 Zefeng Shen

August 23, 2020

IT FDN 110A

Assignment07

<https://github.com/zzsocool/ITFnd100-Mode07>

Binary File and Error Handling

**Introduction:**

In this week, I learned about how to create and unpack binary files and how to use expect and extract the error messages and custom the error messages to display to the user. I also learned a little more about how to edit my Github page.

**File Mode:**

This will open a text file called "TestingText" in read-only mode. Take note that only the filename parameter was specified, this is due to the "read" mode being the default mode for the open function.

The access modes available for the open() function are as follows:

* r: Opens the file in read-only mode. Starts reading from the beginning of the file and is the default mode for the open() function.
* rb: Opens the file as read-only in binary format and starts reading from the beginning of the file. While binary format can be used for different purposes, it is *usually* used when dealing with things like images, videos, etc.
* r+: Opens a file for reading and writing, placing the pointer at the beginning of the file.
* w: Opens in write-only mode. The pointer is placed at the beginning of the file and this will overwrite any existing file with the same name. It will create a new file if one with the same name doesn't exist.
* wb: Opens a write-only file in binary mode.
* w+: Opens a file for writing and reading.
* wb+: Opens a file for writing and reading in binary mode.
* a: Opens a file for appending new information to it. The pointer is placed at the end of the file. A new file is created if one with the same name does not exist.
* ab: Opens a file for appending in binary mode.
* a+: Opens a file for both appending and reading.
* ab+: Opens a file for both appending and reading in binary mode.

(Stackabouse, <https://stackabuse.com/file-handling-in-python/>) (External website)

**Binary file:**

A binary file is a [file](https://www.webopedia.com/TERM/F/file.html) stored in [binary format](https://www.webopedia.com/TERM/B/binary_format.html). A binary file is [computer](https://www.webopedia.com/TERM/C/computer.html)-readable but not human-readable. All executable [programs](https://www.webopedia.com/TERM/P/program.html) are stored in [binary files](https://www.webopedia.com/TERM/B/binary_file.html), as are most numeric [data](https://www.webopedia.com/TERM/D/data.html) files. In contrast, [text files](https://www.webopedia.com/TERM/T/text_file.html) are stored in a form (usually [ASCII](https://www.webopedia.com/TERM/A/ASCII.html)) that is human-readable.

(Webopedia,<https://www.webopedia.com/TERM/B/binary_file.html#:~:text=A%20binary%20file%20is%20a,)%20that%20is%20human%2Dreadable.>) (External website)

**Pickle:**

The [pickle](https://docs.python.org/3/library/pickle.html#module-pickle) module implements binary protocols for serializing and de-serializing a Python object structure. “Pickling” is the process whereby a Python object hierarchy is converted into a byte stream, and “unpickling” is the inverse operation, whereby a byte stream (from a [binary file](https://docs.python.org/3/glossary.html#term-binary-file) or [bytes-like object](https://docs.python.org/3/glossary.html#term-bytes-like-object)) is converted back into an object hierarchy. Pickling (and unpickling) is alternatively known as “serialization”, “marshalling,” [1](https://docs.python.org/3/library/pickle.html#id7) or “flattening”; however, to avoid confusion, the terms used here are “pickling” and “unpickling”.

 The pickle module is not secure. Only unpickle data you trust. It is possible to construct malicious pickle data which will execute arbitrary code during unpickling. Never unpickle data that could have come from an untrusted source, or that could have been tampered with. Consider signing data with [hmac](https://docs.python.org/3/library/hmac.html#module-hmac) if you need to ensure that it has not been tampered with. Safer serialization formats such as [json](https://docs.python.org/3/library/json.html#module-json) may be more appropriate if you are processing untrusted data

(Docpython,<https://docs.python.org/3/library/pickle.html#:~:text=%E2%80%9CPickling%E2%80%9D%20is%20the%20process%20whereby,back%20into%20an%20object%20hierarchy.>) (External website)

**Try and Except:**

How try() works?

* First try clause is executed i.e. the code between try and except clause.
* If there is no exception, then only try clause will run, except clause is finished.
* If any exception occured, try clause will be skipped and except clause will run.
* If any exception occurs, but the except clause within the code doesn’t handle it, it is passed on to the outer try statements. If the exception left unhandled, then the execution stops.
* A try statement can have more than one except clause

(Geekssforgeeks,<https://www.geeksforgeeks.org/python-try-except/>) ( External website)

**Built-in Exception**

The following table lists important built-in exceptions in Python.

Exception Description

AssertionError Raised when the assert statement fails.

AttributeError Raised on the attribute assignment or reference fails.

EOFError Raised when the input() function hits the end-of-file condition.

FloatingPointError Raised when a floating point operation fails.

GeneratorExit Raised when a generator's close() method is called.

ImportError Raised when the imported module is not found.

IndexError Raised when the index of a sequence is out of range.

KeyError Raised when a key is not found in a dictionary.

KeyboardInterrupt Raised when the user hits the interrupt key (Ctrl+c or delete).

MemoryError Raised when an operation runs out of memory.

NameError Raised when a variable is not found in the local or global scope.

NotImplementedError Raised by abstract methods.

OSError Raised when a system operation causes a system-related error.

OverflowError Raised when the result of an arithmetic operation is too large to be represented.

ReferenceError Raised when a weak reference proxy is used to access a garbage collected referent.

RuntimeError Raised when an error does not fall under any other category.

StopIteration Raised by the next() function to indicate that there is no further item to be returned by the iterator.

SyntaxError Raised by the parser when a syntax error is encountered.

IndentationError Raised when there is an incorrect indentation.

TabError Raised when the indentation consists of inconsistent tabs and spaces.

SystemError Raised when the interpreter detects internal error.

SystemExit Raised by the sys.exit() function.

TypeError Raised when a function or operation is applied to an object of an incorrect type.

UnboundLocalError Raised when a reference is made to a local variable in a function or method, but no value has been bound to that variable.

UnicodeError Raised when a Unicode-related encoding or decoding error occurs.

UnicodeEncodeError Raised when a Unicode-related error occurs during encoding.

UnicodeDecodeError Raised when a Unicode-related error occurs during decoding.

UnicodeTranslateError Raised when a Unicode-related error occurs during translation.

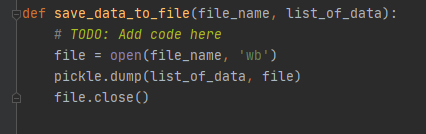
ValueError Raised when a function gets an argument of correct type but improper value.

ZeroDivisionError Raised when the second operand of a division or module operation is zero.

(Tutorialsteacher,<https://www.tutorialsteacher.com/python/error-types-in-python>) (External website)

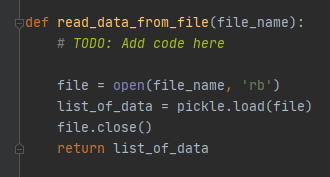
**Script:**

For this week assignment, I started with creating a binary file. Binary file mode was different than normal text file. For writing mode, I input “wb”. For writing a binary file, I used pickle.dump() as shown in figure 1.

****

**Figure 1:** Save data to a binary file.

For read mode, I input “rb”. For reading a binary, I used pickle.load() as shown in figure 2.

****

**Figure 2:** Read data from binary file.

By following professor’s template, I took user’s input into a list and use called out save data function and read data function to complete first part of the assignments as show in figure 3.

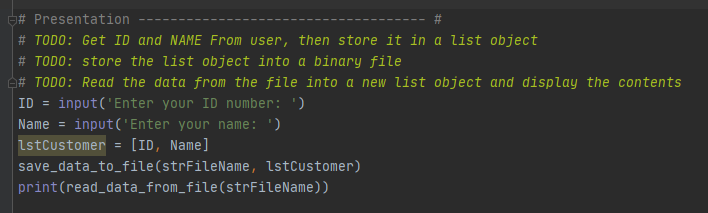
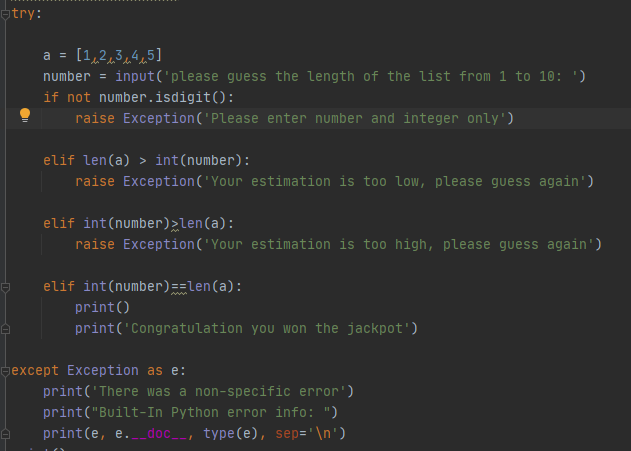


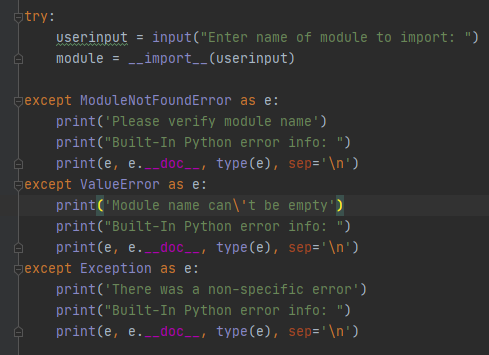
Figure 3: Part one pickle demo main body code.

For part 2 error handling. I started with guess the length of a list from 1 to 10. I created a custom error message to tell the users if their input is wrong or not. If the input is not an integer, they will get error message to tell them enter number and integer only. If input is smaller than 5, they will be told the number is too low. If the input is bigger than 5, they will be told the number is too high as shown in figure 4.



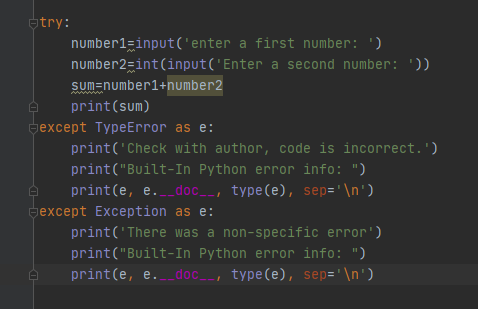
**Figure 4:** Custom error message**.**

Next, I found a built-in error call ModuleNotFoundError. So, I let the users to type the name of module to import. If they enter the name of the module did not exist in the system, it will trigger the error message. If the user did not enter anything, it will trigger ValueError as show in figure 5.



**Figure 5:** ModuleNotFoundError Demo.

I intentionally made a mistake that I had encountered before in the third demo: TypeError. I concatenated “str” to “int”. If two numbers are input, it will automatically trigger the TypeError. If two letters are input, it will display ValueError as shown in figure 6.



**Figure 6:** TypeError Demo.

**Github:**

By following video tutorial, I created an advanced Github web page. I had a same issue that the picture did not display as well. After following the troubleshooting steps, the picture finally displayed correctly as show in figure 7.



**Figure 7**: Github web page.

**Summary:**

After researching about error handling, I get more familiar and understand what do the error messages mean and what do I need to change about the code. Binary file is not for human to read. It is not safe to open any unknown source binary file. There may be a virus hide inside of the file.