Mercedes Gonzalez Gonzalez

August 23th, 2020

IT FDN 110 A

Assignment 07

GitHub Repository: <https://github.com/mglezglez/IntroToProg-Python-Mod07>

GitHub Website: <https://mglezglez.github.io/IntroToProg-Python-Mod07/>

PICKLE MODULE AND EXCEPTION HANDLING IN PYTHON:

# Introduction

This week, as part of Module 7 in Foundation of Programming (Python) course, I learned how to serialize and deserialize objects in Python using the Pickle module and how to handle exceptions gracefully in a program to communicate the details of an execution error in a more informative and human friendly way to the user. In this document, I will present briefly some theoretical foundations on how to work with the Pickle module in Python and how to handle exceptions that could occur while working with this module as well as other operations that could raise them. Then, I will explain how I created a script that puts all the theory to work.

# Exceptions in Python

Exceptions should not be confused with Syntax Errors. Syntax errors take place when the parser detects an incorrect statement. For example:

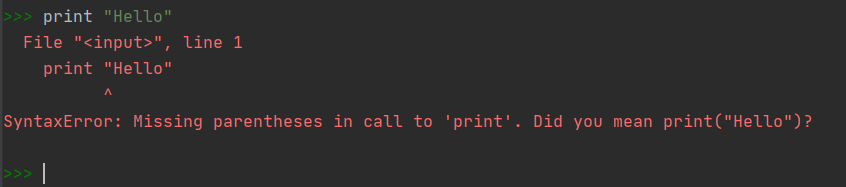


Figure 1. Syntax Error example

The arrow is used to indicate where the parser found the syntax error. In this example, the brackets in the print statement were missing.

Exceptions occur whenever syntactically correct Python code results in an error. The last line of the traceback message will indicate the type of exception error that the program runs into. When this happens (as shown in Figure 2), the program will come to halt and will not continue. In this case, Python was not able to cast a non-numeric string into an integer.

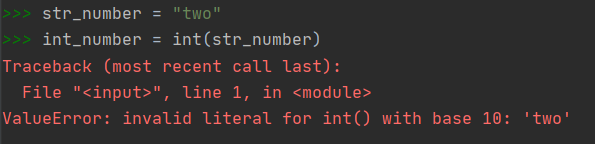


Figure 2. ValueError Exception while casting str to int

Python comes with several built-in exceptions like the one shown in Figure 2, but it also allows you to create self-defined exceptions by inheriting from the base Exception class.

There are two ways to throw Exceptions in Python. Figure 3 contains examples for both. You can throw an exception if a certain condition occurs by using the keyword ***“raise”.*** On the other hand, you can assert that a certain condition is met. If the condition is True, then the program continues running, otherwise, the program throws an ***AssertionError*** exception.

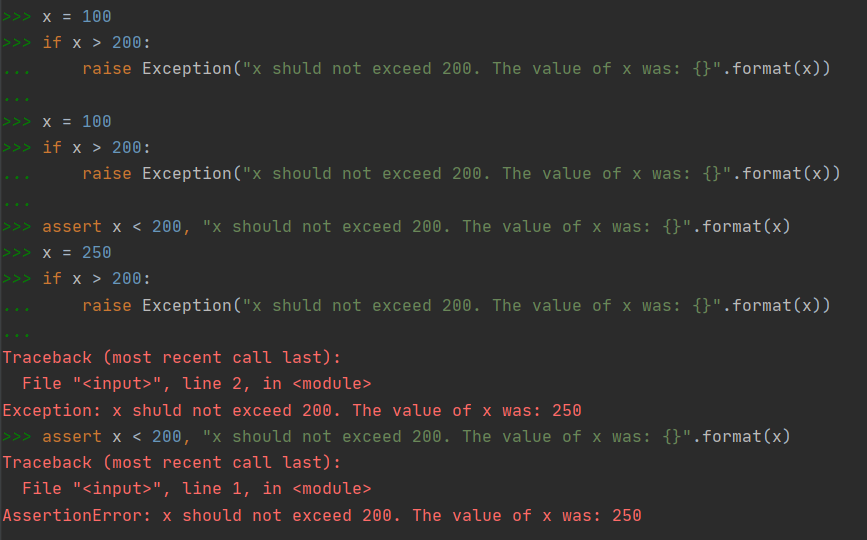


Figure 3. Throwing Exceptions in Python

In order to catch these exceptions in Python and handle them gracefully in order for the program not to crash, Python provides the try/except/else/finally block as shown in Figure 4.

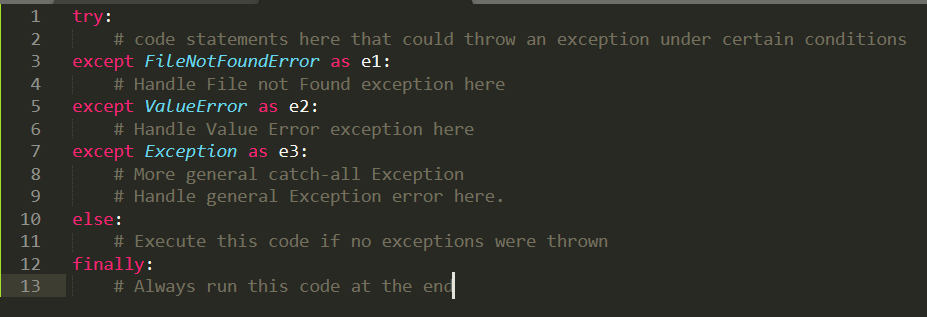


Figure 4. Handling Exceptions in Python

In a nutshell:

* A ***try*** block is executed up until the point where the first exception is encountered
* The ***except*** block is executed if and only if an exception ocurrs. This is the section of code where you determine how the program respond to the error.
* You can anticipate multiple exceptions and differentiate how the program should respond to them.
* The ***else*** block (optional) is executed if and only if no exceptions were thrown.
* The ***finally*** block (optional) always executes if present, regardless of whether there were exceptions caught in the except block or not. You could use this block to implement some final cleanup that would always be needed at the end of this code section.

# Serialization with Pickle in Python

The serialization process allows converting a data structure into a stream of bytes that can be save to a disk, a database or sent over the network (Internet). The reverse process, called deserialization, takes a stream of bytes and converts them back into a data structure.

Python offers three different modules to serialize and deserialize objects: marshal, json and pickle.

Which one to use?

* Don’t use the ***marshal*** module. It exists mainly to read and write the compiled bytecode of Python modules, or the .pyc files you get when the interpreter imports a Python module. It’s used mainly by the interpreter, and the official documentation recommends not to use it.
* The ***json*** module is the newest of the three and it is a good choice if you need interoperability with different languages or a human-readable format. The json modules allows you to serialize/deserialize the following data types: bool, dict, int, float, list, string, tuple, None.
* The ***pickle*** module is a better choice for all the remaining use cases. If you don’t need a human-readable format or a standard interoperable format, or if you need to serialize custom objects, then go with pickle. It’s also faster and it works with many more Python types right out of the box, including your custom-defined objects.

The Python pickle module basically consists of four methods:

* pickle.dump(obj, file, protocol=None, \*, fix\_imports=True, buffer\_callback=None)
* pickle.dumps(obj, protocol=None, \*, fix\_imports=True, buffer\_callback=None)
* pickle.load(file, \*, fix\_imports=True, encoding="ASCII", errors="strict", buffers=None)
* pickle.loads(bytes\_object, \*, fix\_imports=True, encoding="ASCII", errors="strict", buffers=None)

The first two methods are used in the pickling process (serialization) while the last two methods are used in the unpickling process (deserialization). The difference between dump() and dumps() is that the former push the serialization result to a file, whereas the latter returns a string containing the serialization result. The same concept applies to load() and loads(). The first one reads a file to start the deserialization process, while the second one operates on a string.

Even though the pickle module can serialize many more types than the json module, not all types are picklable. The list of unpicklable objects include generators, inner classes, nested functions, lambda functions, defaultdicts, database connections, network sockets, running threads and others.

What about Exceptions?

There are three primary exceptions defined in the Pickle module:

* pickle.PickleError: This is just the base class for the other exceptions. This inherits from Exception (base class)
* pickle.PicklingError: Raised when an unpicklable object is encountered
* pickle.UnpicklingError: Raised during unpickling of an object, if there is any problem (such as data corruption, access violation, etc)

# Program

The program created for this assignment reuses many components created in previous assignments and integrates the use of the Pickle module to handle serialization and deserialization tasks and exception handling.

## The main program:

Figure 5 shows the main body of the program. As you can see, the program initially loads a list of customers to memory from a file on disk. Each customer is represented with a dictionary. Then the program presents the user with a menu of options that allows the user to add new customers, remove existing customers, serialize the list of customers to a file, deserialize the list of customers from file and re-load it into memory and exit the program.

## Serialization/Deserialization in Processor class

In this assignment, I modified the Processor class from Assignment 6 in order to use the Pickle module to serialize and deserialize the list of customers to a file. Figure 6 shows the static methods within the Processor class that were modified in order to use Pickle. As you can see, these methods handle possible exceptions that could be thrown if, for example, you try to serialize an object that is not picklable, using the pickle.dump() function, or if you attempt to deserialize a corrupted file that might not contain picklable data using the pickle.load() function.



Figure 5. Main Program

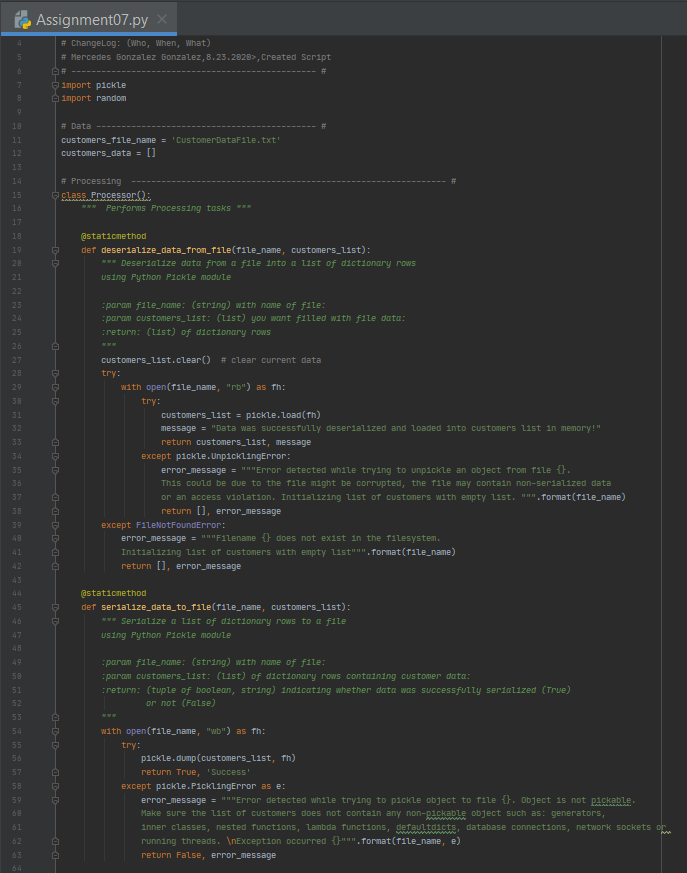


Figure 6. Serialization and Deserialization methods in Processor Class

# Testing the Main Program

Three tests were conducted to validate the main program worked as expected.

Test # 1:

Validates that:

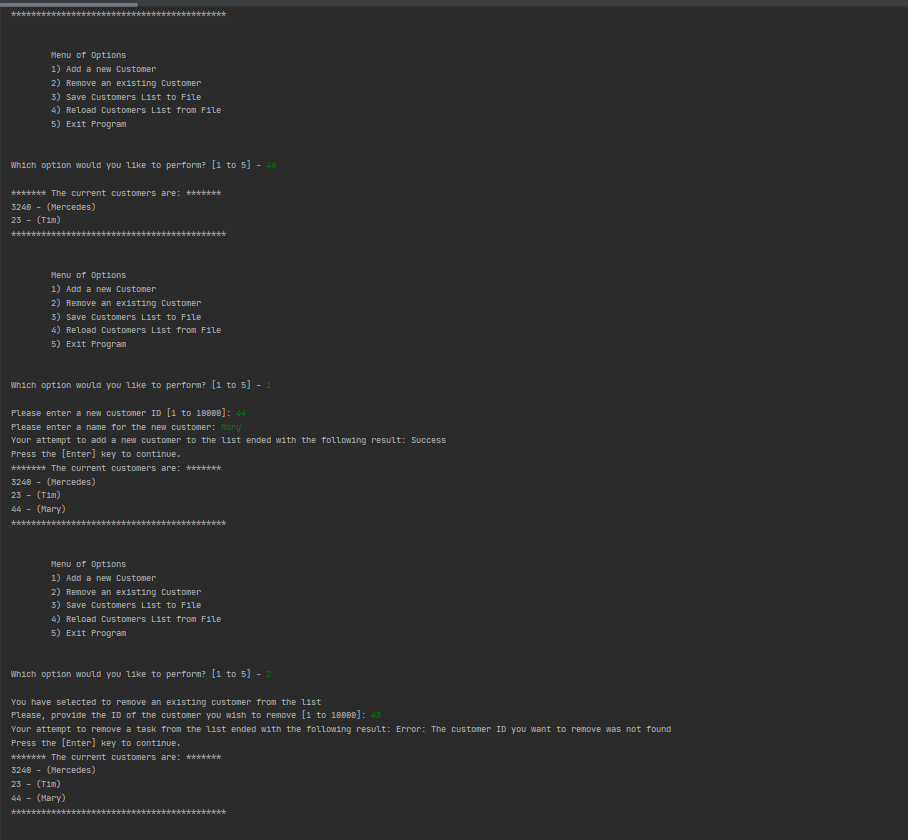
* Initially there is no file so an empty list of customer is returned
* If the user tries to provide an invalid customer ID, he/she is presented with an error indicating that a random customer ID was created. Further validation was added to avoid duplicate customer IDs, but testing this is very difficult due to the random nature of the randonint() function and the fact that the range is very large to hit this validation case.
* A legit combination of customer ID and Name is successfully added to the list of customers.



Test # 2:

Validates that:

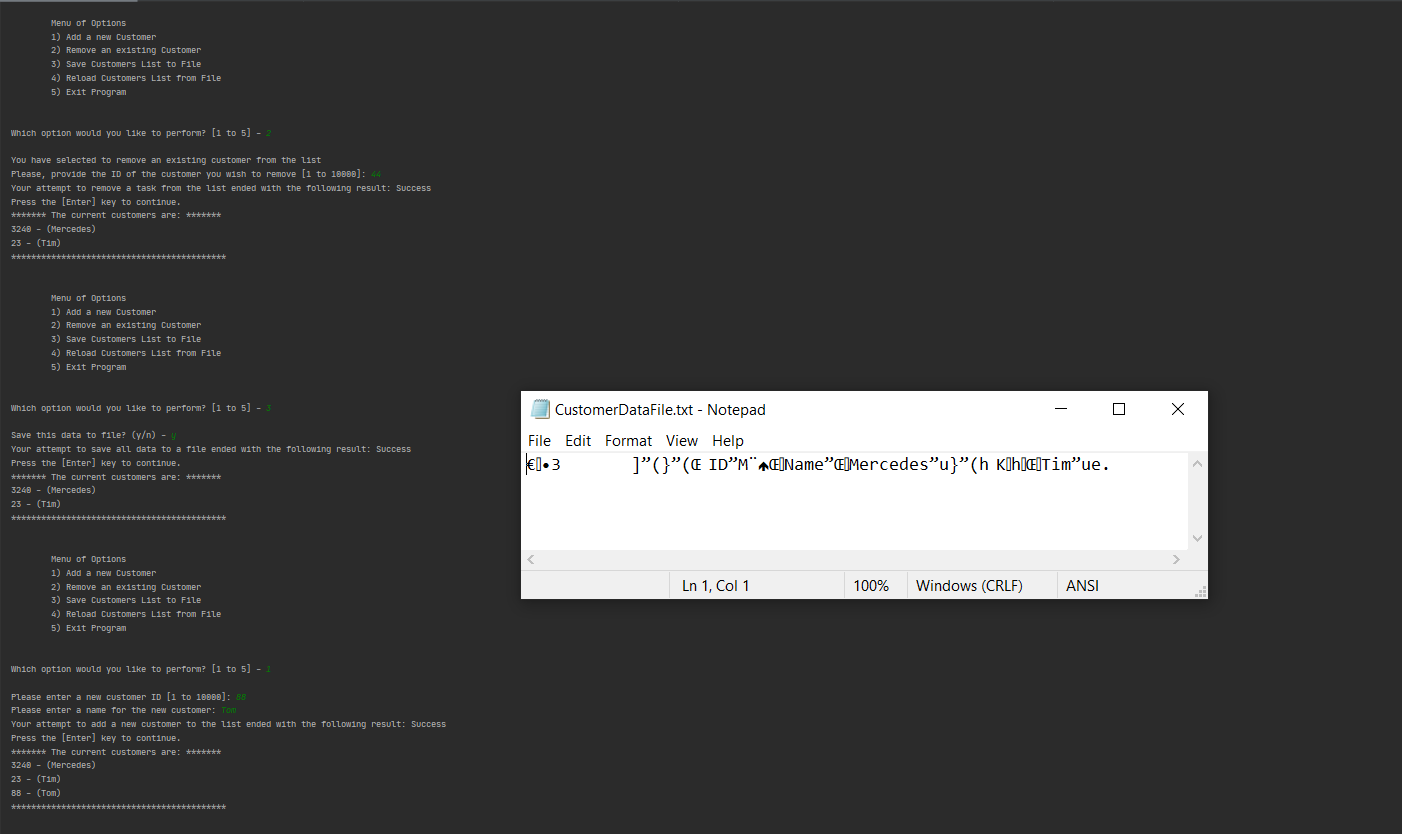
* A customer ID that does not exist cannot be removed from the list of customers and an error is shown to the user to inform that.

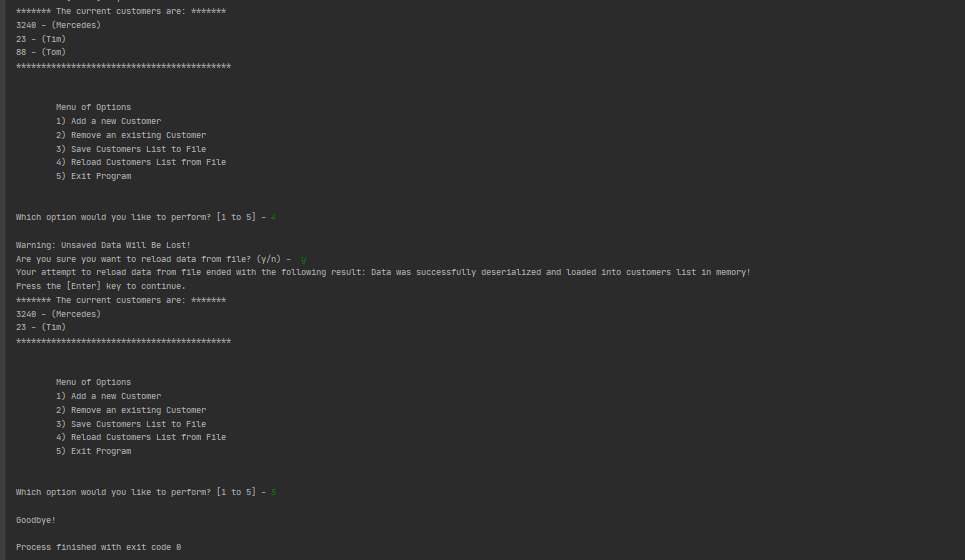


Test # 3:

Validates that:

* List of customers(data) is successfully serialized to a file
* Data is successfully deserialized from the file.





# Summary

The pickle module, which can be used to serialize/deserialize Python objects to/from files. It is a quick and easy way to transfer and store Python objects, which helps programmers to store data easily and quickly for data transfer. Exception handling is paramount when dealing with errors in your code, such as, those related when pickling and unpickling objects with the Pickle module.