**Module 7: Files and Exceptions**

GitHub Repo: <https://github.com/einsteinboricua/IntroToProg-Python-Mod07>

# Introduction

Week 7’s module introduces students to Python’s Pickle function as well as to error handling. Python’s Pickle function allows data to be serialized into a binary stream and read by another program. Meanwhile, with error handling, a programmer can attempt to “protect” the program from failing by instead “catching” a known potential error and “handling” it. A programmer can print a custom message letting the user know why the program failed in the first place or bring the program to a close in a graceful manner rather than a crash.

# Course Content

The course content this week was minimal. Students were introduced to Python’s Pickle function as well as how to handle exceptions. The author has worked with exception handling as part of his experience with Java which also allows the user to “try” and “catch” exceptions, either by defining an exception, explicitly calling out known exceptions, or attempting a wide, “catch any exception” as Exception e. With Python, the author noticed he also had similar flexibility.

Pickling data, however, is a different thing altogether, as the author is not aware of any languages that offer similar functionality. Doing further research, however, the closest the author has seen is Marshal, as a means to share data between C and C++. The author didn’t understand too much what the benefit of pickling is, besides serializing data, but perhaps it’s because this is just a controlled setting, where we show that data is serialized into binary then read back.

# Assignment

For the assignment, students were tasked with creating a script that shows how exception handling works and how pickling works. The author went perhaps too far with the exception handling, attempting to demonstrate a few exceptions, and ended up creating a second script to demonstrate the Pickle functionality separately.

## Exception Handling

The author selected five errors (exceptions) to showcase:

1. IndexError – thrown when attempting to access an index that does not exist on a list
2. TypeError – thrown when there is a type mismatch (like a character instead of a number)
3. ZeroDivisionError – thrown when a number is divided by zero
4. ValueError – thrown when there’s an error in the math domain (like a square root of a negative number)
5. IOError – thrown when the program attempts to work with a file that does not exist

The way the script is built requires the user to follow the prompts (mostly just pressing Enter); it also assumes that the user is entering valid data (example: entering a number when told) and shows the user what happens when an exception is thrown. A custom function was defined to eliminate redundant code (mostly what Python would print after throwing the exception). Figure 1 shows the script being executed on VS Code.

Text

Description automatically generated

## Figure 1 ErrorHandling script execution

### Pickle

For the Pickle function, the author decided to define four different data types:

1. An integer
2. A string
3. A list
4. A dictionary

The script was built in the same manner as the ErrorHandling script: all the user must do is press Enter when prompted. As all the work happens behind the scenes, the user won’t see the pickling of the data until the file is generated and can be reviewed. Figure 2 shows the Pickling script being executed.

Text

Description automatically generated

## Figure 2 Pickling script execution

Below are the full outputs of both scripts on VS Code and Terminal.



## File 1: VS Code Output



## File 2: Terminal output

# Summary

This week’s module was short but introduced the author to the concept of Pickle. The author is eager to see where Pickle can be used in an effective manner but is content with having learned about Python’s many items. Exception handling was familiar to the author, so this week served to understand how Python declares it (namely the try-except as opposed to try-catch).