples I found when I was researching felt above my level of comprehension or irrelevant. *Most because I am sure that after a while, I was in a dazed state of clicking links and closing tabs on my internet browser.



```
objFile = open("AppData.dat", "ab")
pickle.dump(lstCustomer, objFile)
objFile.close()

# And, we read the data back with the pickle.load method
objFile = open("AppData.dat", "rb")
objFileData = pickle.load(objFile) #load() only loads one row of data.
objFile.close()

print(objFileData.__str__())
```

```
C:\Users\admin\AppData\Local\Programs\Python\Python37-32\python.exe C:/_PythonClass/Mod07Listings/Listing09.py
Enter an Id: 2
Enter a Name: Sue Jones
[2, 'Sue Jones']
[1, 'Bob Smith']

Process finished with exit code 0
```

Figure 3. The timestamp of the concept I was stuck on (39:40).

The problem I was having with Listing 3 is outlined in this week's lecture video, so I wanted to make sure I was able to get it, so I kept trying.

The problem

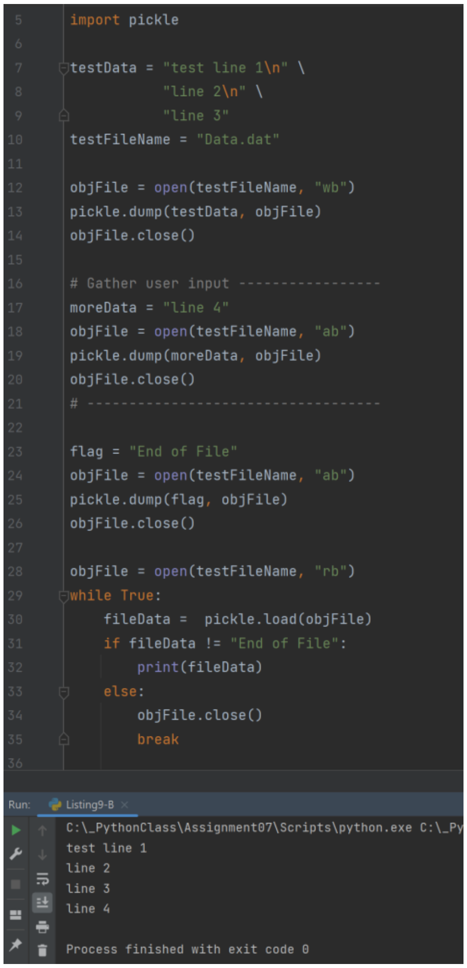
I want to be able to read a binary file of unknown length and print it to the console. I did not know the syntax to do this with a Python for-loop and the “in” keyword and was stuck in a painful trial and error cycle wasting tons of time.

Eventually I stopped trying to jam python’s pickle.load() function into for-loops and tried a while-loop. I thought that if I have a while loop that checks for an empty string at the end of a file, I might be able to read a binary file with any number of lines in it. It worked! I was able to read a binary file with any number of lines, but it generated an exception saying that it was out of data to read. I was so excited that it worked I tried to fix the error and forgot to document it.

Dealing With the Error

I had not yet made it to typing the listings from the Panopto recording that detailed exception handling and I thought I could work through the problem without the “try & except” structure.

I could append a flag variable to the end of my file and have the while-loop and load() check for the flag to know when to stop reading the file. With my handy flag any number of lines of data could be written to a file and I could make the flag the last thing in the file.



```
5   import pickle
6
7   testData = "test line 1\n" \
8              "line 2\n" \
9              "line 3"
10  testFileName = "Data.dat"
11
12  objFile = open(testFileName, "wb")
13  pickle.dump(testData, objFile)
14  objFile.close()
15
16  # Gather user input ----------------
17  moreData = "line 4"
18  objFile = open(testFileName, "ab")
19  pickle.dump(moreData, objFile)
20  objFile.close()
21  # ----------------------------------
22
23  flag = "End of File"
24  objFile = open(testFileName, "ab")
25  pickle.dump(flag, objFile)
26  objFile.close()
27
28  objFile = open(testFileName, "rb")
29  while True:
30      fileData =  pickle.load(objFile)
31      if fileData != "End of File":
32          print(fileData)
33      else:
34          objFile.close()
35          break
36
```

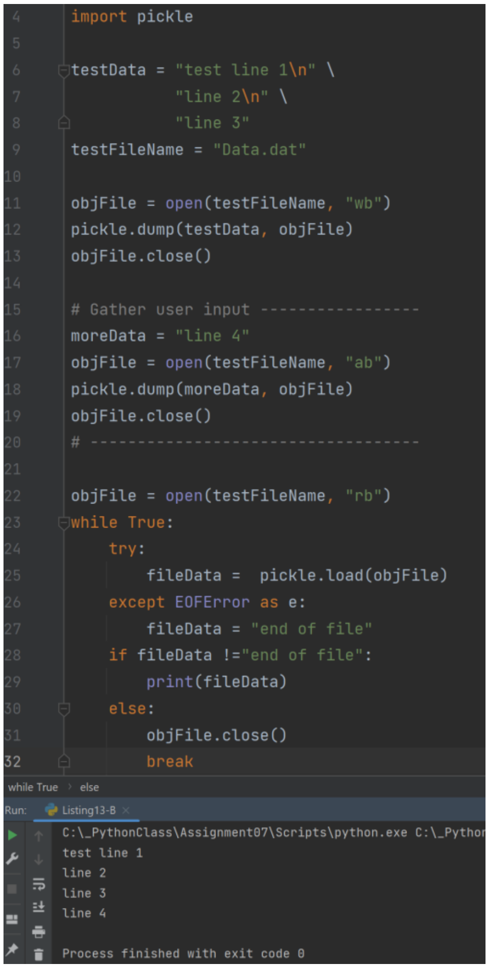
```
C:\_PythonClass\Assignment07\Scripts\python.exe C:\_Py
test line 1
line 2
line 3
line 4

Process finished with exit code 0
```

Figure 4. Listing9-B, a structure to use the pickle module to read any number of lines from a binary file. Substitute lines 16 to 21 with a loop that gathers data from a user and the flag is still the last thing in the file (lines 23 to 26). The user would not see the flag because it is never printed.

Finally, I had actualized my idea and I began writing the remaining listings like a bat out of heck.

Handling the Exception

After working through all the listings, I realized that I could rewrite the code from figure 4 to show the use of the “try & except” structure.



```python
import pickle

testData = "test line 1\n" \
           "line 2\n" \
           "line 3"
testFileName = "Data.dat"

objFile = open(testFileName, "wb")
pickle.dump(testData, objFile)
objFile.close()

# Gather user input -----------------
moreData = "line 4"
objFile = open(testFileName, "ab")
pickle.dump(moreData, objFile)
objFile.close()
# ----------------------------------

objFile = open(testFileName, "rb")
while True:
    try:
        fileData =  pickle.load(objFile)
    except EOFError as e:
        fileData = "end of file"
    if fileData !="end of file":
        print(fileData)
    else:
        objFile.close()
        break
```

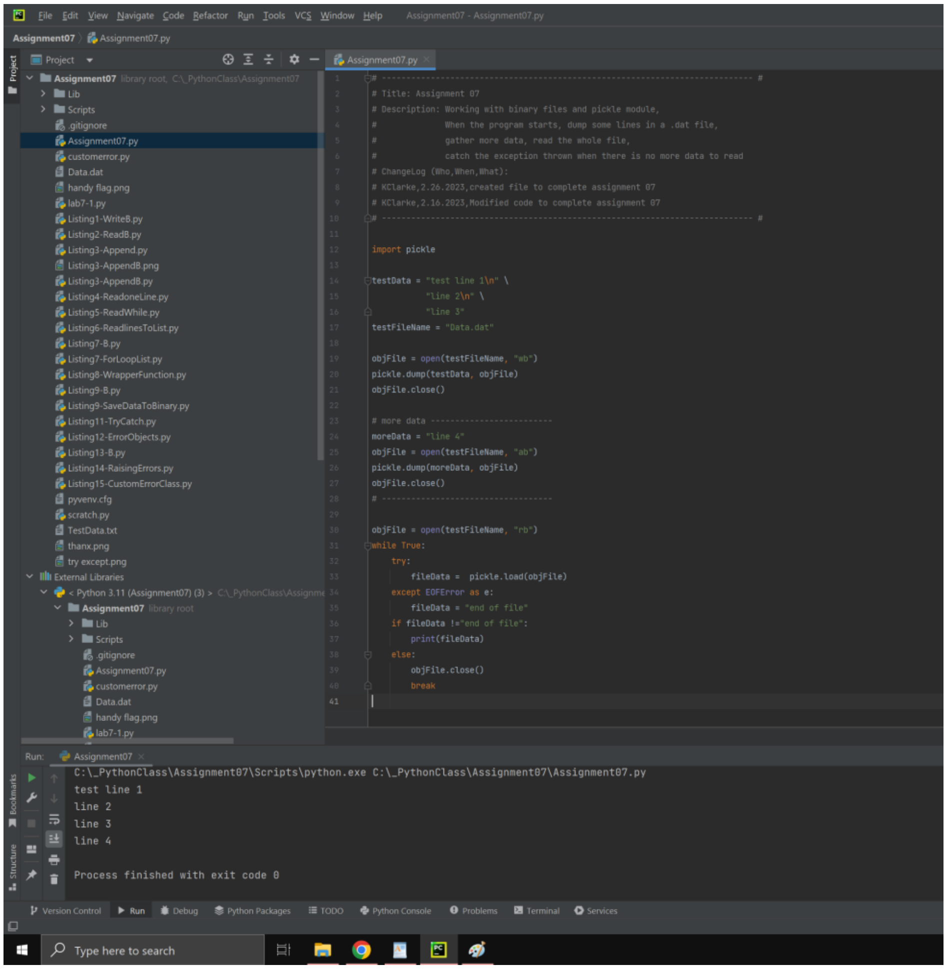
```
C:\_PythonClass\Assignment07\Scripts\python.exe C:\_Pythor
test line 1
line 2
line 3
line 4

Process finished with exit code 0
```

Figure 5. Listing 13-B, a structure with the same net result as in figure 4 using the "try except" block and 3 less lines of code.

Conclusion

I am trying something new, in the earlier 6 assignments I have avoided the first-person pronoun. The first three example assignments used “I” and the fourth did not. I was aiming at imitating “Writing example 4.pdf” because it seemed the most professional to me, but it seemed to be costing me points (precious points!). The point may not be the points but the lesson I learned.

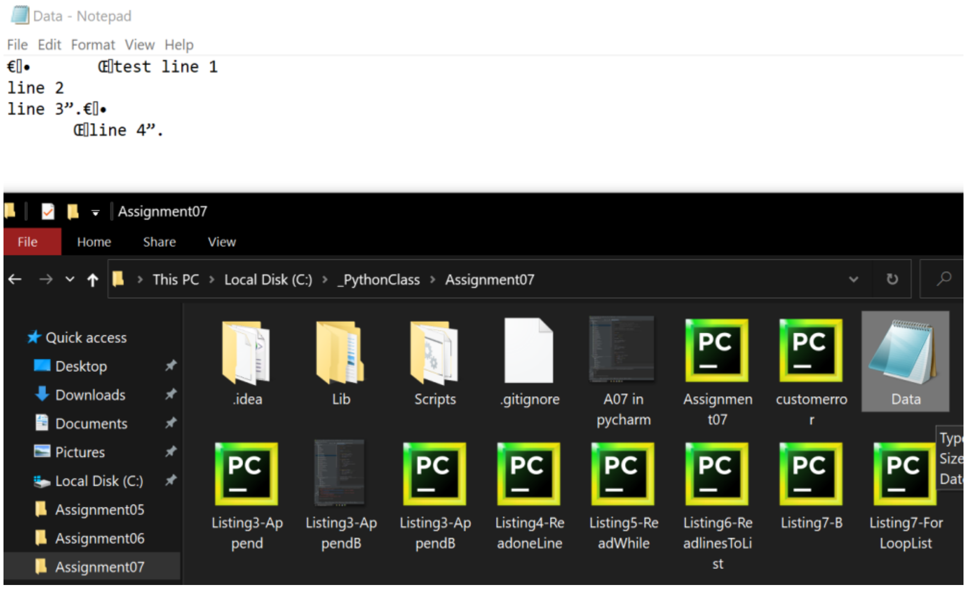
The most important thing I learned in this module is that the solution to the specific problem I have may not be published on the internet. The material in the lectures was sufficient for me to solve my problem but I may not have realized that had I not stress-surfed the internet for hours.



```
# ----------------------------------------------------------------------------- #
# Title: Assignment 07
# Description: Working with binary files and pickle module,
#              When the program starts, dump some lines in a .dat file,
#              gather more data, read the whole file,
#              catch the exception thrown when there is no more data to read
# ChangeLog (Who,When,What):
# KClarke,2.26.2023,created file to complete assignment 07
# KClarke,2.16.2023,Modified code to complete assignment 07
# ----------------------------------------------------------------------------- #


import pickle

testData = "test line 1\n" \
           "line 2\n" \
           "line 3"
testFileName = "Data.dat"

objFile = open(testFileName, "wb")
pickle.dump(testData, objFile)
objFile.close()

# more data -------------------------
moreData = "line 4"
objFile = open(testFileName, "ab")
pickle.dump(moreData, objFile)
objFile.close()
# -----------------------------------


objFile = open(testFileName, "rb")
while True:
    try:
        fileData =  pickle.load(objFile)
    except EOFError as e:
        fileData = "end of file"
    if fileData !="end of file":
        print(fileData)
    else:
        objFile.close()
        break

```

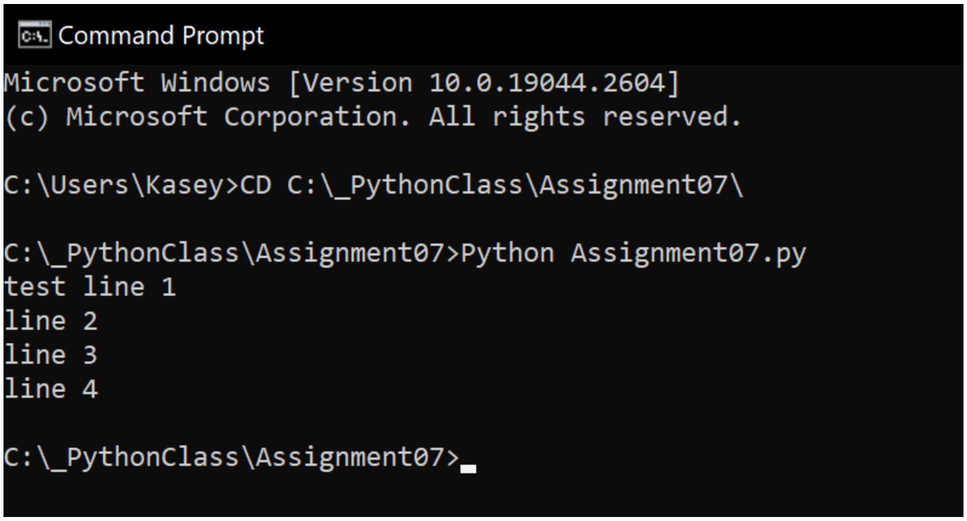
```
C:\_PythonClass\Assignment07\Scripts\python.exe C:\_PythonClass\Assignment07\Assignment07.py
test line 1
line 2
line 3
line 4

Process finished with exit code 0
```

Figure 6. Assignment 7 in PyCharm.



```
€▯•       Œ▯test line 1
line 2
line 3”.€▯•
      Œ▯line 4”.
```

Figure 7. Binary file used in assignment 7.



```
Microsoft Windows [Version 10.0.19044.2604]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Kasey>CD C:\_PythonClass\Assignment07\

C:\_PythonClass\Assignment07>Python Assignment07.py
test line 1
line 2
line 3
line 4

C:\_PythonClass\Assignment07>_
```

Figure 8. Assignment 7 in the command window.