

**Name:** Mohammed Qadri  
**Date:** 2023-08-25  
**Course:** IT FDN 110 A  
**GitHub URL:** <https://github.com/mqadri22/IntroToProg-Python-Mod07>  
**GitHub Site:** <https://mqadri22.github.io/IntroToProg-Python-Mod07/>

## **Assignment 07 – Files & Exceptions**

### **Introduction**

The objective of Assignment 07 is to gain familiarity with working with files (specifically, binary files) and structured error handling in Python. Unlike previous assignments, Assignment 07 was relatively free form – with the only stipulation being that a script be created demonstrating how pickling (serializing data and saving it in binary format) and structured error handling work.

As the author works in the aerospace industry, it was decided that a script be created that would create, load, pickle (serialize as binary), unpickle (un-serialize from binary), and analyze basic data recorded during a hypothetical airplane flight. The following sections describe the basic structure of the script, how pickling was utilized, and how errors are handled in a structured fashion therein.

### **Basic Code Structure**

As mentioned previously, this assignment was relatively free form, so no starter file was provided. However, the author elected to use a similar basic structure to previous assignments, namely the separation of concerns.

The script was thus broken into four sections: Data declaration, Presentation (Input/Output), Processing (of data), and the Main Body. Data declaration consists of the declaration of variables that are used in the main body and passed in as arguments to methods in other sections.

Presentation (Input/Output) consists of methods to display menus to the user and capture their input in response to the menus; these captured inputs are saved as variables in the main body for use in methods from other sections. A custom class “IO” was created to encompass these methods.

Processing consists of methods to process data either stored in memory or persistently in a file. Methods are included for the “import” (unpickling) and “export” (pickling) of data out of or into binary format, respectively. Also contained in Processing are methods to load data at the outset of the script main body and to display data when prompted by the user. Lastly, another method is included to perform rudimentary analyses on the recorded flight data.

The main body of the script consists of a while loop that performs actions based on user response to displayed menus until the user elects to exit. The subsequent sections cover the four sections of the script in more detail.

## Data Declaration

The Data Declaration portion of the script consists of the declaration of variables that will be used later in the script, primarily in the main body. Despite only being called only in the main body of the script, these functions are there passed as arguments for methods defined in other sections.

Before the variables are declared, however, code is imported from two different sets of files: pickle and os.path. pickle is imported to handle the pickling/unpickling portion of the assignment, whereas os.path enables use of methods later on that pertain to file paths. Figure 1 below shows the importation of these two packages.

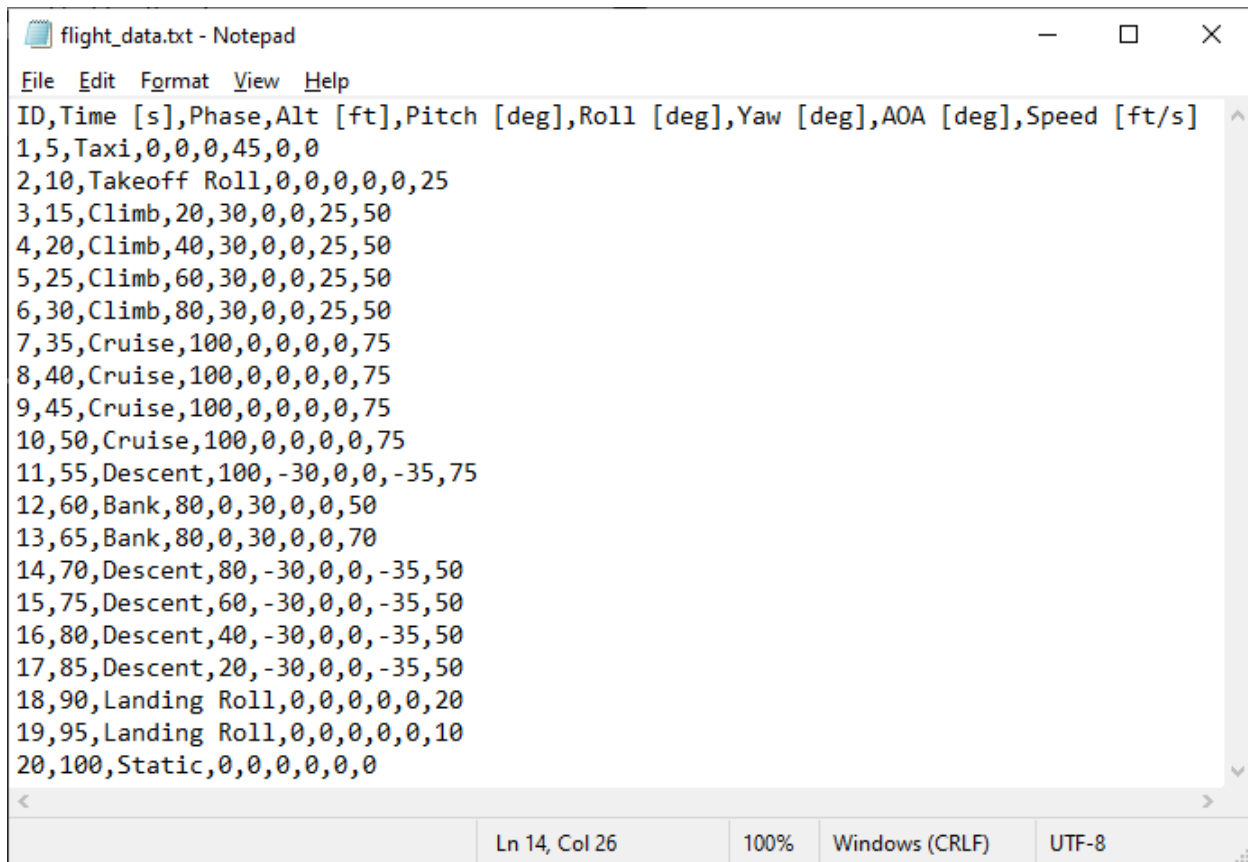
```
9 import pickle
10 import os.path
```

*Figure 1. Importation of code from other files*

Two variables are declared in the Data Declaration section: fileName and binaryFile. fileName is a string consisting of the name of a txt file containing hypothetical flight parameter data captured by the flight recorder on an airplane during a flight. This data is tabulated with the top row being a header describing the parameters quantified in subsequent rows. This tabulated data can be visualized as seen in Table 1 and Figure 2 below, in table and txt forms.

*Table 1. Tabulated flight data format*

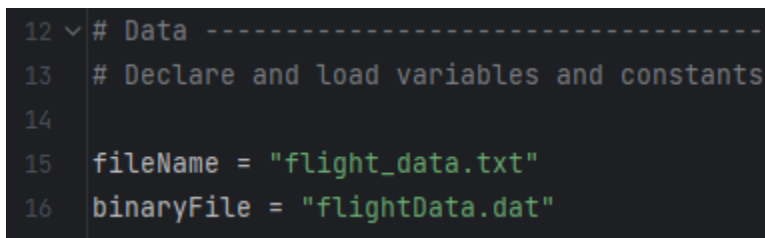
ID	Time [s]	Phase	Alt [ft]	Pitch [deg]	Roll [deg]	Yaw [deg]	AOA [deg]	Speed [ft/s]
⋮								



```
flight_data.txt - Notepad
File Edit Format View Help
ID,Time [s],Phase,Alt [ft],Pitch [deg],Roll [deg],Yaw [deg],AOA [deg],Speed [ft/s]
1,5,Taxi,0,0,0,45,0,0
2,10,Takeoff Roll,0,0,0,0,0,25
3,15,Climb,20,30,0,0,25,50
4,20,Climb,40,30,0,0,25,50
5,25,Climb,60,30,0,0,25,50
6,30,Climb,80,30,0,0,25,50
7,35,Cruise,100,0,0,0,0,75
8,40,Cruise,100,0,0,0,0,75
9,45,Cruise,100,0,0,0,0,75
10,50,Cruise,100,0,0,0,0,75
11,55,Descent,100,-30,0,0,-35,75
12,60,Bank,80,0,30,0,0,50
13,65,Bank,80,0,30,0,0,70
14,70,Descent,80,-30,0,0,-35,50
15,75,Descent,60,-30,0,0,-35,50
16,80,Descent,40,-30,0,0,-35,50
17,85,Descent,20,-30,0,0,-35,50
18,90,Landing Roll,0,0,0,0,0,20
19,95,Landing Roll,0,0,0,0,0,10
20,100,Static,0,0,0,0,0,0
Ln 14, Col 26 100% Windows (CRLF) UTF-8
```

Figure 2. Hypothetical recorded flight data

binaryFile is a string containing the name of a file that the user may elect to import binary data from or export binary data to. The serialization of data into binary and de-serialization from binary are covered in the Processing section of the script. Figure 3 below shows how the two variables were declared.



```
12 # Data -----
13 # Declare and load variables and constants
14
15 fileName = "flight_data.txt"
16 binaryFile = "flightData.dat"
```

Figure 3. Declaration of fileName and binaryFile variables

## Presentation

The presentation portion of the script consists of the “IO” method class. The methods in this class are fairly similar to those in previous assignments, namely Assignments 06 and 05. These methods either present menus to the user or record user responses to these menus. There are two menus in particular that are displayed to the user: the Main Menu and the Performance Analysis menu. The Main Menu provides the user a list of options (import data, export data, analyze data, or exit) whereas the Performance Analysis Menu is a sub-option of the analyze data option where the user can choose what type of analysis to perform on the data. Due to the similarity to previous assignments, these methods

are not discussed here in great detail. Figures 4 and 5 below show the menu display and input retrieval methods for both menus.

```
20 class IO:
21
22     1 usage
23     @staticmethod
24     def main_menu_tasks():
25         """Display a main menu of choices to the user..."""
26
27         print('
28
29             Menu of Options
30             1) Display flight data
31             2) Export flight data
32             3) Import flight data
33             4) Analyze flight performance
34             5) Exit Program
35         ')
36
37         print()
38
39     1 usage
40     @staticmethod
41     def main_menu_choice():
42         """Gets the main menu choice from a user..."""
43
44         choice = str(input("Which option would you like to perform? [1 to 5] - ")).strip()
45         print()
46         return choice
```

Figure 4. Main menu display and user input retrieval methods

```
48 @staticmethod
49 def perf_menu_tasks():
50     """Displays a menu of performance analyses for the user to choose from..."""
51
52     print('
53
54         Performance Analyses
55         1) Calculate flight time
56         2) Calculate maximum climb rate
57         3) Return to Main Menu
58     ')
59
60     print()
61
62     1 usage
63     @staticmethod
64     def perf_menu_choice():
65         """Gets the performance analysis menu choice from a user..."""
66
67         choice = str(input("Which analysis would you like to perform? [1 to 3] - ")).strip()
68         print()
69         return choice
```

Figure 5. Performance analysis menu display and user input retrieval methods

## Processing

The Processing portion of the script consists of the “Processor” class, which contains methods that “process” data. “Processing” here encompasses the reading of data from a text file (i.e., imagined as received from a flight recorder system, though this more realistically would be a binary file, json, or other), generation of placeholder data if no data exists, pickling of data and export to a file, unpickling of binary data from a file and importation into memory, performance of rudimentary analysis on data in memory, and display of data in memory.

The loading of data from a text file has been accomplished in previous assignments, and is accomplished similarly here – with the exception of structured error handling being introduced. This and other instances of structured error handling are covered in the Structured Error Handling section below, but at a high level this instance handles errors that could emerge from a specific file not being found where expected. Note that the list being created is of mixed data types, which aids in the analysis of the data later on. Figure 6 below shows the method that handles the loading of data (as formatted in previous images) from a text file.

```
76 @staticmethod
77 def loadDataFromFile(file_name):
78     """This function loads data from a file into a list table, converting any string numbers to floats..."""
79     try:
80         targetFile = open(file_name, "r")
81         outputData = targetFile.readlines()
82         for idx, row in enumerate(outputData):
83             outputData[idx] = row.split(",")
84             for idx2, elem in enumerate(outputData[idx]):
85                 if elem.isnumeric():
86                     outputData[idx][idx2] = float(elem)
87                 else:
88                     outputData[idx][idx2] = elem.strip()
89             return outputData
90             # print(outputData)
91     except FileNotFoundError:
92         print(file_name, " does not exist. Building empty file... ")
93         outputData = []
94         header = "ID,Time [s],Phase,Alt [ft],Pitch [deg],Roll [deg],Yaw [deg],AOA [deg],Speed [ft/s]"
95         outputData.append(header.split(","))
96         numel = 11
97         for idx in range(1, numel):
98             ID = idx
99             t = 5.0 * idx
100             phase = "Static"
101             params = [0.0] * 6
102             row = [ID, t, phase]
103             row.extend(params)
104             outputData.append(row)
105         return outputData
```

Figure 6. Method to load data from a text file into a mixed-data-type list for processing

Serializing of data into binary format and exporting to a file is accomplished via the exportBinaryData function. This function receives two inputs, the name of a target file and data to be serialized, and then proceeds to serialize the data and then save it to the specified target file. The specific mechanics of the

pickling are discussed in the Pickling & Unpickling section below. Figure 7 below shows the method that handles the serialization of data into binary and its saving into data.

```
111 @staticmethod
112 def exportBinaryData(file_name="exportedData.dat", exportData=[]):
113     """This function pickles list data and exports it to a binary file..."""
119     print("Exporting data to ", file_name, "... \n")
120     targetFile = open(file_name, "ab")
121     pickle.dump(exportData, targetFile)
122     targetFile.close()
123     print("Data successfully exported to ", file_name)
124
```

Figure 7. Method for data serialization and export

De-serialization (i.e., “unpickling”) of binary data and importation into memory is accomplished by the importBinaryData function. This method receives one argument, the name of a binary data file from which to import and de-serialize data from. The details of de-serialization are covered in the Pickling & Unpickling section below. Figure 8 below shows the method through which serialized (“pickled”) data is de-serialized (“unpickled”) and imported back into memory.

```
125 @staticmethod
126 def importBinaryData(file_name):
127     """This function reads in data from a binary file and unpickles it..."""
132     ###
133     if os.path.isfile(file_name):
134         print("Importing data from ", file_name, "... \n")
135         outputData = []
136         targetFile = open(file_name, "rb")
137         outputData = pickle.load(targetFile)
138         targetFile.close()
139         print("Data imported from ", file_name, ". Returning to main menu... \n")
140         return outputData
141     else:
142         print("No binary file named "+file_name+" exists. Returning to main menu... \n")
```

Figure 8. Method for data de-serialization and import

The next method in the Processor class is analyzePerformance, which performs airplane performance analysis on data in list form. The actual analysis performed on the data is rudimentary and not intended to meet the objectives of the assignment. However, structured error handling is incorporated in this method in an attempt to preclude expected errors. This is covered in more detail in the Structured Error Handling section below. Figure 9 below shows analyzePerformance method.

```

144     @staticmethod
145     def analyzePerformance(option, inputData):
146         """
147         """
148         phase_column = None
149         alt_column = None
150         time_column = None
151         for idx, elem in enumerate(inputData[0]):
152             if elem.lower() == "Alt [ft]".lower():
153                 alt_column = idx
154             elif elem.lower() == "Time [s]".lower():
155                 time_column = idx
156             elif elem.lower() == "Phase".lower():
157                 phase_column = idx
158
159         if option == "1":      # Calculate Flight Time
160             takeoff_check = None
161             landing_check = None
162             takeoff_time = None
163             landing_time = None
164             if not(phase_column is None):
165                 for idx, row in enumerate(inputData):
166                     if row[phase_column].lower() == "Takeoff Roll".lower():
167                         takeoff_check = True
168                         takeoff_time = row[time_column]
169                 for idx2, row in enumerate(reversed(inputData)):
170                     if row[phase_column].lower() == "Landing Roll".lower():
171                         landing_check = True
172                         landing_time = row[time_column]
173                 try:
174                     if takeoff_check is None or landing_check is None:
175                         raise Exception
176                     flight_time = landing_time - takeoff_time
177                     print("Flight time = %.2f s" % flight_time)
178                 except:
179                     if takeoff_check is None and landing_check is None:
180                         print("The plane never took off or landed. Flight time = 0 s \n")
181                     elif takeoff_check is None:
182                         print("The plane never took off, but landed. Data unintelligible \n")
183                     elif landing_check is None:
184                         print("The plane took off but never landed. Data unintelligible \n")
185             else:
186                 print("Data is partial or missing. Returning to main menu...")

```

Figure 9. Method to analyze data (1 of 2)

```

193         elif option == "2":      # Calculate Max Climb Rate
194             max_climb_rate = 0
195             if not(alt_column is None or time_column is None):
196                 for idx2, row in enumerate(inputData):
197                     try:
198                         climb_rate = (row[alt_column]-inputData[idx2-1][alt_column])/
199                                     (row[time_column]-inputData[idx2-1][time_column])
200                         if climb_rate > max_climb_rate:
201                             max_climb_rate = climb_rate
202                     except ZeroDivisionError:
203                         None
204                     except TypeError:
205                         None
206                 print("\n The maximum climb rate is %.2f ft/s \n" % max_climb_rate)
207             else:
208                 print("Data is partial or missing. Returning to main menu...")
209
210         elif option == "3":      # Return to Main Menu
211             print("Returning to Main Menu... \n")
212
213         else:
214             print("Invalid selection. Returning to Main Menu... \n")
215         print()
216

```

Figure 9. Method to analyze data (2 of 2)

The method to display data from memory is very similar to previous assignments, so is not discussed in detail here. Figure 10 below shows the method that displays data from memory.

```

217     @staticmethod
218     def displayListData(inputList):
219         """This function displays data from a list object in a readable format..."""
224         print("\n-----This is your current data-----\n")
225         for row in inputList:
226             print("|".join(map(str,row)))
227         print("\n-----\n")

```

Figure 10. Method to display data from memory

## Main Body

The main body of the script resembles that of Assignment 06 to an extent. It begins with the loading of data from a text file using a method from the Processor class. Then a while loop is initiated; at the beginning of each iteration, the main menu is displayed to the user and their response is recorded. This recorded response is fed into an if-elif-else statement, through which the correct corresponding method



is called and the results produced. The novelty in this script lies in the called functions, which are discussed previously and in the next two sections; thus, no further discussion of the main body is warranted. Figure 11 below shows the main body of the script.

```

229 # Main Body of Script ----- #
230
231 flight_data = Processor.loadDataFromFile(file_name=fileName)
232
233 while (True):
234
235     IO.main_menu_tasks()
236     str_main_choice = IO.main_menu_choice()
237
238     if str_main_choice.strip() == "1":      # Display flight data
239         Processor.displayListData(flight_data)
240     elif str_main_choice.strip() == "2":    # Export flight data
241         Processor.exportBinaryData(file_name=binaryFile,exportData=flight_data)
242     elif str_main_choice.strip() == "3":    # Import flight data
243         unpickledData=Processor.importBinaryData(file_name=binaryFile)
244         if not(unpickledData is None):
245             flight_data=unpickledData
246     elif str_main_choice.strip() == "4":    # Analyze flight performance
247         IO.perf_menu_tasks()
248         str_perf_choice = IO.perf_menu_choice()
249         Processor.analyzePerformance(str_perf_choice,inputData=flight_data)
250     elif str_main_choice.strip() == "5":    # Exit
251         print("Exiting Program. Goodbye. ")
252         break
253     else:
254         print("Invalid selection. Returning to main menu... \n")

```

Figure 11. Main body of script

## Pickling & Unpickling

As alluded to in the Processing section above, this script makes use of pickling and unpickling (i.e., data serialization and de-serialization). In the theoretical use case for this script, flight parameter data is received in the form of a text file directly from the flight recorder. (In all likelihood such data would already be serialized, but this is ignored here for the purposes of illustration). Unlike the example data provided, real life flight parameter data can consist of hundreds of columns and millions of rows; serialization of such data would thus be useful to reduce file sizes. Figure 2 above shows how tabulated data may be received or generated; for persistent, reduced size storage, it would be useful for this to be serialized as binary data (i.e., “pickled”). Using the code in Figure 7, the data in Figure 2 can be serialized

and then saved persistently as a file. Figure 12 below shows the resultant binary data file after the data in Figure 2 is serialized and saved.

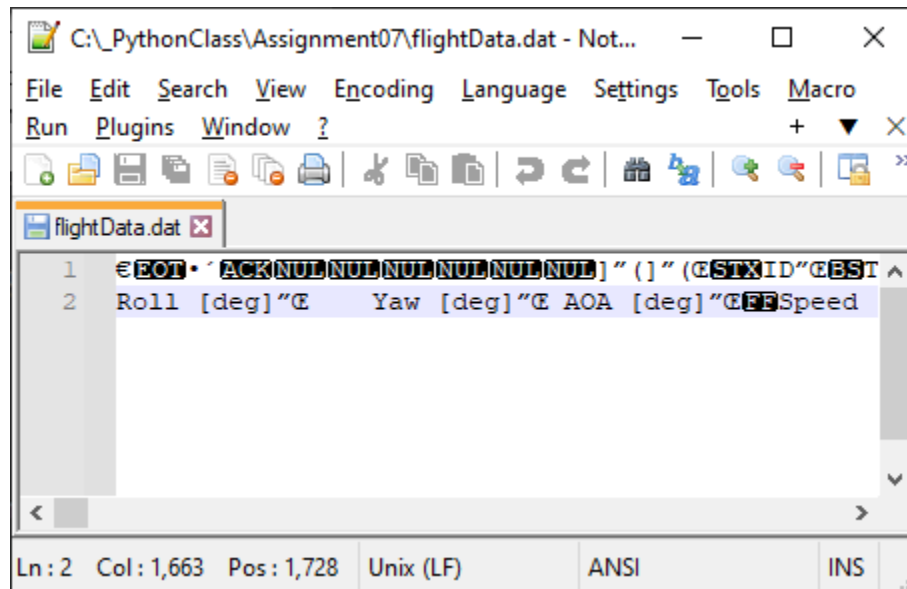


Figure 12. Result of serializing and saving flight data from Figure 2 as binary data

As expected, the data has been obscured following serialization. As can be seen via the scroll wheel in Figure 12, pickling this small an amount of data isn't very useful as far as reducing file sizes. But in instances with enormous files (e.g., real airplane flight parameter data), serializing can aid in file size reduction.

If instead serialized data was received for analysis purposes, the `importBinaryData` shown in Figure 8 can be called to “unpickle” the data and read it into memory, overwriting the existing data object. Figure 13 below illustrates such an example. Here, the initial flight data (see Figure 2) is deleted from the directory. This prompts the program to generate largely empty data (zeros for parameters and Static for flight phase). When the user elects to import data (i.e., “unpickle” and load data), it can be seen that the binary data file has been “unpickled” and passed into the flight data variable. As can be seen in these two examples, pickling and unpickling can be of substantial use in data handling.

```

1 C:\_PythonClass\Assignment07\venv\Scripts\python.exe C:\_PythonClass\
  Assignment07\Assignment07.py
2 flight_data.txt does not exist. Building empty file...
3
4             Menu of Options
5             1) Display flight data
6             2) Export flight data
7             3) Import flight data
8             4) Analyze flight performance
9             5) Exit Program
10
11
12 Which option would you like to perform? [1 to 5] - 1
13
14
15 -----This is your current data-----
16
17 ID|Time [s]|Phase|Alt [ft]|Pitch [deg]|Roll [deg]|Yaw [deg]|AOA [deg]|
  Speed [ft/s]
18 1|5.0|Static|0.0|0.0|0.0|0.0|0.0|0.0
19 2|10.0|Static|0.0|0.0|0.0|0.0|0.0|0.0
20 3|15.0|Static|0.0|0.0|0.0|0.0|0.0|0.0
21 4|20.0|Static|0.0|0.0|0.0|0.0|0.0|0.0
22 5|25.0|Static|0.0|0.0|0.0|0.0|0.0|0.0
23 6|30.0|Static|0.0|0.0|0.0|0.0|0.0|0.0
24 7|35.0|Static|0.0|0.0|0.0|0.0|0.0|0.0
25 8|40.0|Static|0.0|0.0|0.0|0.0|0.0|0.0
26 9|45.0|Static|0.0|0.0|0.0|0.0|0.0|0.0
27 10|50.0|Static|0.0|0.0|0.0|0.0|0.0|0.0
28
29 -----
30
31
32             Menu of Options
33             1) Display flight data
34             2) Export flight data
35             3) Import flight data
36             4) Analyze flight performance
37             5) Exit Program
38
39
40 Which option would you like to perform? [1 to 5] - 3
41
42 Importing data from flightData.dat ...
43
44 Data imported from flightData.dat . Returning to main menu...
45
46
47             Menu of Options
48             1) Display flight data

```

```

49          2) Export flight data
50          3) Import flight data
51          4) Analyze flight performance
52          5) Exit Program
53
54
55 Which option would you like to perform? [1 to 5] - 1
56
57
58 -----This is your current data-----
59
60 ID|Time [s]|Phase|Alt [ft]|Pitch [deg]|Roll [deg]|Yaw [deg]|AOA [deg]|
   Speed [ft/s]
61 1.0|5.0|Taxi|0.0|0.0|0.0|45.0|0.0|0
62 2.0|10.0|Takeoff Roll|0.0|0.0|0.0|0.0|0.0|25
63 3.0|15.0|Climb|20.0|30.0|0.0|0.0|25.0|50
64 4.0|20.0|Climb|40.0|30.0|0.0|0.0|25.0|50
65 5.0|25.0|Climb|60.0|30.0|0.0|0.0|25.0|50
66 6.0|30.0|Climb|80.0|30.0|0.0|0.0|25.0|50
67 7.0|35.0|Cruise|100.0|0.0|0.0|0.0|0.0|75
68 8.0|40.0|Cruise|100.0|0.0|0.0|0.0|0.0|75
69 9.0|45.0|Cruise|100.0|0.0|0.0|0.0|0.0|75
70 10.0|50.0|Cruise|100.0|0.0|0.0|0.0|0.0|75
71 11.0|55.0|Descent|100.0|-30|0.0|0.0|-35|75
72 12.0|60.0|Bank|80.0|0.0|30.0|0.0|0.0|50
73 13.0|65.0|Bank|80.0|0.0|30.0|0.0|0.0|70
74 14.0|70.0|Descent|80.0|-30|0.0|0.0|-35|50
75 15.0|75.0|Descent|60.0|-30|0.0|0.0|-35|50
76 16.0|80.0|Descent|40.0|-30|0.0|0.0|-35|50
77 17.0|85.0|Descent|20.0|-30|0.0|0.0|-35|50
78 18.0|90.0|Landing Roll|0.0|0.0|0.0|0.0|0.0|20
79 19.0|95.0|Landing Roll|0.0|0.0|0.0|0.0|0.0|10
80 20.0|100.0|Static|0.0|0.0|0.0|0.0|0.0|0.0
81
82 -----
83
84
85          Menu of Options
86          1) Display flight data
87          2) Export flight data
88          3) Import flight data
89          4) Analyze flight performance
90          5) Exit Program
91
92
93 Which option would you like to perform? [1 to 5] - 5
94
95 Exiting Program. Goodbye.
96
97 Process finished with exit code 0

```

*Figure 13. De-serialization of binary data and loading into memory for processing*

### Structured Error Handling

Structured error handling is utilized in a few instances throughout the script. It is first used in the `loadDataFromFile` method in the `Processor` class (see Figure 6). In this case, structured error handling is used in the event that no file of the specified name can be found in the same folder. Attempting to open a non-existent file with read permissions would generate a `FileNotFoundError` exception. To preclude the premature shutdown of the program, an exception is made in the method whereby a `FileNotFoundError` would prompt the program to generate a placeholder data table list populated by zeros (see beginning of Figure 13).

Structured error handling is again used in the performance analysis method (`analyzePerformance`) in the `Processor` class. The first instance arises when the user elects to calculate flight time. If it is determined that airplane never took off and/or never landed, then the flight time calculation would yield a `TypeError` due to a one or more non-numbers being subtracted from each other. Depending on which case (never took off and/or never landed), an exception is raised via the testing of a variable; and if statement in the exception statement then filters which response to send the user, informing them of the data's unsuitability. See Figure 14 for the results of this structured error handling.

```
49 Which option would you like to perform? [1 to 5] - 4
50
51
52             Performance Analyses
53             1) Calculate flight time
54             2) Calculate maximum climb rate
55             3) Return to Main Menu
56
57
58 Which analysis would you like to perform? [1 to 3] - 1
59
60 The plane never took off, but landed. Data unintelligible
61
```

*Figure 14. Result of structured error handling when running analysis on faulty data*

The second instance arises if the user elects to calculate the maximum climb rate (change in altitude/change in time) during flight. Two possible errors were thought of: erroneous data where two successive data points share the same time resulting in a `ZeroDivisionError`, and `TypeError` if a string were to be subtracted from a number (e.g., header subtracted from first row data during loop). In both cases, the program should continue on (skipping over faulty data or skipping the header row in math operations). As such, exceptions are raised for both errors with no action taken, allowing for the loop to continue. See the second part of Figure 9 to see this script.

### Results

As the implementation of pickling/unpickling and structured error handling are discussed in detail in the previous two sections, this section covers the results of running the script – in instances without errors and instances where errors are handled.

First is the trivial case: when the necessary data text file is present, the data can be pickled, and analysis performed without error. Figure 15 below shows the preconditions for this case, with a valid flight\_data.txt file, but no binary .dat file.

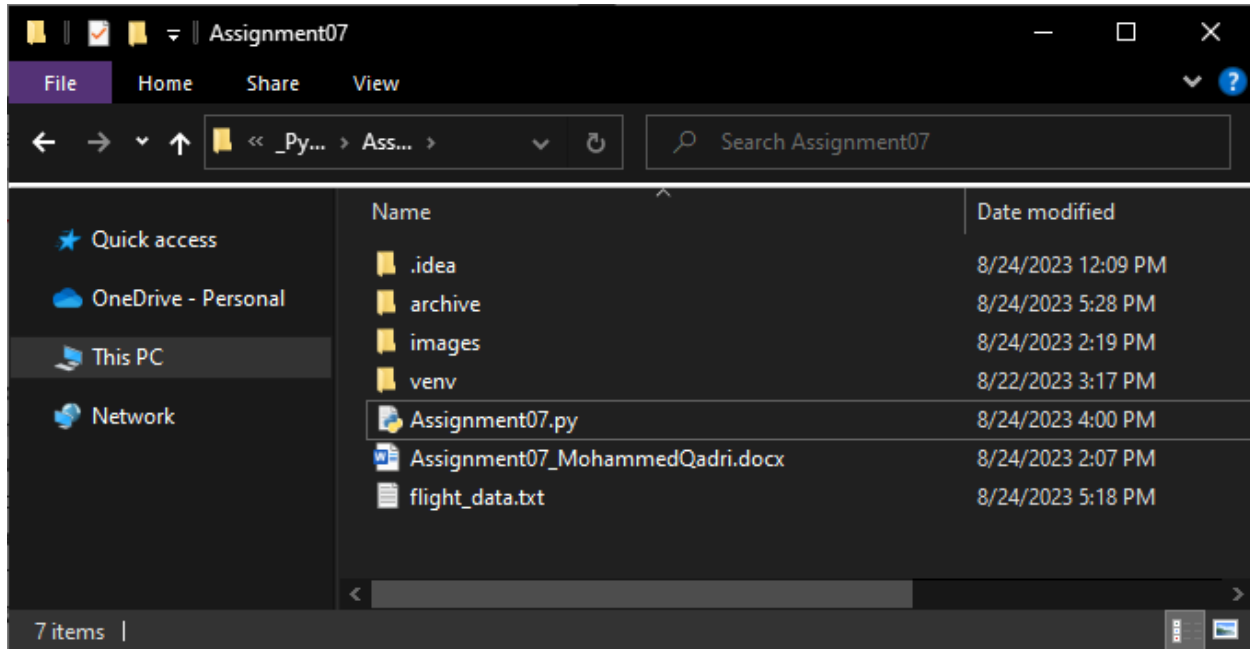


Figure 15. Trivial case preconditions

Figure 16 below shows execution of the script in a PyCharm console printout. First, the data is displayed to demonstrate successful loading from text file (see Figure 2 for default data). Next, the user selects to export (serialize and save) the data (see Figure 12 for serialization results). The user then elects to perform an analysis, with the expected results displayed. The program is then exited. Figure 17 shows the same being performed in the command module. Figure 18 shows the resulting binary data file (flightData.dat) saved in the folder.

```

1 C:\_PythonClass\Assignment07\venv\Scripts\python.exe C:\_PythonClass\
  Assignment07\Assignment07.py
2
3             Menu of Options
4             1) Display flight data
5             2) Export flight data
6             3) Import flight data
7             4) Analyze flight performance
8             5) Exit Program
9
10
11 Which option would you like to perform? [1 to 5] - 1
12
13
14 -----This is your current data-----
15
16 ID|Time [s]|Phase|Alt [ft]|Pitch [deg]|Roll [deg]|Yaw [deg]|AOA [deg]|
  Speed [ft/s]
17 1.0|5.0|Taxi|0.0|0.0|0.0|45.0|0.0|0
18 2.0|10.0|Takeoff Roll|0.0|0.0|0.0|0.0|0.0|25
19 3.0|15.0|Climb|20.0|30.0|0.0|0.0|25.0|50
20 4.0|20.0|Climb|40.0|30.0|0.0|0.0|25.0|50
21 5.0|25.0|Climb|60.0|30.0|0.0|0.0|25.0|50
22 6.0|30.0|Climb|80.0|30.0|0.0|0.0|25.0|50
23 7.0|35.0|Cruise|100.0|0.0|0.0|0.0|0.0|75
24 8.0|40.0|Cruise|100.0|0.0|0.0|0.0|0.0|75
25 9.0|45.0|Cruise|100.0|0.0|0.0|0.0|0.0|75
26 10.0|50.0|Cruise|100.0|0.0|0.0|0.0|0.0|75
27 11.0|55.0|Descent|100.0|-30|0.0|0.0|-35|75
28 12.0|60.0|Bank|80.0|0.0|30.0|0.0|0.0|50
29 13.0|65.0|Bank|80.0|0.0|30.0|0.0|0.0|70
30 14.0|70.0|Descent|80.0|-30|0.0|0.0|-35|50
31 15.0|75.0|Descent|60.0|-30|0.0|0.0|-35|50
32 16.0|80.0|Descent|40.0|-30|0.0|0.0|-35|50
33 17.0|85.0|Descent|20.0|-30|0.0|0.0|-35|50
34 18.0|90.0|Landing Roll|0.0|0.0|0.0|0.0|0.0|20
35 19.0|95.0|Landing Roll|0.0|0.0|0.0|0.0|0.0|10
36 20.0|100.0|Static|0.0|0.0|0.0|0.0|0.0|0.0
37
38 -----
39
40
41             Menu of Options
42             1) Display flight data
43             2) Export flight data
44             3) Import flight data
45             4) Analyze flight performance
46             5) Exit Program
47
48

```

```

49 Which option would you like to perform? [1 to 5] - 2
50
51 Exporting data to  flightData.dat ...
52
53 Data successfully exported to  flightData.dat
54
55         Menu of Options
56         1) Display flight data
57         2) Export flight data
58         3) Import flight data
59         4) Analyze flight performance
60         5) Exit Program
61
62
63 Which option would you like to perform? [1 to 5] - 4
64
65
66         Performance Analyses
67         1) Calculate flight time
68         2) Calculate maximum climb rate
69         3) Return to Main Menu
70
71
72 Which analysis would you like to perform? [1 to 3] - 1
73
74 Flight time = 80.00 s
75
76
77         Menu of Options
78         1) Display flight data
79         2) Export flight data
80         3) Import flight data
81         4) Analyze flight performance
82         5) Exit Program
83
84
85 Which option would you like to perform? [1 to 5] - 5
86
87 Exiting Program. Goodbye.
88
89 Process finished with exit code 0
90

```

*Figure 16. Trivial case running in PyCharm*



```
Command Prompt
Microsoft Windows [Version 10.0.19045.3324]
(c) Microsoft Corporation. All rights reserved.

C:\Users\mqadr>python "C:\_PythonClass\Assignment07\Assignment07.py"

    Menu of Options
    1) Display flight data
    2) Export flight data
    3) Import flight data
    4) Analyze flight performance
    5) Exit Program

Which option would you like to perform? [1 to 5] - 1

-----This is your current data-----

ID|Time [s]|Phase|Alt [ft]|Pitch [deg]|Roll [deg]|Yaw [deg]|AOA [deg]|Speed [ft/s]
1.0|5.0|Taxi|0.0|0.0|0.0|45.0|0.0|0
2.0|10.0|Takeoff Roll|0.0|0.0|0.0|0.0|0.0|25
3.0|15.0|Climb|20.0|30.0|0.0|0.0|25.0|50
4.0|20.0|Climb|40.0|30.0|0.0|0.0|25.0|50
5.0|25.0|Climb|60.0|30.0|0.0|0.0|25.0|50
6.0|30.0|Climb|80.0|30.0|0.0|0.0|25.0|50
7.0|35.0|Cruise|100.0|0.0|0.0|0.0|0.0|75
8.0|40.0|Cruise|100.0|0.0|0.0|0.0|0.0|75
9.0|45.0|Cruise|100.0|0.0|0.0|0.0|0.0|75
10.0|50.0|Cruise|100.0|0.0|0.0|0.0|0.0|75
11.0|55.0|Descent|100.0|-30|0.0|0.0|-35|75
12.0|60.0|Bank|80.0|0.0|30.0|0.0|0.0|50
13.0|65.0|Bank|80.0|0.0|30.0|0.0|0.0|70
14.0|70.0|Descent|80.0|-30|0.0|0.0|-35|50
15.0|75.0|Descent|60.0|-30|0.0|0.0|-35|50
16.0|80.0|Descent|40.0|-30|0.0|0.0|-35|50
17.0|85.0|Descent|20.0|-30|0.0|0.0|-35|50
18.0|90.0|Landing Roll|0.0|0.0|0.0|0.0|0.0|20
19.0|95.0|Landing Roll|0.0|0.0|0.0|0.0|0.0|10
20.0|100.0|Static|0.0|0.0|0.0|0.0|0.0|0

-----
```

```
Command Prompt

Which option would you like to perform? [1 to 5] - 2
Exporting data to flightData.dat ...
Data successfully exported to flightData.dat

Menu of Options
1) Display flight data
2) Export flight data
3) Import flight data
4) Analyze flight performance
5) Exit Program

Which option would you like to perform? [1 to 5] - 4

Performance Analyses
1) Calculate flight time
2) Calculate maximum climb rate
3) Return to Main Menu

Which analysis would you like to perform? [1 to 3] - 1
Flight time = 80.00 s

Menu of Options
1) Display flight data
2) Export flight data
3) Import flight data
4) Analyze flight performance
5) Exit Program

Which option would you like to perform? [1 to 5] - 5
Exiting Program. Goodbye.
```

Figure 17. Trivial case running in command module

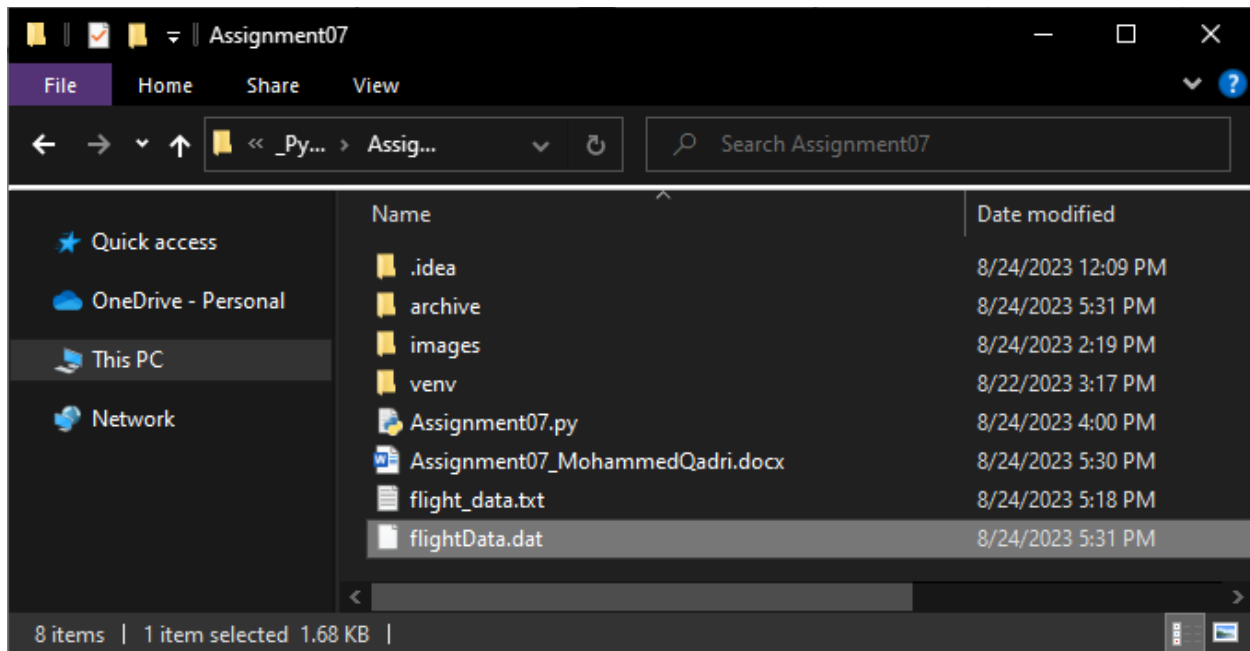


Figure 18. Trivial case binary data saved in folder

The second case focuses on unpickling. In this instance, the pickled data .dat file is present, but no flight\_data.txt file is to be found. The program begins by failing to load data from a non-existent text file. Instead, the program creates filler data that is displayed by the user. Binary data (see Figure 12) is unpickled and loaded; the imported data is displayed. Analysis is then successfully performed. Figure 19 below shows the preconditions, Figure 20 below shows the script running in PyCharm via a console printout, and Figure 21 below shows the script running in command window.

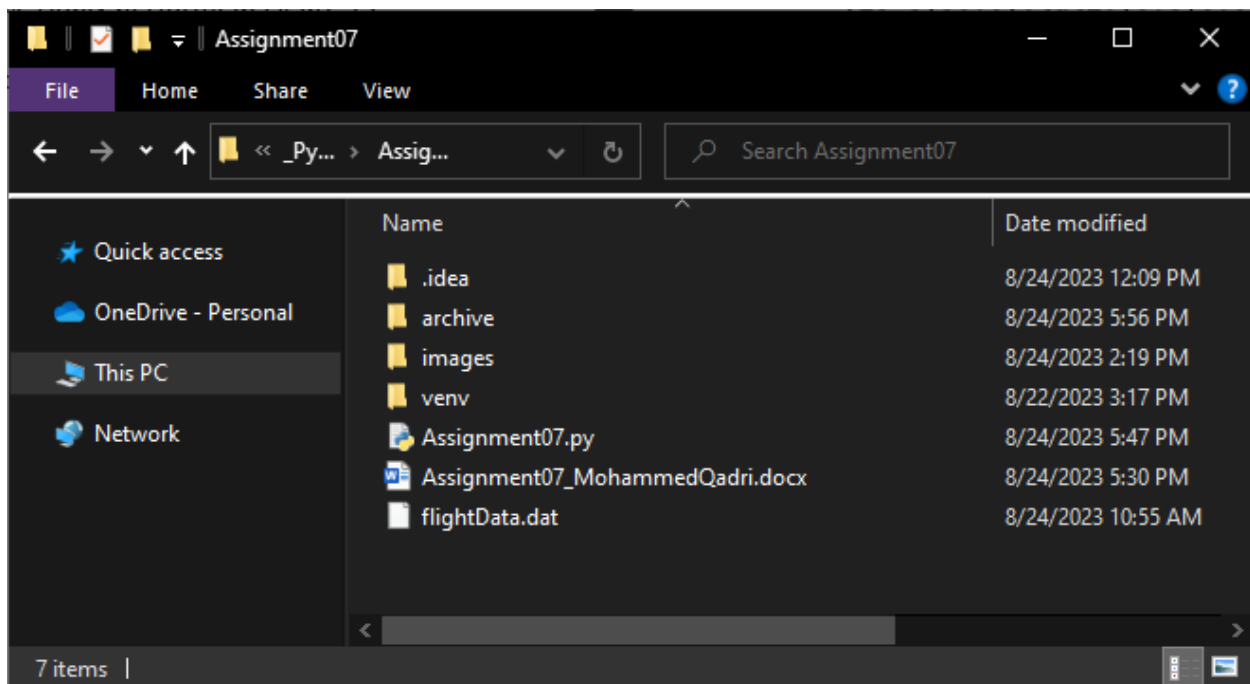


Figure 19. Unpickling case preconditions

```

1 C:\_PythonClass\Assignment07\venv\Scripts\python.exe C:\_PythonClass\
  Assignment07\Assignment07.py
2 flight_data.txt does not exist. Building empty file...
3
4             Menu of Options
5             1) Display flight data
6             2) Export flight data
7             3) Import flight data
8             4) Analyze flight performance
9             5) Exit Program
10
11
12 Which option would you like to perform? [1 to 5] - 1
13
14
15 -----This is your current data-----
16
17 ID|Time [s]|Phase|Alt [ft]|Pitch [deg]|Roll [deg]|Yaw [deg]|AOA [deg]|
  Speed [ft/s]
18 1|5.0|Static|0.0|0.0|0.0|0.0|0.0|0.0
19 2|10.0|Static|0.0|0.0|0.0|0.0|0.0|0.0
20 3|15.0|Static|0.0|0.0|0.0|0.0|0.0|0.0
21 4|20.0|Static|0.0|0.0|0.0|0.0|0.0|0.0
22 5|25.0|Static|0.0|0.0|0.0|0.0|0.0|0.0
23 6|30.0|Static|0.0|0.0|0.0|0.0|0.0|0.0
24 7|35.0|Static|0.0|0.0|0.0|0.0|0.0|0.0
25 8|40.0|Static|0.0|0.0|0.0|0.0|0.0|0.0
26 9|45.0|Static|0.0|0.0|0.0|0.0|0.0|0.0
27 10|50.0|Static|0.0|0.0|0.0|0.0|0.0|0.0
28
29 -----
30
31
32             Menu of Options
33             1) Display flight data
34             2) Export flight data
35             3) Import flight data
36             4) Analyze flight performance
37             5) Exit Program
38
39
40 Which option would you like to perform? [1 to 5] - 3
41
42 Importing data from flightData.dat ...
43
44 Data imported from flightData.dat . Returning to main menu...
45
46
47             Menu of Options
48             1) Display flight data

```

```

49          2) Export flight data
50          3) Import flight data
51          4) Analyze flight performance
52          5) Exit Program
53
54
55 Which option would you like to perform? [1 to 5] - 1
56
57
58 -----This is your current data-----
59
60 ID|Time [s]|Phase|Alt [ft]|Pitch [deg]|Roll [deg]|Yaw [deg]|AOA [deg]|
   Speed [ft/s]
61 1.0|5.0|Taxi|0.0|0.0|0.0|45.0|0.0|0
62 2.0|10.0|Takeoff Roll|0.0|0.0|0.0|0.0|0.0|25
63 3.0|15.0|Climb|20.0|30.0|0.0|0.0|25.0|50
64 4.0|20.0|Climb|40.0|30.0|0.0|0.0|25.0|50
65 5.0|25.0|Climb|60.0|30.0|0.0|0.0|25.0|50
66 6.0|30.0|Climb|80.0|30.0|0.0|0.0|25.0|50
67 7.0|35.0|Cruise|100.0|0.0|0.0|0.0|0.0|75
68 8.0|40.0|Cruise|100.0|0.0|0.0|0.0|0.0|75
69 9.0|45.0|Cruise|100.0|0.0|0.0|0.0|0.0|75
70 10.0|50.0|Cruise|100.0|0.0|0.0|0.0|0.0|75
71 11.0|55.0|Descent|100.0|-30|0.0|0.0|-35|75
72 12.0|60.0|Bank|80.0|0.0|30.0|0.0|0.0|50
73 13.0|65.0|Bank|80.0|0.0|30.0|0.0|0.0|70
74 14.0|70.0|Descent|80.0|-30|0.0|0.0|-35|50
75 15.0|75.0|Descent|60.0|-30|0.0|0.0|-35|50
76 16.0|80.0|Descent|40.0|-30|0.0|0.0|-35|50
77 17.0|85.0|Descent|20.0|-30|0.0|0.0|-35|50
78 18.0|90.0|Landing Roll|0.0|0.0|0.0|0.0|0.0|20
79 19.0|95.0|Landing Roll|0.0|0.0|0.0|0.0|0.0|10
80 20.0|100.0|Static|0.0|0.0|0.0|0.0|0.0|0.0
81
82 -----
83
84
85          Menu of Options
86          1) Display flight data
87          2) Export flight data
88          3) Import flight data
89          4) Analyze flight performance
90          5) Exit Program
91
92
93 Which option would you like to perform? [1 to 5] - 4
94
95
96          Performance Analyses
97          1) Calculate flight time

```

```
98             2) Calculate maximum climb rate
99             3) Return to Main Menu
100
101
102 Which analysis would you like to perform? [1 to 3] - 2
103
104
105 The maximum climb rate is 4.00 ft/s
106
107
108
109             Menu of Options
110             1) Display flight data
111             2) Export flight data
112             3) Import flight data
113             4) Analyze flight performance
114             5) Exit Program
115
116
117 Which option would you like to perform? [1 to 5] - 5
118
119 Exiting Program. Goodbye.
120
121 Process finished with exit code 0
122
```

*Figure 20. Unpickling case running in PyCharm*

```
Command Prompt
C:\Users\mqadr>python "C:\_PythonClass\Assignment07\Assignment07.py"
C:\_PythonClass\Assignment07\flight_data.txt does not exist. Building empty file...

Menu of Options
1) Display flight data
2) Export flight data
3) Import flight data
4) Analyze flight performance
5) Exit Program

Which option would you like to perform? [1 to 5] - 1

-----This is your current data-----

ID|Time [s]|Phase|Alt [ft]|Pitch [deg]|Roll [deg]|Yaw [deg]|AOA [deg]|Speed [ft/s]
1|5.0|Static|0.0|0.0|0.0|0.0|0.0|0.0
2|10.0|Static|0.0|0.0|0.0|0.0|0.0|0.0
3|15.0|Static|0.0|0.0|0.0|0.0|0.0|0.0
4|20.0|Static|0.0|0.0|0.0|0.0|0.0|0.0
5|25.0|Static|0.0|0.0|0.0|0.0|0.0|0.0
6|30.0|Static|0.0|0.0|0.0|0.0|0.0|0.0
7|35.0|Static|0.0|0.0|0.0|0.0|0.0|0.0
8|40.0|Static|0.0|0.0|0.0|0.0|0.0|0.0
9|45.0|Static|0.0|0.0|0.0|0.0|0.0|0.0
10|50.0|Static|0.0|0.0|0.0|0.0|0.0|0.0

-----
```

```
Command Prompt

Menu of Options
1) Display flight data
2) Export flight data
3) Import flight data
4) Analyze flight performance
5) Exit Program

Which option would you like to perform? [1 to 5] - 3
Importing data from C:\_PythonClass\Assignment07\flightData.dat ...
Data imported from C:\_PythonClass\Assignment07\flightData.dat . Returning to main menu...

Menu of Options
1) Display flight data
2) Export flight data
3) Import flight data
4) Analyze flight performance
5) Exit Program

Which option would you like to perform? [1 to 5] - 1

-----This is your current data-----
ID|Time [s]|Phase|Alt [ft]|Pitch [deg]|Roll [deg]|Yaw [deg]|AOA [deg]|Speed [ft/s]
1.0|5.0|Taxi|0.0|0.0|0.0|45.0|0.0|0
```



```
Command Prompt
Which option would you like to perform? [1 to 5] - 1

-----This is your current data-----

ID|Time [s]|Phase|Alt [ft]|Pitch [deg]|Roll [deg]|Yaw [deg]|AOA [deg]|Speed [ft/s]
1.0|5.0|Taxi|0.0|0.0|0.0|45.0|0.0|0
2.0|10.0|Takeoff Roll|0.0|0.0|0.0|0.0|0.0|25
3.0|15.0|Climb|20.0|30.0|0.0|0.0|25.0|50
4.0|20.0|Climb|40.0|30.0|0.0|0.0|25.0|50
5.0|25.0|Climb|60.0|30.0|0.0|0.0|25.0|50
6.0|30.0|Climb|80.0|30.0|0.0|0.0|25.0|50
7.0|35.0|Cruise|100.0|0.0|0.0|0.0|0.0|75
8.0|40.0|Cruise|100.0|0.0|0.0|0.0|0.0|75
9.0|45.0|Cruise|100.0|0.0|0.0|0.0|0.0|75
10.0|50.0|Cruise|100.0|0.0|0.0|0.0|0.0|75
11.0|55.0|Descent|100.0|-30|0.0|0.0|-35|75
12.0|60.0|Bank|80.0|0.0|30.0|0.0|0.0|50
13.0|65.0|Bank|80.0|0.0|30.0|0.0|0.0|70
14.0|70.0|Descent|80.0|-30|0.0|0.0|-35|50
15.0|75.0|Descent|60.0|-30|0.0|0.0|-35|50
16.0|80.0|Descent|40.0|-30|0.0|0.0|-35|50
17.0|85.0|Descent|20.0|-30|0.0|0.0|-35|50
18.0|90.0|Landing Roll|0.0|0.0|0.0|0.0|0.0|20
19.0|95.0|Landing Roll|0.0|0.0|0.0|0.0|0.0|10
20.0|100.0|Static|0.0|0.0|0.0|0.0|0.0|0.0

-----

Menu of Options
```

```
Command Prompt

Menu of Options
1) Display flight data
2) Export flight data
3) Import flight data
4) Analyze flight performance
5) Exit Program

Which option would you like to perform? [1 to 5] - 4

Performance Analyses
1) Calculate flight time
2) Calculate maximum climb rate
3) Return to Main Menu

Which analysis would you like to perform? [1 to 3] - 2

The maximum climb rate is 4.00 ft/s

Menu of Options
1) Display flight data
2) Export flight data
3) Import flight data
4) Analyze flight performance
5) Exit Program

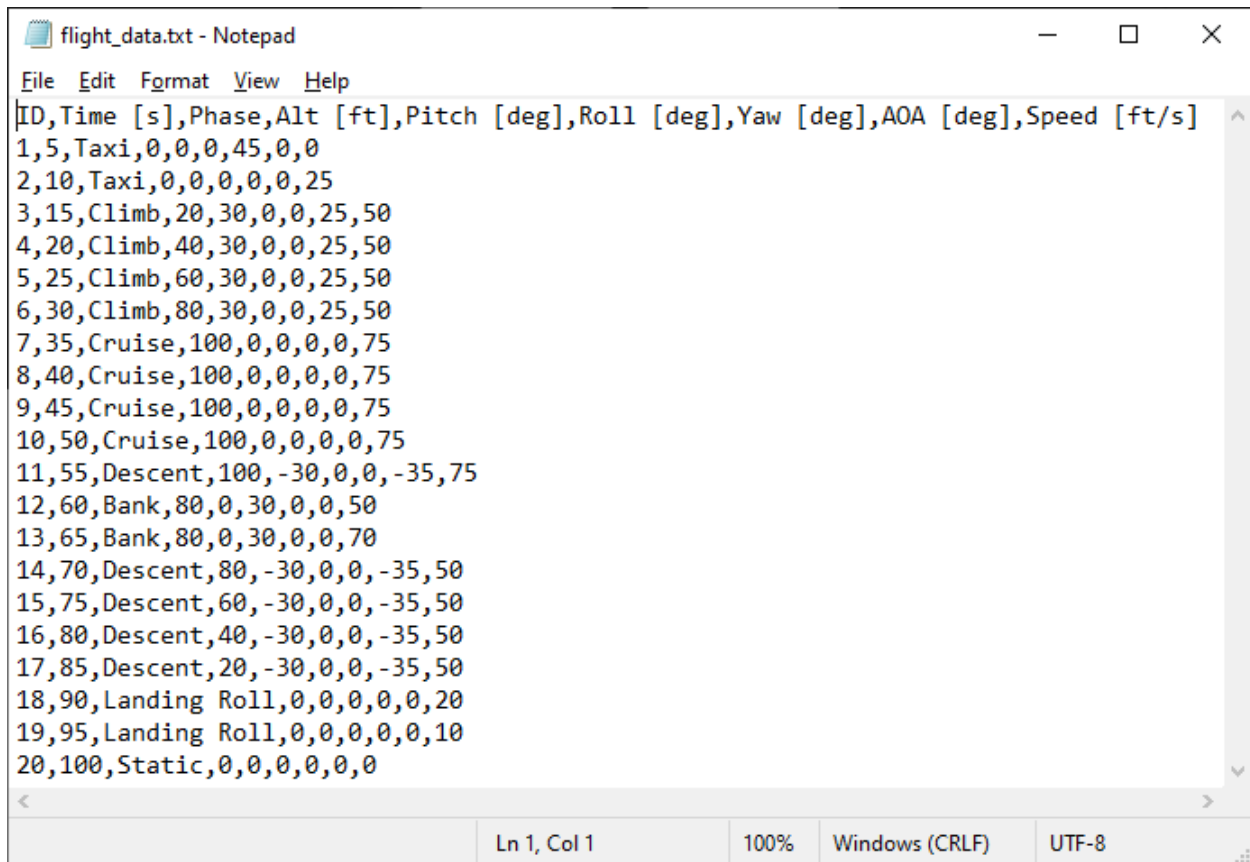
Which option would you like to perform? [1 to 5] - 5

Exiting Program. Goodbye.

C:\Users\mqadr>
```

Figure 21. Unpickling case running in command window

The final case focuses on structured error handling. In this case, the necessary flight\_data.txt file is present, but contains faulty data (no Takeoff is recorded). In this case, the data is read from the text file and analysis attempted. However, the program recognizes a fault in the data and informs the user of the issue before returning to the menu. Figure 22 below shows the faulty data (note no Takeoff Roll), Figure 23 below shows the case running in PyCharm, and Figure 24 shows the case running in command window.



```
flight_data.txt - Notepad
File Edit Format View Help
ID,Time [s],Phase,Alt [ft],Pitch [deg],Roll [deg],Yaw [deg],AOA [deg],Speed [ft/s]
1,5,Taxi,0,0,0,45,0,0
2,10,Taxi,0,0,0,0,0,25
3,15,Climb,20,30,0,0,25,50
4,20,Climb,40,30,0,0,25,50
5,25,Climb,60,30,0,0,25,50
6,30,Climb,80,30,0,0,25,50
7,35,Cruise,100,0,0,0,0,75
8,40,Cruise,100,0,0,0,0,75
9,45,Cruise,100,0,0,0,0,75
10,50,Cruise,100,0,0,0,0,75
11,55,Descent,100,-30,0,0,-35,75
12,60,Bank,80,0,30,0,0,50
13,65,Bank,80,0,30,0,0,70
14,70,Descent,80,-30,0,0,-35,50
15,75,Descent,60,-30,0,0,-35,50
16,80,Descent,40,-30,0,0,-35,50
17,85,Descent,20,-30,0,0,-35,50
18,90,Landing Roll,0,0,0,0,0,20
19,95,Landing Roll,0,0,0,0,0,10
20,100,Static,0,0,0,0,0,0
Ln 1, Col 1 100% Windows (CRLF) UTF-8
```

Figure 22. Error handling case erroneous data

```
C:\_PythonClass\Assignment07\venv\Scripts\python.exe C:\_PythonClass\Assignment07\Assignment07.py

Menu of Options
1) Display flight data
2) Export flight data
3) Import flight data
4) Analyze flight performance
5) Exit Program

Which option would you like to perform? [1 to 5] - 4

Performance Analyses
1) Calculate flight time
2) Calculate maximum climb rate
3) Return to Main Menu

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

The plane never took off, but landed. Data unintelligible

Menu of Options
1) Display flight data
2) Export flight data
3) Import flight data
4) Analyze flight performance
5) Exit Program

Which option would you like to perform? [1 to 5] - 5

Exiting Program. Goodbye.

Process finished with exit code 0
```

Figure 23. Error handling case running in PyCharm

```
Command Prompt
Microsoft Windows [Version 10.0.19045.3324]
(c) Microsoft Corporation. All rights reserved.

C:\Users\mqadr>python "C:\_PythonClass\Assignment07\Assignment07.py"

Menu of Options
1) Display flight data
2) Export flight data
3) Import flight data
4) Analyze flight performance
5) Exit Program

Which option would you like to perform? [1 to 5] - 4

Performance Analyses
1) Calculate flight time
2) Calculate maximum climb rate
3) Return to Main Menu

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

The plane never took off, but landed. Data unintelligible

Menu of Options
1) Display flight data
2) Export flight data
3) Import flight data
4) Analyze flight performance
5) Exit Program

Which option would you like to perform? [1 to 5] - 5

Exiting Program. Goodbye.

C:\Users\mqadr>
```

Figure 24. Error handling case running in command window

## Conclusion

Though the examples used here to demonstrate pickling and structured error handling are perhaps contrived or unrealistic, they provided good experience in both aspects. Robustness to additional errors could likely be implemented and code efficiency code likely be improved as well, but this module demonstrated additional means by which realistic and useful programs can be created in Python. Therefore, the objective of the assignment was accomplished.

## **Resources**

The only resources used in this module with respect to pickling or structured error handling were the module notes, lecture, and the textbook.

Miscellaneous sources were used in helping to correct syntax, however.