Irina Zubova

March 1, 2021

Foundations of Programming: Python

Assignment 07

https://github.com/i-zuzu/IntroToProg-Python-Mod07

Exception Handling and Pickling in Python

Introduction

In this document I'll describe how I worked on Assignment07. I'll add some links from the web that describe how Pickling and Exception Handling work in Python. I'll explain why I liked them and how they helped me to learn and understand new things.

Then I'll create 2 new scripts that demonstrate how Pickling and Structured error handling work and explain what these scripts do.

Web Research on Exception Handling and Python

For Exception handling I found the following webpages really helpful:

- https://www.w3schools.com/python/python_try_except.asp (external site)
- https://www.freecodecamp.org/news/exception-handling-python/ (external site)
- https://www.tutorialspoint.com/python/python_exceptions.htm (external site)

The first one is relatively short, it has many examples that helped me to understand how Exception works. I like starting from short, simple explanations and then build up more content on them. Then the next 2 links provide more details and descriptions. They outline at the beginning what they cover, also the third one has behind the scene diagrams showing how the program works.

For Pickling I found these webpages really valuable:

- https://www.pitt.edu/~naraehan/python3/pickling.html (external site)
- https://stackabuse.com/introduction-to-the-python-pickle-module/ (external site)

Again, the first link didn't have a lot, but it helped to understand fast the concept of pickling. Then the next link gives much more details and theory. I really liked it gave some explanations from a developer's perspective.

Structured Error Demonstration Script

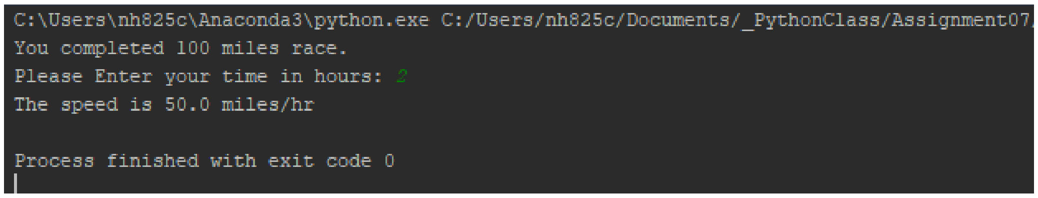
I wanted to show in one script several aspects of error handling with the help of try-except block . My simple script that calculates the speed demonstrates catching several Specific exceptions and Custom errors.
I created new Assignment 07 project in PyCharm and the new Python file called Assignment07_SrtructuredError.py.

Below is the simple code I wrote to test and find the name of errors that cause the program end abruptly:

```
print("You completed 100 miles race.")
distance = 100
time = int(input("Please enter your time in hours: "))
speed = 100/time
print("The speed is", str(speed),  "miles/hr")
```

In my script distance of the race = 100 miles. The program asks the user to enter time spent to finish the race in hours and then calculates the speed (speed =distance/time). And displays the result.

Figure1 below shows the output of the initial script.



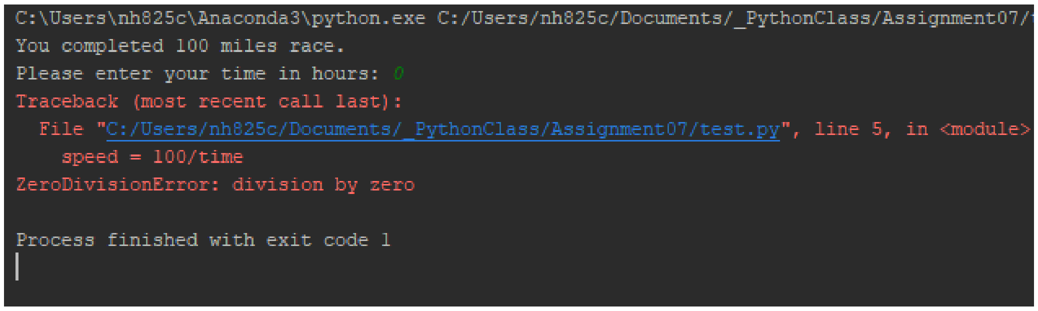
```
C:\Users\nh825c\Anaconda3\python.exe C:/Users/nh825c/Documents/_PythonClass/Assignment07
You completed 100 miles race.
Please Enter your time in hours: 2
The speed is 50.0 miles/hr

Process finished with exit code 0
```

Figure1. Result after the initial script runs in PyCharm

I identified 2 errors that could cause the error and the end of the program:

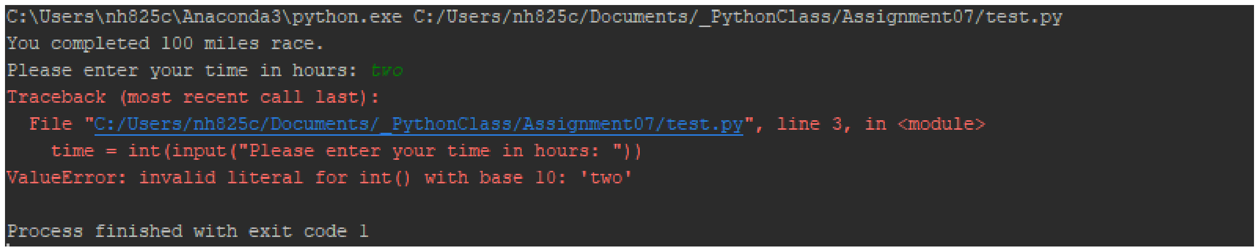
1. Division by zero (if the user enters 0 for time)
2. Unconvertible String to integer conversion (if the user enters types in “two” instead of “2” for time, it’s impossible to convert “two” to integer or float to perform calculations).

I did test to initiate these errors and find the name of these errors. Figures 2 and 3 show the results.



```
C:\Users\nh825c\Anaconda3\python.exe C:/Users/nh825c/Documents/_PythonClass/Assignment07/
You completed 100 miles race.
Please enter your time in hours: 0
Traceback (most recent call last):
  File "C:/Users/nh825c/Documents/_PythonClass/Assignment07/test.py", line 5, in <module>
    speed = 100/time
ZeroDivisionError: division by zero

Process finished with exit code 1
```

Figure2. ZeroDivisionError (Error1)



```
C:\Users\nh825c\Anaconda3\python.exe C:/Users/nh825c/Documents/_PythonClass/Assignment07/test.py
You completed 100 miles race.
Please enter your time in hours: two
Traceback (most recent call last):
  File "C:/Users/nh825c/Documents/_PythonClass/Assignment07/test.py", line 3, in <module>
    time = int(input("Please enter your time in hours: "))
ValueError: invalid literal for int() with base 10: 'two'

Process finished with exit code 1
```

Figure3. Value Error (Error2)

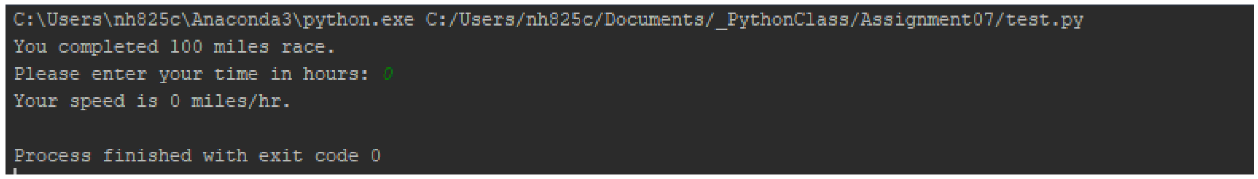
To handle these errors I applied `try` statement with an `except` clause.
Here's the one to handle the Structured Error Demonstration Script:

```
try:
    speed = 100/time
except ZeroDivisionError:
    print("You didn't race.")
else:
    print("The speed is", str(speed),  "miles/hr")
```

By using a `try` statement, I sectioned off the code that could potentially raise an exception. Then, I wrote an `except` clause with a block of statements that are executed only if an exception is raised. Since I specified the name of the error after try statement, this exception is only going to raise if the user enters '0' for time.

If the call to calculate speed raises an exception (as a result of the user entering 0 for time), the exception is caught and the user is informed that You didn't race. If no exception is raised, time string entered by the user converts to number and the program skips the except clause, continuing with the rest of the code.

Figure 4 below shows the script output when the Zero Division Exception raised.



```
C:\Users\nh825c\Anaconda3\python.exe C:/Users/nh825c/Documents/_PythonClass/Assignment07/test.py
You completed 100 miles race.
Please enter your time in hours: 0
Your speed is 0 miles/hr.

Process finished with exit code 0
```

Figure4. Code output when ZeroDivision Exception raised.

Python allows to catch multiple exceptions with multiple `except` clauses. You can list as many `except` clauses as you like. So I added another one to catch the Value Error (in case the user enters unconvertible to integer string):

```
     try:
        time = int(input("Please enter your time in hours: "))
        speed = round(100/time)
    except ZeroDivisionError:
        print("You didn't race.")
    except ValueError:
        print("This was not a number.")
    else:
        print("The speed is", str(speed),  "miles/hr")
```

I also added a single `else` clause after my 2 `except` clauses in a try statement. The else block executes only if no exception is raised in the `try` block. So if none of the exceptions raised the user will see the message with the calculated speed.

Python also allows to raise errors based on custom condition. I added demonstration of this to my script. My custom condition is that the time must be greater than 0 and less than 1:

```
time = float(strInput)
if 1 > time > 0 or time < 0:
    raise Exception('Time evolved should not be less than 1 hr')
speed = round(100/time)
```

And the custom Exception that will be raised:

```
except Exception as e:           #custom exception
    print('Seems like a wrong number. Try again.')
    print (e)
```

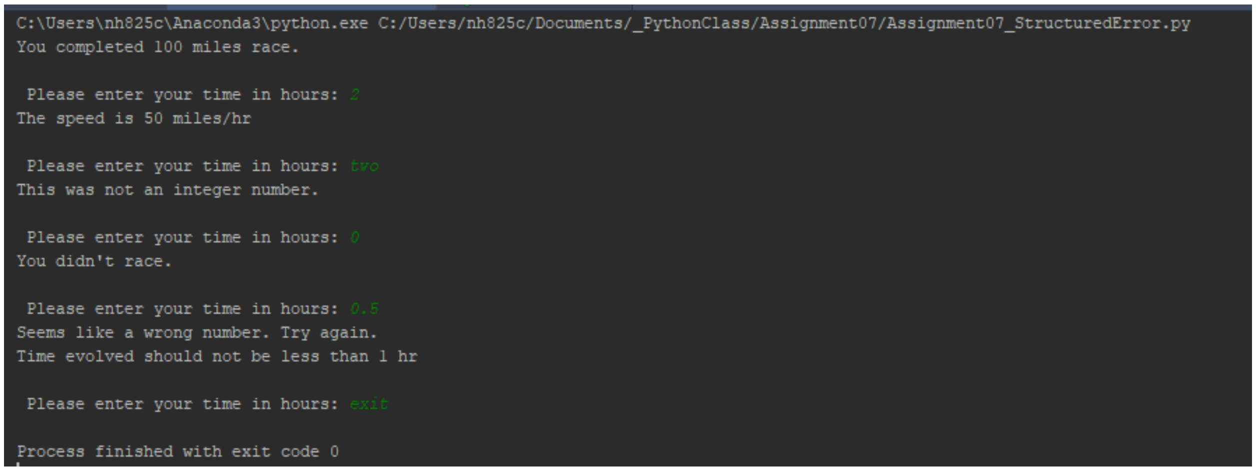
So for example if the user enters “-7” or “0.5” he’ll get 2 messages that something went wrong and that the time should not be less than 1 hour.

The final code to demonstrate Structured Error is below. I added while loop to allow user to test multiple exceptions without restarting the program.

```
# ------------------------------------------------------------------------ #
# Title: Assignment 07 Structured Error (Demo)
# Description: This script calculates the speed and
#              demonstrates how Structured error handling works
# ChangeLog (Who,When,What):
# Irina Zubova,2-27-2020, Created the script
# ------------------------------------------------------------------------ #

#---Data---

distance = 100 # total distance of the race
strInput = ""  # time to finish the race

#Input/Output and Processing---

print("You completed 100 miles race.")
while True:
    try:
        strInput = input("\n Please enter your time in hours or ‘exit’ to stop: ")
        if strInput == 'exit':
            break
        time = float(strInput)
        if 1 > time > 0 or time < 0:
            raise Exception('Time evolved should not be less than 1 hr')
        speed = round(100/time)
    except ZeroDivisionError:       #2nd specific exception
        print("You didn't race.")
    except ValueError:              #1st specific exception
        print("This was not an integer number.")
    except Exception as e:          #custom exception
        print('Seems like a wrong number. Try again.')
        print (e)
    else:                           #no exception
        print("The speed is", str(speed),  "miles/hr")
```

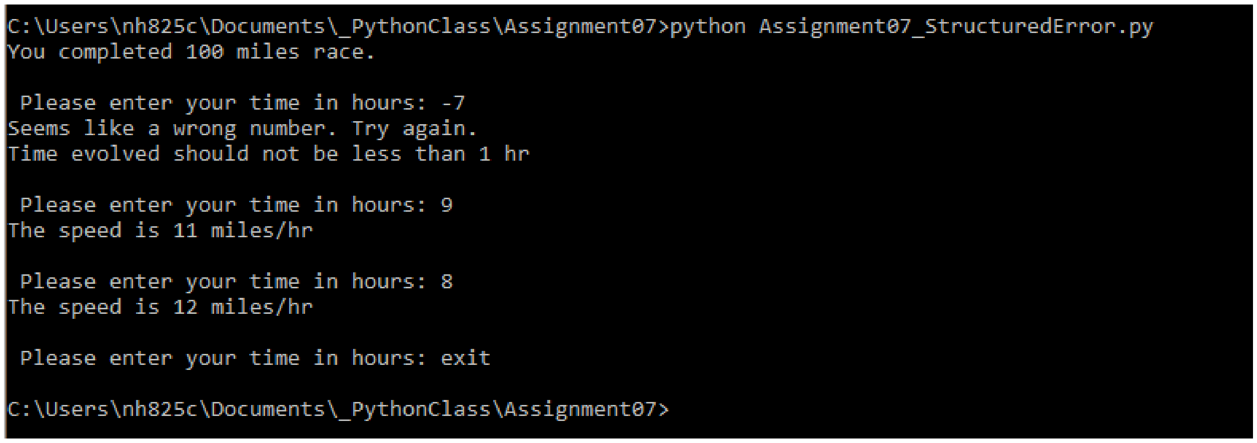
Of course these code could have been improved with adding the separation of concerns and functions. I thought the main purpose of this script was to demonstrate Error Handling in a simple way and the way the code is organized now serves this purpose well.

The next Figure shows the code running in PyCharm:



```
C:\Users\nh825c\Anaconda3\python.exe C:/Users/nh825c/Documents/_PythonClass/Assignment07/Assignment07_StructuredError.py
You completed 100 miles race.

 Please enter your time in hours: 2
The speed is 50 miles/hr

 Please enter your time in hours: two
This was not an integer number.

 Please enter your time in hours: 0
You didn't race.

 Please enter your time in hours: 0.5
Seems like a wrong number. Try again.
Time evolved should not be less than 1 hr

 Please enter your time in hours: exit

Process finished with exit code 0
```

Figure5. Output of Structured Error Demo script in PyCharm



```
C:\Users\nh825c\Documents\_PythonClass\Assignment07>python Assignment07_StructuredError.py
You completed 100 miles race.

 Please enter your time in hours: -7
Seems like a wrong number. Try again.
Time evolved should not be less than 1 hr

 Please enter your time in hours: 9
The speed is 11 miles/hr

 Please enter your time in hours: 8
The speed is 12 miles/hr

 Please enter your time in hours: exit

C:\Users\nh825c\Documents\_PythonClass\Assignment07>
```

Figure6. Output of Structured Error Demo script in Terminal

Pickling Demonstration Script

This script demonstrates how to pickle, unpickle and rewrite pickled data. It also has the example of another option to handle multiple exception times by listing the in a single except clause. The script will allow the user to create a Birthday wihslist [], add some items to this list and then perform pickle features.

My script will be pickling list [], but Python allows to pickle different Types of objects.

I import pickle module at the beginning:

```
import pickle
```

Then the script offers the user Menu of choice to choose the pickling feature to test and manage the wishlist.

```
print("\n Your birthday is coming soon and you need to manage your
wishlist.")

while(True):
    print("""
    Menu of Options
    1) Show current wishlist (pickled data)
    2) Add new data to your wishlist (pickled list)
    3) Rewrite wishlist (current pickled data will be deleted)
    4) Exit Program
    """)
    strChoice = str(input("Which option would you like to perform? [1 to 4] -
"))
    print()  # adding a new line for looks
```

If the user chooses option 1 then the program opens binary `"BirthdayWishList.dat"` file and unpickles the data using function the `load()` function. Then it displays the content of the file to the user:

```
objFile = open("BirthdayWishList.dat", "rb")
lstWishitem = pickle.load(objFile)
print("Here's your current wishlist (unpickled): ")
print(lstWishitem)
objFile.close()
```

I also called the exception to handle 2 types of error: `FileNotFoundError` (if program can't find the file, it doesn't exist) and `EOFError` (there's no data in the file). These 2 errors are listed in a single `except` clause as a comma-separated group enclosed in a set of parentheses. If the program gets to one of these errors the message `"You haven't pickled your Wishlist data yet."` Will be displayed to the user.

Below is a piece of code to execute option 1:

```
if strChoice == '1':
    try:
        objFile = open("BirthdayWishList.dat", "rb")
        lstWishitem = pickle.load(objFile)
        print("Here's your current wishlist (unpickled): ")
        print(lstWishitem)
        objFile.close()
    except (FileNotFoundError, EOFError):  # exception in case the file with
pickled data doesn't exist
        print("You haven't pickled your Wishlist data yet.")
        continue
```

Now let's look at option 2 that allows user to add data to pickled list. First I open the binary file and load the data. Then I get the input from the user and add it to `lstWishitem` list []. And the program adds this list to a binary file (pickles it) using `dump()` function. And I also added try – except statement to handle the exceptions.

Below is a piece of code to execute option 2:

```
elif strChoice == '2':
    strItem = input("Enter the wish item you want to add to your wishlist: ")
    try:
        objFile = open("BirthdayWishList.dat", "rb")
        lstWishitem = pickle.load(objFile)
        objFile.close()
    except ((FileNotFoundError, EOFError)): # exception in case there were no
pickled data
        lstWishitem = []
        print("I created a list to add your desired gifts to it.")
    lstWishitem.append(strItem)
    objFile = open("BirthdayWishList.dat", "wb")
    pickle.dump(lstWishitem, objFile)
    objFile.close()
```

Now about Option 3 that allows to rewrite pickled data.

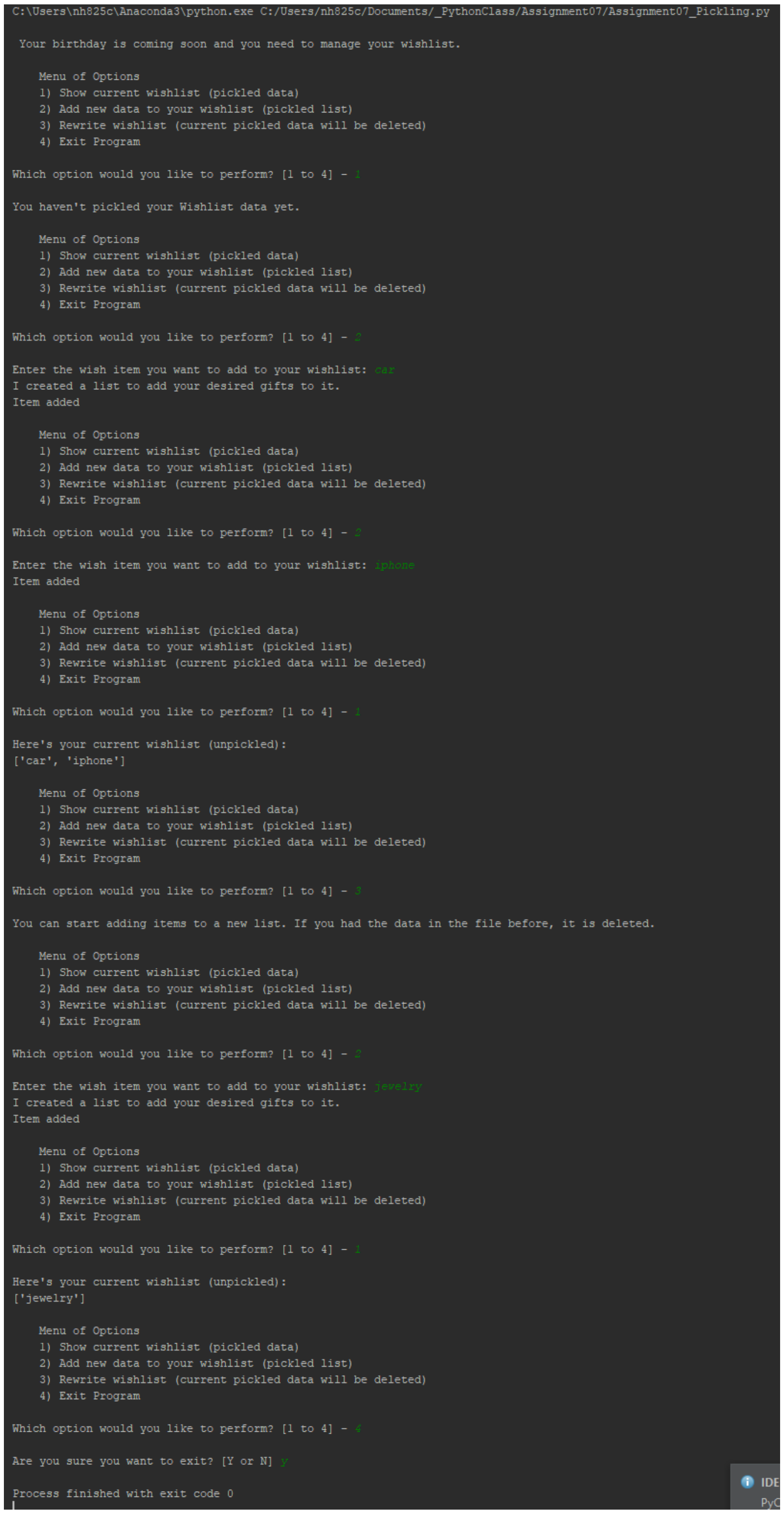
If the user chooses Option 3, then the new binary file opened in write binary mode (“`wb`”).If the file exists, its content will be overwritten, if the file doesn’t exist it will be created. And the user can add data to the file using option 2 from the menu. Below is the piece of code for option 3:

```
elif strChoice == '3':
    objFile = open("BirthdayWishList.dat", "wb")
    objFile.close()
    print("You can start adding items to a new list. If you had the data in
the file before, it is deleted. ")
```

Below is my Pickle Demonstration Script:

```
# ------------------------------------------------------------------------ #
# Title: Assignment 07
# Description: This script demonstrates how Pickling works.
#              It allows you to create a list that contains gifts you'd like
to receive
#              You can also pickle,unpickle and rewrite pickled data in your
Birthday Wishlist.
#
# ChangeLog (Who,When,What):
# Irina Zubova,2-28-2020, Created the script
# ------------------------------------------------------------------------ #

import pickle

# -- Data -- #
# declare variables and constants
strChoice = "" # Captures the user option selection for menu options
objFile = "BirthdayWishlist.dat"   # An object that represents a file
lstWishitem = []
strYorN = "" # Captures the user option selection to exit or continue

print("\n Your birthday is coming up and you need to manage your wishlist.")

while(True):
    print("""
    Menu of Options
    1) Show current wishlist (pickled data)
    2) Add new data to your wishlist (pickled list)
    3) Rewrite wishlist (current pickled data will be deleted)
    4) Exit Program
    """)
    strChoice = str(input("Which option would you like to perform? [1 to 4] -
"))
    print()  # adding a new line for looks

# -- Input/Output and Processing -- #
    # 1.Unpickle and display data
    if strChoice == '1':
        try:
            objFile = open("BirthdayWishList.dat", "rb")
            lstWishitem = pickle.load(objFile)
            print("Here's your current wishlist (unpickled): ")
            print(lstWishitem)
            objFile.close()
        except (FileNotFoundError, EOFError):  # exception in case the file
with pickled data doesn't exist
            print("You haven't pickled your Wishlist data yet.")
            continue

    # 2. Pickle the data
    elif strChoice == '2':
        strItem = input("Enter the wish item you want to add to your
wishlist: ")
        try:
            objFile = open("BirthdayWishList.dat", "rb")
            lstWishitem = pickle.load(objFile)
            objFile.close()
        except ((FileNotFoundError, EOFError)): # exception in case there
were no pickled data
            lstWishitem = []
            print("I created a list to add your desired gifts to it.")
        lstWishitem.append(strItem)
        objFile = open("BirthdayWishList.dat", "wb")
        pickle.dump(lstWishitem, objFile)
        objFile.close()

        print ("Item added")

    # 3. Rewrite Pickled data
    elif strChoice == '3':
        objFile = open("BirthdayWishList.dat", "wb")
        objFile.close()
        print("You can start adding items to a new list. If you had the data
in the file before, it is deleted. ")

    # 4. Exit the Program
    elif strChoice == '4':
        strYorN = input("Are you sure you want to exit? [Y or N] ").upper()
        if strYorN == "Y":
            break  # and Exit the program
    else:
        print(strChoice, "is not in the Menu of Options.")
```

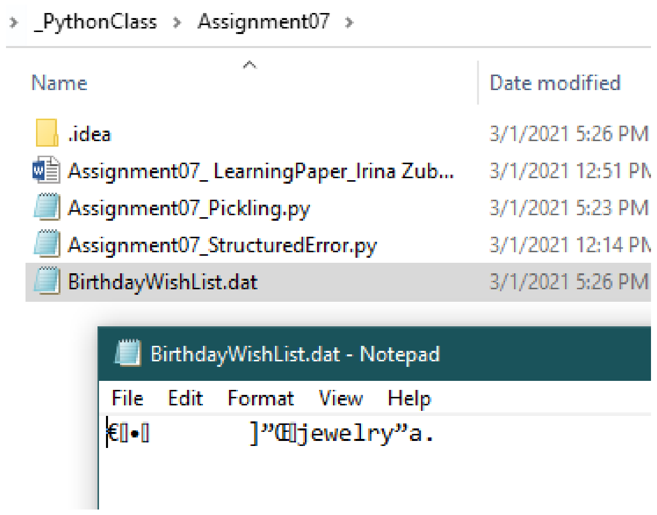
Figure 6 shows the Pickle script running in PyCharm



```
C:\Users\nh825c\Anaconda3\python.exe C:/Users/nh825c/Documents/_PythonClass/Assignment07/Assignment07_Pickling.py

 Your birthday is coming soon and you need to manage your wishlist.

    Menu of Options
    1) Show current wishlist (pickled data)
    2) Add new data to your wishlist (pickled list)
    3) Rewrite wishlist (current pickled data will be deleted)
    4) Exit Program

Which option would you like to perform? [1 to 4] - 1

You haven't pickled your Wishlist data yet.

    Menu of Options
    1) Show current wishlist (pickled data)
    2) Add new data to your wishlist (pickled list)
    3) Rewrite wishlist (current pickled data will be deleted)
    4) Exit Program

Which option would you like to perform? [1 to 4] - 2

Enter the wish item you want to add to your wishlist: car
I created a list to add your desired gifts to it.
Item added

    Menu of Options
    1) Show current wishlist (pickled data)
    2) Add new data to your wishlist (pickled list)
    3) Rewrite wishlist (current pickled data will be deleted)
    4) Exit Program

Which option would you like to perform? [1 to 4] - 2

Enter the wish item you want to add to your wishlist: iphone
Item added

    Menu of Options
    1) Show current wishlist (pickled data)
    2) Add new data to your wishlist (pickled list)
    3) Rewrite wishlist (current pickled data will be deleted)
    4) Exit Program

Which option would you like to perform? [1 to 4] - 1

Here's your current wishlist (unpickled):
['car', 'iphone']

    Menu of Options
    1) Show current wishlist (pickled data)
    2) Add new data to your wishlist (pickled list)
    3) Rewrite wishlist (current pickled data will be deleted)
    4) Exit Program

Which option would you like to perform? [1 to 4] - 3

You can start adding items to a new list. If you had the data in the file before, it is deleted.

    Menu of Options
    1) Show current wishlist (pickled data)
    2) Add new data to your wishlist (pickled list)
    3) Rewrite wishlist (current pickled data will be deleted)
    4) Exit Program

Which option would you like to perform? [1 to 4] - 2

Enter the wish item you want to add to your wishlist: jewelry
I created a list to add your desired gifts to it.
Item added

    Menu of Options
    1) Show current wishlist (pickled data)
    2) Add new data to your wishlist (pickled list)
    3) Rewrite wishlist (current pickled data will be deleted)
    4) Exit Program

Which option would you like to perform? [1 to 4] - 1

Here's your current wishlist (unpickled):
['jewelry']

    Menu of Options
    1) Show current wishlist (pickled data)
    2) Add new data to your wishlist (pickled list)
    3) Rewrite wishlist (current pickled data will be deleted)
    4) Exit Program

Which option would you like to perform? [1 to 4] - 4

Are you sure you want to exit? [Y or N] y

Process finished with exit code 0
```

Figure6. Output of Pickling Demo script in PyCharm

Figure 7 shows the Binary file created.


_PythonClass > Assignment07 >
Name
Date modified

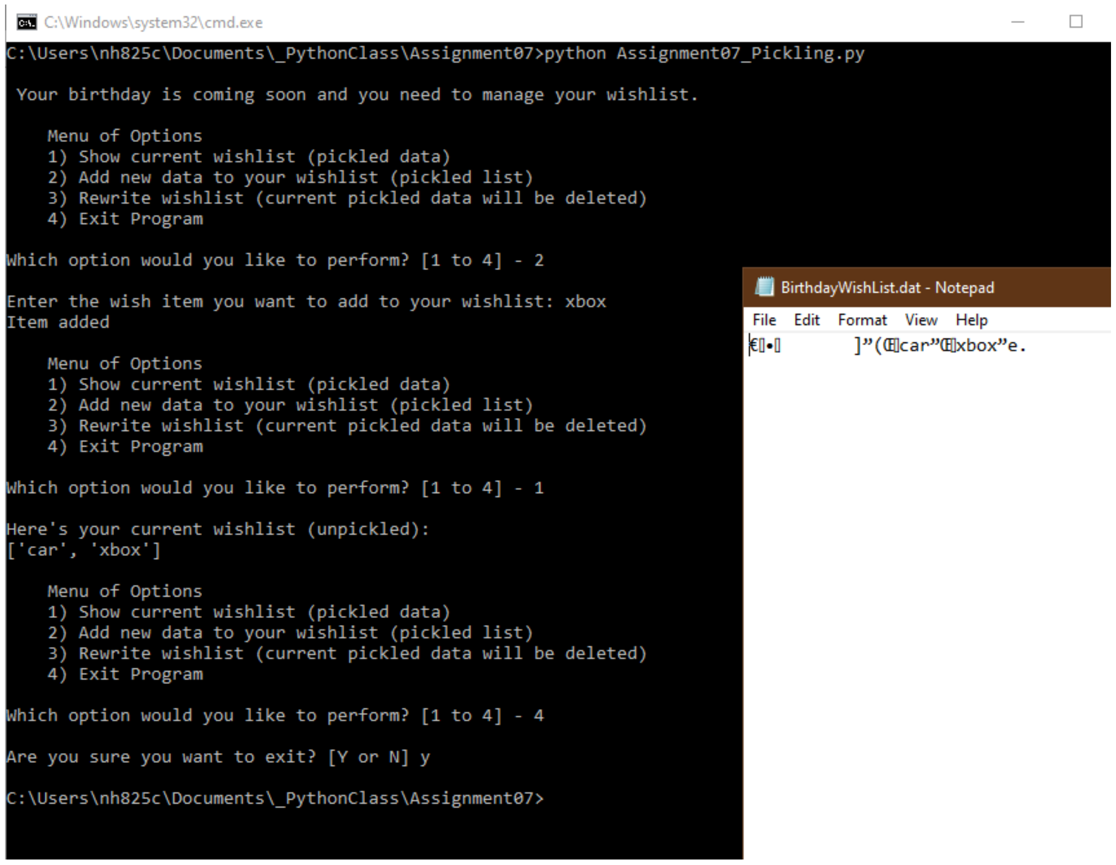
```
.idea                                      3/1/2021 5:26 PM
Assignment07_ LearningPaper_Irina Zub...   3/1/2021 12:51 PM
Assignment07_Pickling.py                   3/1/2021 5:23 PM
Assignment07_StructuredError.py            3/1/2021 12:14 PM
BirthdayWishList.dat                       3/1/2021 5:26 PM
```

BirthdayWishList.dat - Notepad
File Edit Format View Help

```
€•       ]”Œjewelry”a.
```

Figure7. Binary file created by Pickling demo script

The next picture shows pickling file running in Terminal.



```
C:\Users\nh825c\Documents\_PythonClass\Assignment07>python Assignment07_Pickling.py

 Your birthday is coming soon and you need to manage your wishlist.

    Menu of Options
    1) Show current wishlist (pickled data)
    2) Add new data to your wishlist (pickled list)
    3) Rewrite wishlist (current pickled data will be deleted)
    4) Exit Program

Which option would you like to perform? [1 to 4] - 2

Enter the wish item you want to add to your wishlist: xbox
Item added

    Menu of Options
    1) Show current wishlist (pickled data)
    2) Add new data to your wishlist (pickled list)
    3) Rewrite wishlist (current pickled data will be deleted)
    4) Exit Program

Which option would you like to perform? [1 to 4] - 1

Here's your current wishlist (unpickled):
['car', 'xbox']

    Menu of Options
    1) Show current wishlist (pickled data)
    2) Add new data to your wishlist (pickled list)
    3) Rewrite wishlist (current pickled data will be deleted)
    4) Exit Program

Which option would you like to perform? [1 to 4] - 4

Are you sure you want to exit? [Y or N] y

C:\Users\nh825c\Documents\_PythonClass\Assignment07>
```

```
€•       ]”(Œcar”Œxbox”e.
```

Figure8. Pickling demo script running in Terminal and binary file

Summary

In this assignment I practiced in creating and developing my own code. I learned and practiced how to save data to files through pickling and how to manage a pickled objects in a binary file. I got the knowledge on how to handle exceptions raised during the execution of a program and how to trap for specific exceptions and how to write code to deal with them.