**Name : **Leonardo Espinoza

**Date : **August 23, 2023

**Course : **Foundations of Programming, Python

Assignment 07

Link: https://github.com/lesping/lesping-IntroToProg-Python-Mod07

Files and Exceptions

Introduction

During this module we have reviewed the different ways to work with plain text files to store information permanently and read or manipulate it when necessary. Additionally, we investigated and learned how to use the Python Pickle Module, which allows us to store and work with data that requires more complexity. On the other hand, we also learned to intercept and handle exceptions to prevent the program from stopping abruptly and generating an error message. Finally, we also learned how to create better and more advanced GitHub pages. In the first section of the document, I will briefly explain about the Pickle Module and in the second part I will describe the Structured Error Handling.

I. Pickle Module: To Preserve a Complex Piece of Data

Pickle means preserve, and in Python it is no exception, because with this method it is possible to preserve more complex data. The Pickle Module implements binary protocols for serializing and de-serializing a Python object structure. The process to converts any kind of python objects (list, dict, etc.) into byte streams (0s and 1s) is called pickling or serialization and the process to converts the byte stream (generated through pickling) back into python objects by a process called as unpickling.

The following types can be serialized and deserialized using the Pickle module:

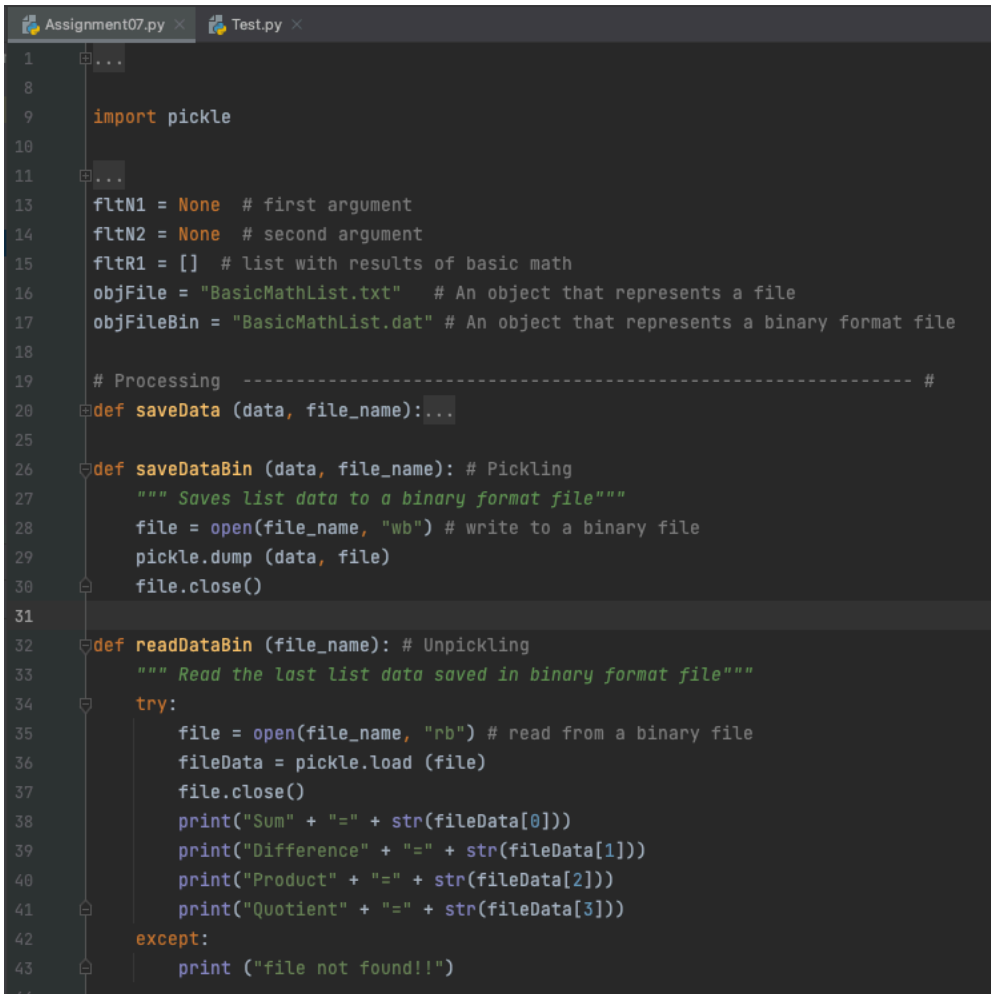
- All native data types supported by Python (booleans, None, integers, floats, complex numbers, strings, bytes, byte arrays).
- Dictionaries, sets, lists, and tuples - as long as they contain pickle-able objects.
- Functions and classes that are defined at the top level of a module.

Pickle objects must be stored in a binary file, they cannot be stored in a text file.

Only after importing pickle module, we can do pickling and unpickling. Importing pickle can be done using the following command: “import pickle”. The pickle module provides the following functions to make the pickling process more convenient:

- dump(object, file, [,bin])
- load(file)

To preparate this assignment I used the BasicMath program we worked at the beginning as a base, I improved it and I used the pickle tool to save a list in binary format with the results of the four basic math operations (sum, difference, product and quotient) (Figure 1).



```python
...

import pickle

...
fltN1 = None  # first argument
fltN2 = None  # second argument
fltR1 = []  # list with results of basic math
objFile = "BasicMathList.txt"   # An object that represents a file
objFileBin = "BasicMathList.dat" # An object that represents a binary format file

# Processing  ---------------------------------------------------------------- #
def saveData (data, file_name):...

def saveDataBin (data, file_name): # Pickling
    """ Saves list data to a binary format file"""
    file = open(file_name, "wb") # write to a binary file
    pickle.dump (data, file)
    file.close()

def readDataBin (file_name): # Unpickling
    """ Read the last list data saved in binary format file"""
    try:
        file = open(file_name, "rb") # read from a binary file
        fileData = pickle.load (file)
        file.close()
        print("Sum" + "=" + str(fileData[0]))
        print("Difference" + "=" + str(fileData[1]))
        print("Product" + "=" + str(fileData[2]))
        print("Quotient" + "=" + str(fileData[3]))
    except:
        print ("file not found!!")
```

Figure 1: Example of using the Pickle Module.

II. Structured Error Handling

An exception is an event, which occurs during the execution of a program that disrupts the normal flow of the program's instructions. In general, when a Python script encounters a situation that it cannot cope with, it raises an exception. An exception is a Python object that represents an error. Using Python's exception handling functionality, it is possible to prevent the program ending abruptly.

Different kinds of errors result in different type of exceptions. Here are some of the most common types of exceptions in Python:

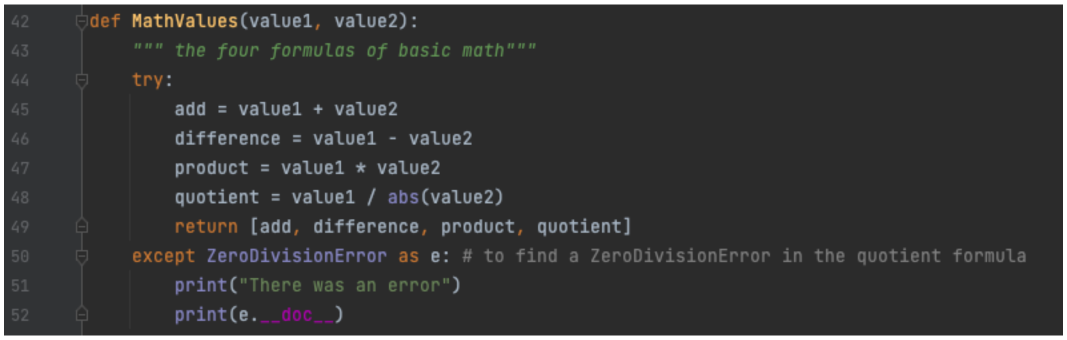
- **SyntaxError:** This exception is raised when the interpreter encounters a syntax error in the code, such as a misspelled keyword, a missing colon, or an unbalanced parenthesis.
- **TypeError:** This exception is raised when an operation or function is applied to an object of the wrong type, such as adding a string to an integer.
- **NameError:** This exception is raised when a variable or function name is not found in the current scope.
- **IndexError:** This exception is raised when an index is out of range for a list, tuple, or other sequence types.

- **KeyError:** This exception is raised when a key is not found in a dictionary.
- **ValueError:** This exception is raised when a function or method is called with an invalid argument or input, such as trying to convert a string to an integer when the string does not represent a valid integer.
- **AttributeError:** This exception is raised when an attribute or method is not found on an object, such as trying to access a non-existent attribute of a class instance.
- **IOError:** This exception is raised when an I/O operation, such as reading or writing a file, fails due to an input/output error.
- **ZeroDivisionError:** This exception is raised when an attempt is made to divide a number by zero.
- **ImportError:** This exception is raised when an import statement fails to find or load a module.

The most basic way to handle exceptions is to use the *try* statement with an *except* clause. Statements that can raise exceptions are kept inside the try clause and the statements that handle the exception are written inside except clause. You can also use the same *except* statement to handle multiple exceptions or with no exceptions defined. Here is simple syntax of *try....except...else* blocks:

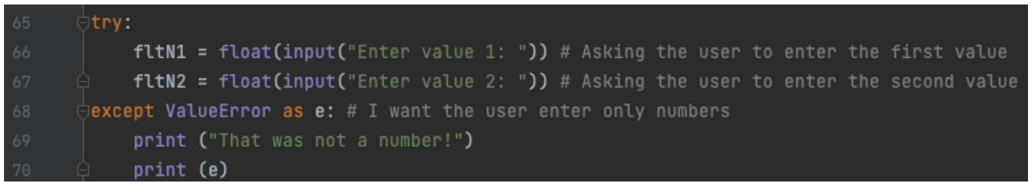
```
try:
  You do your operations here;
  ......................
except(Exception1[, Exception2[,...ExceptionN]]]):
  If there is any exception from the given exception list,
  then execute this block.
  ......................
else:
  If there is no exception then execute this block.
```

In this exercise you use the try statement with the except clause where I specified the exception type ZeroDivisionError (Figure 2) and ValueError (Figure 3).



```
def MathValues(value1, value2):
    """ the four formulas of basic math"""
    try:
        add = value1 + value2
        difference = value1 - value2
        product = value1 * value2
        quotient = value1 / abs(value2)
        return [add, difference, product, quotient]
    except ZeroDivisionError as e: # to find a ZeroDivisionError in the quotient formula
        print("There was an error")
        print(e.__doc__)
```

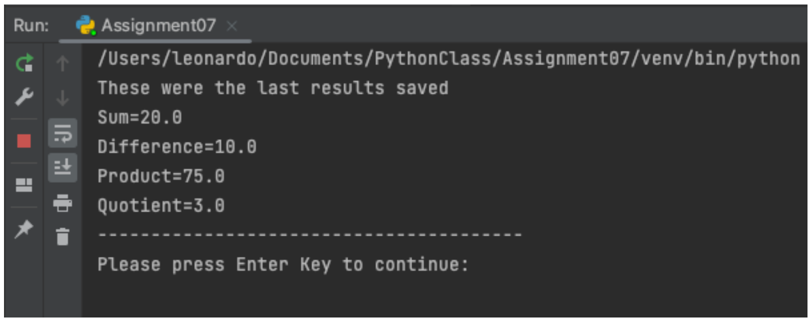
Figure 2: Example of handling exceptions (ZeroDivisionError).



```
try:
    fltN1 = float(input("Enter value 1: ")) # Asking the user to enter the first value
    fltN2 = float(input("Enter value 2: ")) # Asking the user to enter the second value
except ValueError as e: # I want the user enter only numbers
    print ("That was not a number!")
    print (e)
```

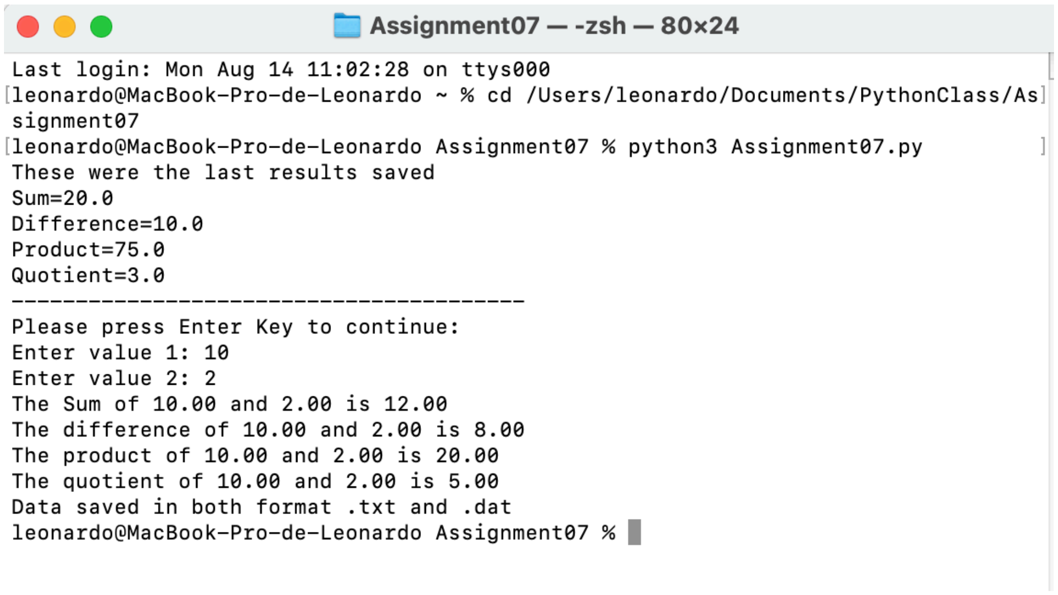
Figure 3: Example of handling exceptions (ValueError).

Then I validated the script in PyCharm and the Command Terminal. I ran the program without problems (Figure 4 and 5).



```
/Users/leonardo/Documents/PythonClass/Assignment07/venv/bin/python
These were the last results saved
Sum=20.0
Difference=10.0
Product=75.0
Quotient=3.0
----------------------------------------
Please press Enter Key to continue:
```

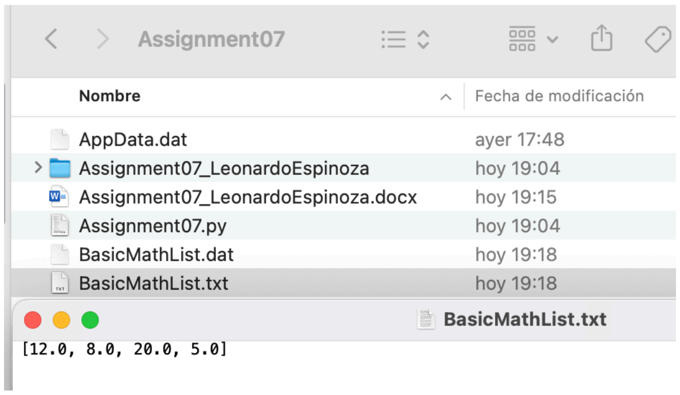
Figure 4: My script running in PyCharm



```
Last login: Mon Aug 14 11:02:28 on ttys000
leonardo@MacBook-Pro-de-Leonardo ~ % cd /Users/leonardo/Documents/PythonClass/As
signment07
leonardo@MacBook-Pro-de-Leonardo Assignment07 % python3 Assignment07.py
These were the last results saved
Sum=20.0
Difference=10.0
Product=75.0
Quotient=3.0
----------------------------------------
Please press Enter Key to continue:
Enter value 1: 10
Enter value 2: 2
The Sum of 10.00 and 2.00 is 12.00
The difference of 10.00 and 2.00 is 8.00
The product of 10.00 and 2.00 is 20.00
The quotient of 10.00 and 2.00 is 5.00
Data saved in both format .txt and .dat
leonardo@MacBook-Pro-de-Leonardo Assignment07 %
```

Figure 5: My script running from the command shell of my computer

Finally, I opened the plain text file *BasicMathList.txt* and it can be demonstrated that the code works by saving the information correctly (Figure 6).



Nombre	Fecha de modificación
AppData.dat	ayer 17:48
Assignment07_LeonardoEspinoza	hoy 19:04
Assignment07_LeonardoEspinoza.docx	hoy 19:15
Assignment07.py	hoy 19:04
BasicMathList.dat	hoy 19:18
BasicMathList.txt	hoy 19:18

```
[12.0, 8.0, 20.0, 5.0]
```

Figure 6: Data saved into .txt and .dat files

Summary

Using the different resources for this module (recommended reading, professor's notes, and video lecture) and the material of the web pages that I researched, in the first part of this document I briefly explained and gave an example about working with the Python Pickle Module for manage more complex data, while in the second part a I tried to describe and give an example related to the Structured Error Handling, both very relevant tools to create and have better control of our scripts. Additionally, in this module I learned a little to create better GitHub pages.

Reference

Randal R, Programming with python notes, module 07, 2023.

Michael D., Python Programming for the Absolute Beginner, Third Edition, 2010, Chapter 7.

Youtube, https://www.youtube.com/playlist?list=PLfycUyp06LG9fZllIqBrxLcNV4CR50HEX, Intro to Python Mod07, 2023) (External site).

Tutorialspoint, https://www.tutorialspoint.com/python-pickling, 2023 (External site).

StackAbuse, https://stackabuse.com/introduction-to-the-python-pickle-module/, 2023 (External site).

Python, https://docs.python.org/3/library/pickle.html, 2023 (External site).

Geeksforgeeks, https://www.geeksforgeeks.org/python-exception-handling/, 2023 (External site).

RealPython, https://realpython.com/python-exceptions/, 2023 (External site).

Tutorialspoint, https://www.tutorialspoint.com/python/python_exceptions.htm, 2023 (External site).