Kevin Phan
February 25, 2023
IT FDN 110 A
Assignment07
https://github.com/EliteDarkLord/IntroToProg-Python-Mod07.git

Pickling and Exception Handling

Introduction

In this assignment, I will explain the steps I used to create a simple python calculator script that will enable a user to select a menu choice, two inputs to calculate, and what calculation to perform. The script will also "pickle" the user's option into a binary file as to encrypt the user's menu choice. The script was created in Pycharm IDE and will be run on the Pycharm IDE and OS Command/Terminal on a Macintosh operating system.

Objective

My objective for this script is to provide the user a menu of options numbered 1-6. As shown below in *figure 1* the options I've provided are the following:

1) Calculate Sum
2) Calculate Difference
3) Calculate Product
4) Calculate Quotient
5) Exit Program

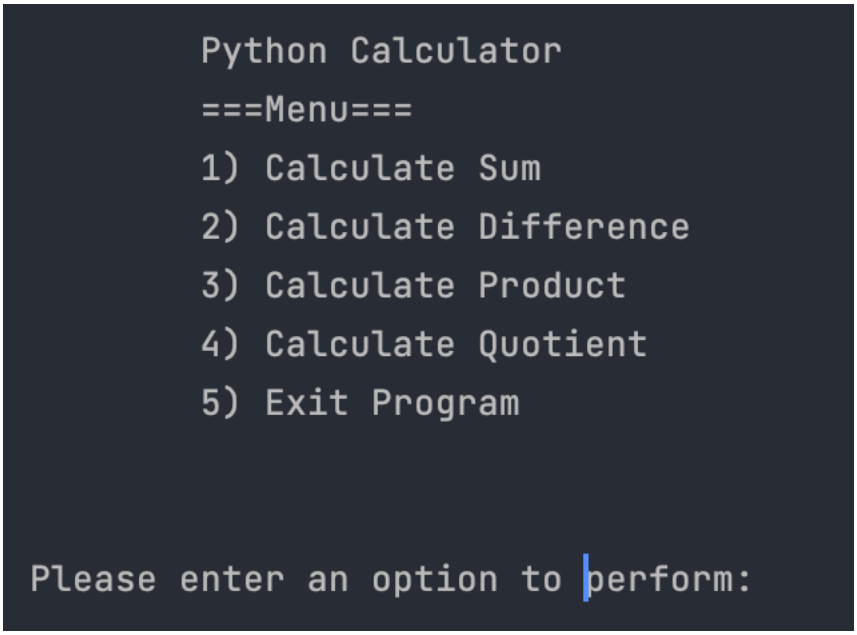

Figure 1: Python Calculator Menu

Drafting the Code

Defining Variables

The global variables I've defined in this script are represented in *figure 2*.

Choice	This global string variable will store the user's menu choice
Value1	This global variable represents the user's num1 input
Value2	This global variable represents the user's num2 input
listData	This list variable will store the user's menu choice into a list object
strFileName	This file will store the user's menu choice in binary form

Table 1: Glabal Variables

```
10  import pickle  # imports code from another code file
11
12  # Data ---------------------------------------- #
13  choice = ""                 # contains user's menu choice
14  value1 = 0                  # represents user's first num input
15  value2 = 0                  # represents user's second num input
16  listData = []               # stores user's choice into a list object
17  strFileName = 'AppData.dat'  # file that will contain user's choice
```

Figure 2: Defining global variables

Classes

The python calculator script is organized into five classes as shown in *figure 3*:
IO – Responsible for inputs and outputs of various functions
Processor – performs the math operations/calculations and error handling
Num1_invalid – Error handling for user's first num input
Num2_invalid – error handling for user's second num input
Divide_by_zero – error handling to check for dividing by 0 particularly for calculating quotient.

```
# This class contains the input and output for the script
class IO:...


# This class will process the user's input data to perform the calculations
class Processor:...


# This class checks if user's first num1 input is valid
class num1_invalid():...


# This class checks if user's second num2 input is valid
class num2_invalid():...


# This class checks if function dividevalues contains a 0 for num2
class divide_by_zero():...
```

Figure 3: Python Calculator Classes

IO (input/output)

The main purpose of Class IO is to gather the user's input of two numbers and output the results based on the user's choice. I created the following functions under this class as shown in *figure 4:*

```
# This class contains the input and output for the script
class IO:

    @staticmethod
    def menu():...

    @staticmethod
    def user_input_task():...

    # Output the sum of two values
    @staticmethod
    def output_sum():...

    # Output the difference of two values
    @staticmethod
    def output_difference():...

    # Output the product of two values
    @staticmethod
    def output_product():...

    # Output the quotient of two values
    @staticmethod
    def output_quotient():...
```

Figure 4: Class IO functions

Menu()

This function is a simple display of the python calculator menu that provides the user information to choose from.

```
24      def menu():
25          print('''
26          Python Calculator
27          ===Menu===
28          1) Calculate Sum
29          2) Calculate Difference
30          3) Calculate Product
31          4) Calculate Quotient
32          5) Exit Program
33          ''')
34          print()
```

Figure 5: Menu of options

User_input_task()

This function uses the *input()* function to request user input by prompting a message to make a menu selection. In line 45, the user's input choice is stored in an object list and is saved into a file. Line 50 returns the user's choice to be utilized elsewhere within the script.

```
37      def user_input_task():
38          """
39          returns the user's choice from the menu
40
41          :return: (string) user's option input
42          """
43
44          choice = input("Please enter an option to perform: ")
45          list = [choice]
46
47          # Pickling user's menu choice
48          Processor.save_data_to_file(strFileName, list)
49
50          return choice
```

Figure 6: Storing user's menu choice

output_sum(), output_difference(), output_product(), output_quotient()

The next set of functions shown in *figure* 7 in this class are the outputs of the sum, difference, product, and quotient. The summary for all of these functions is defining a local variable to another class function value and outputs a message with the results of the calculation. Each function passes two arguments or parameters which is “num1” and “num2”. These parameters represents the user’s two input values for each calculation to perform.

```
52      # Output the sum of two values
53      @staticmethod
54      def output_sum():
55          sum = Processor.addvalues(num1=value1, num2=value2)
56          print("\nThe sum of the two values is: ", sum)
57
58      # Output the difference of two values
59      @staticmethod
60      def output_difference():
61          difference = Processor.subtractvalues(num1=value1, num2=value2)
62          print("\nThe difference of the two values is: ", difference)
63
64      # Output the product of two values
65      @staticmethod
66      def output_product():
67          product = Processor.multiplyvalues(num1=value1, num2=value2)
68          print("\nThe product of the two values is: ", product)
69
70      # Output the quotient of two values
71      @staticmethod
72      def output_quotient():
73          quotient = Processor.dividevalues(num1=value1, num2=value2)
74          print("\nThe quotient of the two values is: ", quotient)
```

Figure 7: Output functions for sum, difference, product, and quotient

Addvalues(num1, num2)

In *figure* 8 this function under class “Processor” is utilized when the user selects option ‘1’ from the menu to calculate the sum of two values. The function passes two parameters “num1” and “num2” and will add the two values. I used *while* loop with a nested *Try, Except* to capture any invalid inputs such as a char for either input one or two. In the try code block, the script will request for two number inputs from the user and converts the value into a float. An *if-elif-else* statement is inside this *Try* block to capture any invalid data by the user. From lines 94-98 if the user’s first input is a char then the script will raise the exception function *num1_invalid* and *num2_invalid* for num2. If the user inputs two valid inputs, then the script will execute the else condition and add the two inputs and break the while loop. After the loop is broken, the function returns the sum of the two values.

The same code structure is similar for functions subtractvalues() and multiplyvalues().

```python
    # Calculate the sum of two values
    def addvalues(num1, num2):
        """
        returns the sum of two floats

        :param num1: (float) first user's num input
        :param num2: (float) user's second number input
        :return: (float) sum of num1 and num2
        """
        while True:
            try:
                num1 = input("Please enter the 1st number: ")
                num2 = input("Please enter the 2nd number: ")

                if str(num1).isalpha():
                    raise num1_invalid.__str__(num1)

                elif str(num2).isalpha():
                    raise num2_invalid.__str__(num2)

                else:
                    sum = float(num1) + float(num2)
                    break

            except Exception as e:
                print("Non-specific error\n")

        return sum
```

Figure 8: add values function

Dividevalues(num1, num2)

Shown in *figure 9* dividevalues passes two arguments which contains the user's two inputs to calculate the quotient. Similar to the addvalues coding structure, this function uses a while loop to continuously check the user's inputs. However, in the if-elif-else statement, if the user inputs 0 or 0.0 as the second value, a custom divide_by_zero exception class will be raised as shown in *figure 10* and outputs a custom error message to the user shown in *figure 11.*

```
157     # Calculate the quotient of two values
158     def dividevalues(num1, num2):
159
160         while True:
161
162             try:
163                 num1 = input("\nPlease enter the 1st number (numerator): ")
164                 num2 = input("Please enter the 2nd number (denominator): ")
165
166                 if str(num1).isalpha():
167                     raise num1_invalid.__str__(num1)
168
169                 elif str(num2).isalpha():
170                     raise num2_invalid.__str__(num2)
171
172                 elif int(num2) == 0:
173                     raise divide_by_zero()
174
175                 else:
176                     quotient = float(num1) / float(num2)
177                     break
178
179             except Exception as e:
180                 print(e)
181
182         return quotient
```

Figure 9: dividevalues function

```
214 # This class checks if function dividevalues contains a 0 for num2
215 class divide_by_zero(Exception):
216     def __str__(self):
217         return 'CANNOT DIVIDE BY 0!'
```

Figure 10: divide_by_zero exception class

```
        Python Calculator
        ===Menu===
        1) Calculate Sum
        2) Calculate Difference
        3) Calculate Product
        4) Calculate Quotient
        5) Exit Program


Please enter an option to perform: 4

Please enter the 1st number (numerator): 0.0
Please enter the 2nd number (denominator): 0.0
CANNOT DIVIDE BY 0!

Please enter the 1st number (numerator): 1
Please enter the 2nd number (denominator): 0.0000000
CANNOT DIVIDE BY 0!

Please enter the 1st number (numerator): 1
Please enter the 2nd number (denominator): 0.0.0.0
CANNOT DIVIDE BY 0!
```

Figure 11: Divide by 0 custom message output

The last function shown in *figure 12* inside the processor class is the save_data_to_file. The purpose of this function was to dump the user's menu choice input into a file in binary format. This function passes the global variable file name and data list, opens the file and with the imported pickle in line 10 of the script, the *dump()* pickle function is used to place the list of data or user's choice into the file in binary format.

```
184     # Saving user's input into a file
185     def save_data_to_file(file_name, list_of_data):
186         """
187
188         :param list_of_data: (list) stores list of data
189         :param file_name: (string) file name containing list of data
190         :return:
191         """
192         file = open(file_name, "ab")
193         pickle.dump(list_of_data, file)
```

Figure 12: Pickling function

Exception Handling

In *figure* 13 there are three Exception classes in this script as mentioned earlier. Num1_invalid and num2_invalid will be raised if the user enters a char or symbol for the first or second number input. This exception will prompt a message indicating the entered input is not valid.

```
195     # This class checks if user's first num1 input is valid
196     class num1_invalid(Exception):
197         def __str__(num1):
198             print("Please do not use", "(" + num1 + ")", "as an input!\n")
199
200
201     # This class checks if user's second num2 input is valid
202     class num2_invalid(Exception):
203         def __str__(num2):
204             print("Please do not use", "(" + num2 + ")", "as an input!\n")
205
206
207     # This class checks if function dividevalues contains a 0 for num2
208     class divide_by_zero(Exception):...
```

Figure 13: Custom message for exception handling

Presentation

In *figure 14,* the last section of this script contains the presentation of the user's choices. In this while loop in line 215, the menu is displayed followed by the definition of global variable 'choice' assigned to the returned value of IO.user_input_task(). Based on this function, the variable 'choice' will be compared in the if-elif-else statements and execute the functions within. The script will run the script until the user enters '5' to exit the program.

```
213 # Presentation ------------------------------------ #
214 while True:
215     IO.menu()
216     choice = IO.user_input_task()
217
218     # Option 1: Calculate Sum of two numbers
219     if choice == "1":
220         IO.output_sum()
221         continue
222
223     # Option 2: Calculate Difference of two numbers
224     elif choice == "2":
225         IO.output_difference()
226         continue
227
228     # Option 3: Calculate Product of two numbers
229     elif choice == "3":
230         IO.output_product()
231         continue
232
233     # Option 4: Calculate Quotient of two numbers
234     elif choice == "4":
235         IO.output_quotient()
236         continue
237
238     # Option 6: Exit's program
239     elif choice == "5":
240         print("Goodbye!")
241         break
242
243     else:
244         print(choice, "is not on the menu!")
245         continue
```

Figure 14: Presentation

Running the Code

Pycharm IDE

Figure 15 shows a snapshot of performing option 1 of the python calculator to calculate the sum of two values. I entered the char 'a' to raise the custom class exception num1_invalid with the custom message.

```
        Python Calculator
        ===Menu===
        1) Calculate Sum
        2) Calculate Difference
        3) Calculate Product
        4) Calculate Quotient
        5) Exit Program


Please enter an option to perform: 1
Please enter the 1st number: a
Please enter the 2nd number: 1

Please do not use (a) as an input!
A char was entered as input. Expected int or float

Please enter the 1st number: 1
Please enter the 2nd number: 1

The sum of the two values is:  2.0
```

Figure 15: Pycharm IDE output of Calculate Sum

Similarly I tested the divide_by_zero class exception to capture the exception handling as shown in *figure 16*

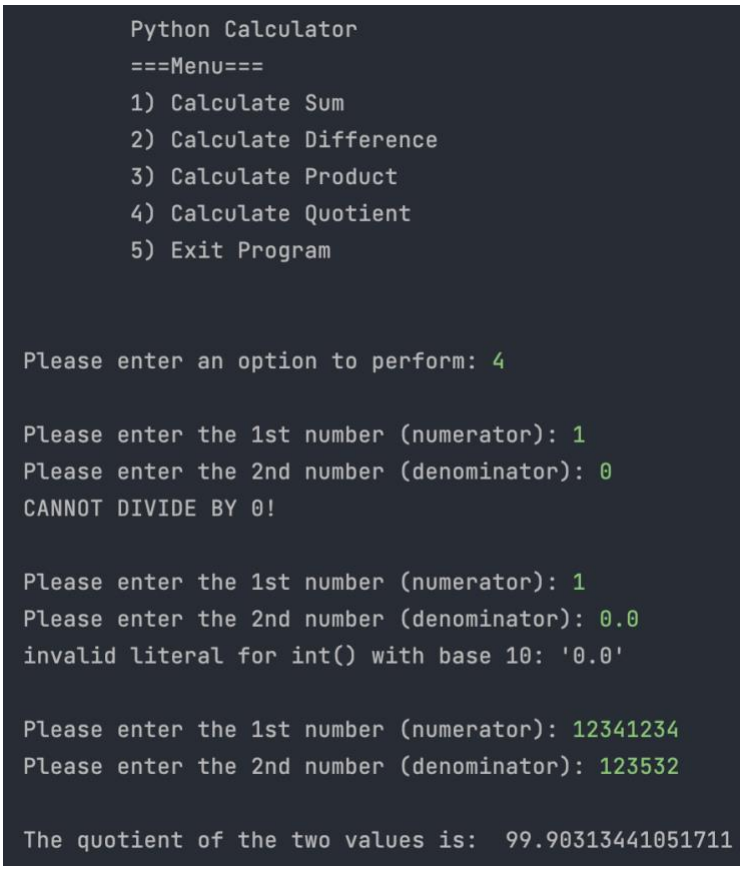

Figure 16: Raising divide_by_zero exception

OS Command/Terminal

Figure 17 provides an example of the python calculator script on OS Command/Terminal. Using the terminal command *cd* to change directory to the appropriate folder containing the program file. Then using *ls* to list out the files within the designated folder I performed the quotient calculation to raise the divide_by_zero exception.

Figure 18 provides a snapshot of the pickling function storing the user's choice into a file in binary format.

```
Last login: Wed Feb 22 17:18:44 on ttys000
Kevin@Trams-Macbook ~ % cd Documents/_PythonClass/Assignment07/
Kevin@Trams-Macbook Assignment07 % ls
AppData.dat                     Assignment07_KevinPhan.pdf
Assignment07.py                 ~$signment07_KevinPhan.docx
Assignment07_KevinPhan.docx
Kevin@Trams-Macbook Assignment07 % python3 Assignment07.py

        Python Calculator
        ===Menu===
        1) Calculate Sum
        2) Calculate Difference
        3) Calculate Product
        4) Calculate Quotient
        5) Exit Program


Please enter an option to perform: 4

Please enter the 1st number (numerator): 2
Please enter the 2nd number (denominator): 0
CANNOT DIVIDE BY 0!

Please enter the 1st number (numerator): 10
Please enter the 2nd number (denominator): 10

The quotient of the two values is:  1.0

        Python Calculator
        ===Menu===
        1) Calculate Sum
        2) Calculate Difference
        3) Calculate Product
        4) Calculate Quotient
        5) Exit Program


Please enter an option to perform: 5
Goodbye!
```

Figure 17: Python Calculator on OS Command/Terminal

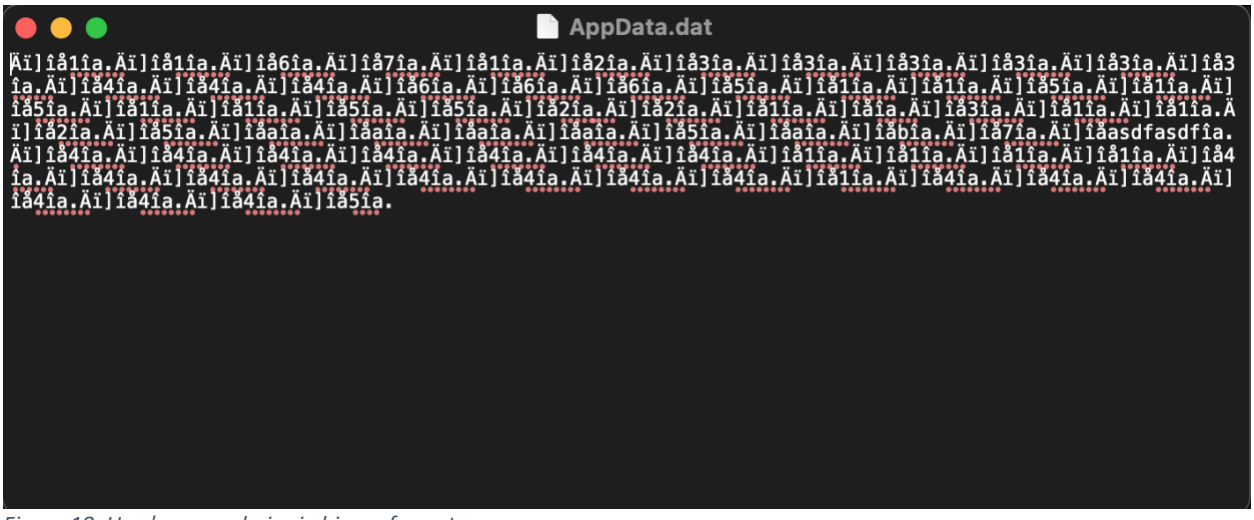

Figure 18: User's menu choice in binary format

Conclusion

From watching the course video, reading chapter 7 of the textbook, and following along the labs in the "_Mod7PythonProgrammingNotes" document, and watching various videos recommended by the instructor, I was able to utilize Pycharm IDE to create a working python calculator script with exception handling and pickling.