Rufus Ayeni
November 20, 2020
IT FDN 110 A
Assignment 07
https://github.com/rayeni/IntroToProg-Python-Mod07

Error Handling and Pickling Demo

Introduction:

In Assignment 07, I researched error handling and picking and applied that research by creating a script that demonstrates the two concepts. I found the following URLs useful in my research because the conveyed the concepts in a clear and concise manner:

The Python pickle Module: How to Persist Objects in Python:
https://realpython.com/python-pickle-module/

Introduction to Python Exceptions:
https://realpython.com/courses/introduction-python-exceptions/

Python Exceptions: An Introduction:
https://realpython.com/python-exceptions/

The script is comprised of three demonstrations:

1. Exception handling demo.
2. Pickling demo.
3. Exception and pickling demo.

The above items are presented to the user as options at the start of the script:

Figure 1

1 Exception Handling Demo:

Bugs are a part of programming. As a programmer, you must anticipate the occurrence of errors and devise methods to catch and handle them. If these errors (*also known as exceptions*) are not anticipated and addressed ahead of time, they can terminate your program abruptly and issue an error message that may not be easily interpreted by the user.

One way to handle exceptions in Python is through the use of the **try** and **except** clauses.

When error occurs in a Python program, if there isn't any exception handling coded into the script, then the program will stop executing and exit abruptly.

1. When **Option #1**, *Exception Handling Demo*, is selected, the demo captures the exception (`FileNotFoundError`), and presents a custom message to the user:

Figure 2

2. The code for this demo starts in the `while` loop on lines 239 - 242. It starts by making a function call to the `Processor.demo_exception_handler()` function, and then to the `IO.input_press_enter_to_continue()` function.

The following is a snippet of the code on lines 239 - 242, in **Figure 3**:

```
234    # Main
235    while(True):
236        IO.print_menu_options()
237        menu_choice = IO.input_menu_choice()
238
239        if menu_choice.strip() == "1": # Exception Handling Demo
240            Processor.demo_exception_handler(demo_file_1)
241            IO.input_press_enter_to_continue("\nPress the [Enter] key to return to Main Menu: ")
242            continue
```

Figure 3

The code for the two functions are in **Figures 4 and 5**:

```
39        @staticmethod
40        def demo_exception_handler(file_name):
41            """ Demonstrates how exception handling works"""
42
43            # deletes file to demonstrate exception handling
44            if os.path.exists(file_name):
45                os.remove(file_name)
46
47            # exception handler
48            try:
49                open(file_name, "rb")
50            except FileNotFoundError:   # Exception generates following message:
51                print("\n[Custom Exception Message]: File is not present, skipping the reading process...\n")
52                print("***************************************************************************************")
53                print("* The above message is a custom exception that either allows the program to continue *")
54                print("* execution or to exit gracefully, instead of exiting abruptly.                      *")
55                print("***************************************************************************************")
56
```

Figure 4

```
173        @staticmethod
174        def input_press_enter_to_continue(message):
175            input(message)
```

Figure 5

2 Pickling Demo:

The pickling demo converts a list to a data stream and stores the stream on disk.

1. When **Option #2**, *Pickling Demo*, is selected, the user is presented a brief definition of pickling, and is asked to press Enter to continue:

```
Please make your selection [1, 2, 3, or 4]: 2

****************************************************
* Pickling is the process of converting an object to  *
* a stream of bytes (serialization) that can be saved *
* to disk.                                          *
****************************************************

Press [Enter] to continue: |
```

Figure 6

2. After the user presses Enter to continue, he/she is presented with a brief comment about the demo:

```
Press [Enter] to continue:

****************************************************
* In this demo we will create a three element list,   *
* pickle it, and save it to a binary file.          *
****************************************************

Press [Enter] to continue: |
```

Figure 7

3. The script continues with the task of creating a list of three cars:

```
Press [Enter] to continue:


****************************************************
* Let's create a list of three cars.              *
****************************************************

Press [Enter] to continue:
```
Figure 8

4. After entering three cars into the list, the script asks the user to press Enter to pickle the list:

```
Press [Enter] to continue:

Enter the first car (ex. Volt, Prius, etc): Accord
Enter the second car (ex. Volt, Prius, etc): Civic
Enter the third car (ex. Volt, Prius, etc): Sonata

Press [Enter] to pickle the list. |
```
Figure 9

5. After the pressing Enter to pickle the list, the script prints a message to inform the user that the pickled list was saved to a file named PickleDemo.dat. The demo ends with the user pressing Enter to return to the Main Menu:

```
The pickle list was written to PickleDemo.dat. File will appear when demo quits.

Press the [Enter] key to return to Main Menu:
```
Figure 10

6. The code for the pickling demo entails three function calls to input items, add items to a list, and to pickle the list:

```
244    if menu_choice.strip() == "2":   # Pickling Demo
245        demo_item1, demo_item2, demo_item3 = IO.input_demo_list_items()
246        list_demo = Processor.add_data_to_list(demo_item1, demo_item2, demo_item3, list_demo)
247        Processor.pickle_object_to_file(list_demo, demo_pickle_file)
248        IO.input_press_enter_to_continue("\nPress the [Enter] key to return to Main Menu: ")
249        continue
```

Figure 11

7. The code for the functions can be viewed in **Figures 12, 13, and 14**:

def input_demo_list_items()

```
122        @staticmethod
123        def input_demo_list_items():
124            """Gets demo list items from user """
125
126            print("**************************************************************")
127            print("* Pickling is the process of converting an object to   *")
128            print("* a stream of bytes (serialization) that can be saved *")
129            print("* to disk.                                             *")
130            print("**************************************************************")
131
132            # Ask user to press Enter to continue with demo.
133            input("\nPress [Enter] to continue: ")
134
135            print("\n***********************************************************")
136            print("* In this demo we will create a three element list,    *")
137            print("* pickle it, and save it to a binary file.             *")
138            print("**************************************************************")
139
140            # Ask user to press Enter to continue with demo.
141            input("\nPress [Enter] to continue: ")
142
143            print("\n**********************************************************")
144            print("* Let's create a list of three cars.                   *")
145            print("**************************************************************")
146
147            # Ask user to press Enter to continue with demo.
148            input("\nPress [Enter] to continue: ")
149
150            car1 = input("\nEnter the first car (ex. Volt, Prius, etc): ")
151            car2 = input("Enter the second car (ex. Volt, Prius, etc): ")
152            car3 = input("Enter the third car (ex. Volt, Prius, etc): ")
153
154            return car1, car2, car3
```

Figure 12

def add_data_to_list()

```
57          @staticmethod
58          def add_data_to_list(element1, element2, element3, demo_list):
59              """ Adds data to demo_list and returns it """
60
61              demo_list = [element1, element2, element3]
62              return demo_list
```
Figure 13

def pickle_object_to_file()

```
64          @staticmethod
65          def pickle_object_to_file(demo_list, demo_file):
66              """ Pickles list and adds to file """
67
68              input("\nPress [Enter] to pickle the list. ")
69              print()   # whitespace
70              f = open(demo_file, "wb")   # open binary file
71              pickle.dump(demo_list, f)   # pickle list and write to file
72              f.close()   # close file
73              # create a pickle object (only to print to screen in demo)
74              pk_object = pickle.dumps(demo_list)
75              # print confirmation message
76              print(f'The pickle list was written to {demo_file}. File will appear when demo quits. ')
77              return pk_object
```
Figure 14

3 Exception and Pickling Demo:

The exception and pickling demo combines the previous two demos (*exception handling and pickling demos*).

1. When the user selects **Option #3**, *Exception and Pickling Demo*, he/she is presented with a brief overview of the demo:

```
Please make your selection [1, 2, 3, or 4]: 3


**********************************************************
* This demo will cover the use of exception handling,    *
* pickling, and saving data to a binary file.            *
**********************************************************


Press [Enter] to continue:
```
Figure 15

2. When the user presses `Enter` to continue, as seen in the previous demo, he/she is presented with a custom exception message telling him/her that the file is not present for reading. Instead of exiting abruptly, the program continues to run:

```
[Custom Exception Message]: File is not present, skipping the reading process...


**********************************************************************************
* The above message is a custom exception that either allows the program to continue *
* execution or to exit gracefully, instead of exiting abruptly.                      *
**********************************************************************************

Press the [Enter] key to continue: |
```
Figure 16

3. After the user presses `Enter` to continue, the user is informed they will enter two movie ratings (*Movie Title, and Rating*):

```
***************************************************
* Let's enter a two movie ratings.
***************************************************

Press [Enter] to continue:
```
Figure 17

4. After the user enters his movie ratings, he receives confirmation that the items are in a list, and is asked press `Enter` to see the data in both list and table forms:

```
Enter the first movie: Rambo
Enter the first movie's rating (ex. Excellent, Good, Fair, Bad): Good

Enter the second movie: Avengers
Enter the second movie's rating (ex. Excellent, Good, Fair, Bad): Excellent


***********************************************************
* Great. We now have two entries in our list object (table) *
***********************************************************

Press [Enter] to see the list object in list and table forms:
```
Figure 18

5. After the data is presented to the user in list and table forms, the user is asked to press Enter to pickle the list:

```
List Form:

[{'Movie': 'Rambo', 'Rating': 'Good'}, {'Movie': 'Avengers', 'Rating': 'Excellent'}]

Table Form:

*************** Movie Ratings ****************
* Rambo (Good)
* Avengers (Excellent)
*********************************************

***********************************************************
* Next. Let's pickle the list object and save it to a file. *
***********************************************************

Press [Enter] to pickle the list. |
```
Figure 19

6. After the user presses Enter to pickle the list, he is notified that the pickled list can be viewed in a file named MovieRatings.dat:

```
The pickle list was written to MovieRatings.dat. File will appear when demo quits.

***********************************************************
* Before we end the demo, let's see what both a pickled list and *
* non-pickled list look like                               *
***********************************************************

Press [Enter] to continue: |
```
Figure 20

7. After the user presses Enter to continue, both lists (*pickled and non-pickled*) are shown to the user for comparison:

```
Press [Enter] to continue:

Non-pickle list:
[{'Movie': 'Rambo', 'Rating': 'Good'}, {'Movie': 'Avengers', 'Rating': 'Excellent'}]

Pickle list:
b'\x80\x04\x95H\x00\x00\x00\x00\x00\x00\x00]\x94(}\x94(\x8c\x05Movie\x94\x8c\x05Rambo\x9
```
Figure 21

8. The contents of MovieRating.dat can be seen in **Figure 22**:
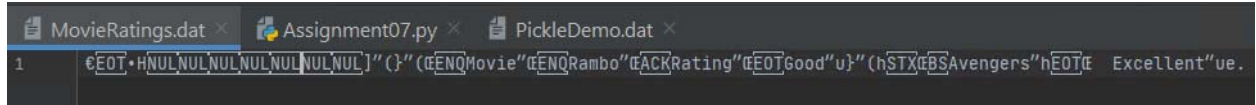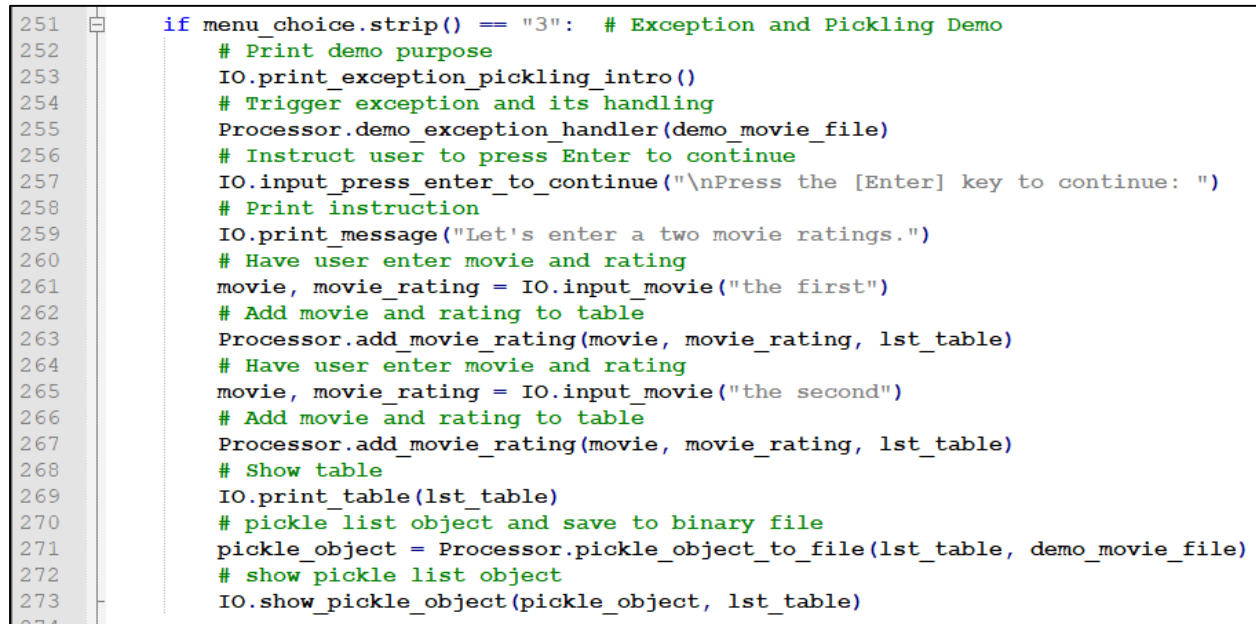

Figure 22

9. The code for exception and pickling demo entails 11 function calls as seen in **Figure 23**:

```
251         if menu_choice.strip() == "3":   # Exception and Pickling Demo
252             # Print demo purpose
253             IO.print_exception_pickling_intro()
254             # Trigger exception and its handling
255             Processor.demo_exception_handler(demo_movie_file)
256             # Instruct user to press Enter to continue
257             IO.input_press_enter_to_continue("\nPress the [Enter] key to continue: ")
258             # Print instruction
259             IO.print_message("Let's enter a two movie ratings.")
260             # Have user enter movie and rating
261             movie, movie_rating = IO.input_movie("the first")
262             # Add movie and rating to table
263             Processor.add_movie_rating(movie, movie_rating, lst_table)
264             # Have user enter movie and rating
265             movie, movie_rating = IO.input_movie("the second")
266             # Add movie and rating to table
267             Processor.add_movie_rating(movie, movie_rating, lst_table)
268             # Show table
269             IO.print_table(lst_table)
270             # pickle list object and save to binary file
271             pickle_object = Processor.pickle_object_to_file(lst_table, demo_movie_file)
272             # show pickle list object
273             IO.show_pickle_object(pickle_object, lst_table)
```

Figure 23

Summary:

In this assignment, I created a Python script that demonstrates exception handling and pickling. Exception handling is an important concept in programming that requires programmers to anticipate and capture errors that users may trigger as a result of faulty programming logic. Pickling is the conversion of a Python object to a byte stream. In other programming languages, this is called serialization. The purpose of serializing Python objects is to lower the time it takes to read and write data to disk.