Deborah Yenubari

03/4/2021

Fundamentals of Programming (Python)

# Assignment 7

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# **Functions, Pickling and Structured Error Handling in Python**

There are two main types of functions in the Python Programming language:

Library Functions: Built-in functions in Python Programming Language called a Library function.

We don’t have to define these functions to use them; we can directly call them. These functions are called built in functions. The very basic built in function that we have used in almost every module is the print() function which is used to display a given object

# Some Python Built in Functions

abs() returns the absolute value of a number, and the returned value is always positive.

chr() returns a character (a string) from an integer.

dict() is used for creating a Python dictionary.

input() usually reads and returns a line of string.

int() returns an integer from a number or a string.

len() returns the length of an object.

list() is used for creating a Python list.

max() returns the largest element.

min() returns the smallest element. tuple() Function This function creates a tuple. staticmethod() This function creates a static method from a function.

Import\_\_() is an advanced function called by ‘import’.

To use these built in functions, we have only to call them and pass the relevant argument

User Defined Functions: Instead of relying on built-in functions, Python programming language allows us to create our own functions called as user defined functions. For example, if we want to implement some mathematical calculations, then put them in separate functions with the correct function name. Then we can call that function multiple times.

Table: 1

# File vs Binary File – Difference between Text File and Binary File

|  |  |
| --- | --- |
| Text File | Binary File |
| Bits represent character | Bits represent custom data |
| Can only store plain text | Can store different types of data(image, audio, text) in a single file |
| Widely used file format , opened using simple text editor | Developed especially for an application and may not be understood by other applications |
| Most .txt and .rtf are used as extensions to text files | Can have any application defined extension |
| Less prone to get corrupt as changes reflect as soon as the file is opened and can easily be undone | Can easily get corrupted, even a single bit change may corrupt the file. |

# **Text Files**

Text files are special subset of binary files that are used to store human readable characters as a rich text document or plain text document. Text files also store data in sequential bytes but bits in text file represents characters.

Text files are less prone to get corrupted as any undesired change may just show up once the file is opened and then can easily be removed..

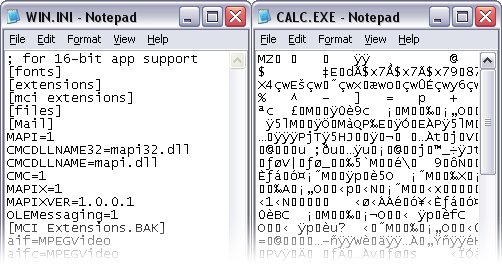
**Binary File**

Binary file are files that store data in the form of sequence of bytes grouped into eight bits or sometimes sixteen bits. These bits represent custom data and such files can store multiple types of data (images, audio, text, etc) under a single file.

Binary file can have custom file formats and the developer, who designs these custom file formats, converts the information, to be stored, in bits and arranges these bits in binary file so that they are well understood by the supporting application and when needed, can easily be read by the supporting application.

One example of binary file is image file is .PNG or .JPG. If one tries open these files using a text editor then, he/she may get unrecognizable characters, but when opened using the supporting image viewer, the file will be shown as a single image. This is because the file is in binary format and contains data in the form of sequence of bytes. When the text editor tries to read these bytes and tries to convert bits into characters, they get undesired special characters and display it to the user.

Binary files also store file information like file name, file format, etc., which may be included in the file as header to the file and is visible even when the file is opened in a text editor.

Since binary files store data in sequential bytes, a small change in the file can corrupt the file and make it unreadable to the supporting application.

Binary File opened using Notepad

**Pickling and Unpickling**

# Python provides pickle modules for Serialization and de-Serialization of python objects like lists, dictionaries, tuples, etc. Pickling is also called marshaling or flattening in other languages. Pickling is used to store python objects.

# Serialization or Pickling: Pickling or Serialization is the process of converting a Python object (lists, dict, tuples, etc) into byte streams that can be saved to disks or can be transferred over a network.

# De-serialization or un pickling: The byte streams saved on file contains the necessary information to reconstruct the original python object. The process of converting byte streams back to python objects is called de-serialization.

**What is Exception?**

"Exception" is a built-in python class used to hold information about an error. Python automatically creates an Exception object when an error occurs. The Exception object automatically fills with information about the error that caused the exception. 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.

In Python, all exceptions must be instances of a class that derives from BaseException. In a ‘try’ statement with an ‘except’ clause that mentions a particular class, that clause also handles any exception classes derived from that class

When a Python script raises an exception, it must either handle the exception immediately otherwise it terminates and quits.

exception ArithmeticError

The base class for those built-in exceptions that are raised for various arithmetic errors: OverflowError, ZeroDivisionError, FloatingPointError.

exception ModuleNotFoundError

A subclass of ImportError which is raised by import when a module could not be located. It is also raised when None is found in sys.modules.

exception IndexError

Raised when a sequence subscript is out of range. (Slice indices are silently truncated to fall in the allowed range; if an index is not an integer, TypeError is raised.)

exception TabError

Raised when indentation contains an inconsistent use of tabs and spaces. This is a subclass of IndentationError.

exception ValueError

Raised when an operation or function receives an argument that has the right type but an inappropriate value, and the situation is not described by a more precise exception such as IndexError.

exception ZeroDivisionError

Raised when the second argument of a division or modulo operation is zero. The associated value is a string indicating the type of the operands and the operation.

# **Handling an exception**

You can capture the Exception object in the except section of a try-except block and extract the error messages

If you have some suspicious code that may raise an exception, you can defend your program by placing the suspicious code in a try: block. After the try: block, include an except: statement, followed by a block of code which handles the problem as elegantly as possible.

-syntax of try....except...else blocks −

try:

operations here;

......................

except ExceptionI:

If there is ExceptionI, then execute this block.

except ExceptionII:

If there is ExceptionII, then execute this block.

......................

else:

If there is no exception then execute this block.

An exception can be a string, a class or an object. Most of the exceptions that the Python core raises are classes, with an argument that is an instance of the class.

Example

Following is an example for a single exception −

# Define a function here.

def temp\_convert(var):

try:

return int(var)

except ValueError, Argument:

print "The argument does not contain numbers\n", Argument

# Call above function here.

temp\_convert("xyz");

This produces the following result −

The argument does not contain numbers

invalid literal for int() with base 10: 'xyz'

"Common base class for all non-exit exceptions."

This message comes from the Exception class's docstring. It tells that the Exception class is a "base" class that can be used to create "derived" classes. Derived classes "inherit" data and functions from the base class that can be replaced and customized.

This script shows two classes that derive code from the Exception class and replace the"\_\_str\_\_()" functionwith a custom message.

class CustomError(Exception):

""" Some custom error info in the DocString """

def \_\_str\_\_(self):

return 'Some custom error message'

class FileNotTXTError

(Exception):""" File extension must end with txt to indicate it is a text file """

def \_\_str\_\_(self):

return 'File extension not txt'

Raising an Exception

You can raise exceptions in several ways by using the raise statement. The general syntax for the raise statement is as follows.

try:

new\_file\_name = input("Enter the name of the file you want to make: ")

if new\_file\_name.is numeric():

raise Exception('Do not use numbers for the file\'s name')

elif new\_file\_name.ends with('txt') == False:

raise FileNotTXTError()

else:

raise CustomError()

except Exception as e:

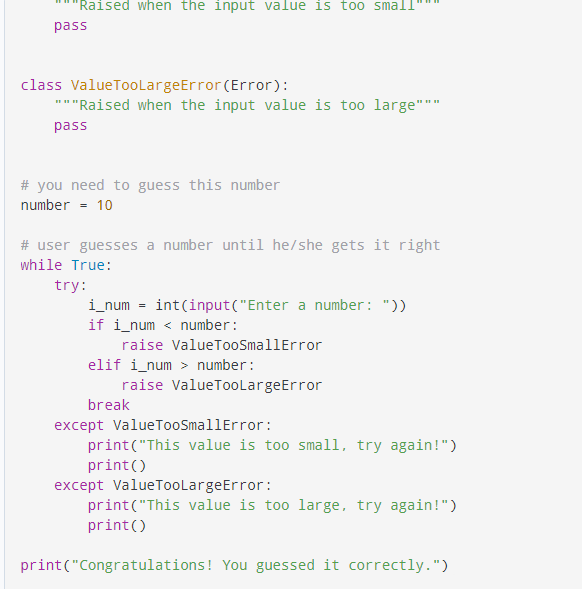
print("There was a non-specific error!")

print("Built-In Python error info: ")

print(e, e.\_\_doc\_\_, type(e), sep='\n')

**In Python, users can define custom exceptions by creating a new class. This exception class has to be derived, either directly or indirectly, from the built-in Exception class.**

**Two exceptions (ValueTooSmallError and ValueTooLargeError) that are actually raised by the program (below) are derived from a base class called Error, this is the standard way to define user-defined exceptions in Python programming,**

****

**Source:** [How to Define Custom Exceptions in Python? (With Examples) (programiz.com)](https://www.programiz.com/python-programming/user-defined-exception)

**What is the Markdown language?**

Markdown is a simple syntax that formats text as headers, lists, boldface, and so on. Adding bold, italics, numbered lists, bullet points, headings, and so on to text, you’re “formatting” it. Markdown is a syntax—or, set of rules—that formats text on web pages.

Traditionally, to format text on web pages, we use Hypertext Markup Language, better known as HTML. HTML is one member of the family of markup languages, along with eXtensible Markup Language (XML) and Standard Generalized Markup Language (SGML).

[To format text with HTML, you put tags around the text. For example, if you want to bold text, you type “<b> this is some bold text </b>”.

The tags, however, make it quite difficult for humans to read HTML and understand how the text will look after the computer renders it. It’s not very “user-friendly” for people who don’t have a lot of experience reading it.

Markdown, on the other hand, is meant “to be as easy-to-read and easy-to-write as is feasible.” In short, Markdown makes it easier to format text for web pages because its tags are simpler than HTML, and they convert to HTML automatically. This means you don’t have to know HTML to write something for a web page because Markdown translates your tags into HTML for you.

How Do You Use Markdown?

To use Markdown, you just apply simple tags to your text. For example, to format text in italics, you put underscores around it like so: \_this is some text in italics\_.

Strong (Bold)

\*\*text\*\*

\_\_text\_\_

Markdown converts both \*\*text\*\* and \_\_text\_\_ to the HTML <strong> tag, which typically appears as bolded text. In other words, for additional emphasis, use double \* or \_ characters.

After you format your text, an application has to convert it to HTML, which is usually done automatically. For example, README files in GitHub use Markdown, and as long as they have a .MD file extension, GitHub automatically converts them to the correct HTML tags when they’re published.

So, in most cases, you won’t have to do this yourself, but if you do, there’s a Markdown tool available.

## Where Can You Use Markdown?

As we mentioned above, you can use Markdown on GitHub, but also on Reddit, StackOverflow, and other websites. If you’ve ever formatted text in WhatsApp messages or Slack conversations, you’ve already used it because these applications use a (very small) subset of Markdown tags to format text.

Markdown is a way to style text on the web. You control the display of the document; formatting words as bold or italic, adding images, and creating lists are just a few of the things we can do with Markdown. Mostly, Markdown is just regular text with a few non-alphabetic characters thrown in, like # or \*.

**Syntax guide**

Here’s an overview of Markdown syntax that you can use anywhere on GitHub.com or in your own text files.

Headers

# This is an <h1> tag

## This is an <h2> tag

###### This is an <h6> tag

Emphasis

\*This text will be italic\*

\_This will also be italic\_

\*\*This text will be bold\*\*

\_\_This will also be bold\_\_

\_You \*\*can\*\* combine them\_

Lists Unordered

\* Item 1

\* Item 2

\* Item 2a

\* Item 2b

Lists Ordered

1. Item 1

1. Item 2

1. Item 3

1. Item 3a

1. Item 3b

Images

![GitHub Logo](/images/logo.png)

Format: ![Alt Text](url)

Links

http://github.com - automatic!

[GitHub](http://github.com)

Block quotes

As Kanye West said:

> We're living the future so

> the present is our past.

Inline code

I think you should use an

`<addr>` element here instead.

GitHub Flavored Markdown

GitHub.com uses its own version of the Markdown syntax that provides an additional set of useful features, many of which make it easier to work with content on GitHub.com.

Tables

You can create tables by assembling a list of words and dividing them with hyphens - (for the first row), and then separating each column with a pipe |:

First Header | Second Header

------------ | -------------

Content from cell 1 | Content from cell 2

Content in the first column | Content in the second column

Would become:

First Header Second Header

Content from cell 1 Content from cell 2

Content in the first column Content in the second column

SHA references

Any reference to a commit’s SHA-1 hash will be automatically converted into a link to that commit on GitHub.

16c999e8c71134401a78d4d46435517b2271d6ac

mojombo@16c999e8c71134401a78d4d46435517b2271d6ac

mojombo/github-flavored-markdown@16c999e8c71134401a78d4d46435517b2271d6ac

Issue references within a repository

Any number that refers to an Issue or Pull Request will be automatically converted into a link.

#1

mojombo#1

mojombo/github-flavored-markdown#1

Username @mentions

Typing an @ symbol, followed by a username, will notify that person to come and view the comment. This is called an “@mention”, because you’re mentioning the individual. You can also @mention teams within an organization.

Automatic linking for URLs

Any URL (like http://www.github.com/) will be automatically converted into a clickable link.

Strikethrough

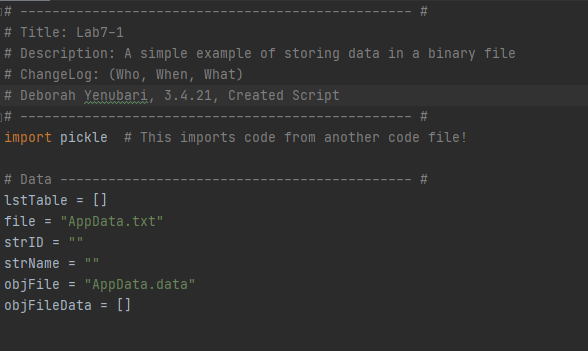
Any word wrapped with two tildes (like ~~this~~) will appear crossed out.

Emoji

GitHub supports emoji!

# **# Title: Lab7-1**

# **# Description: A simple example of storing data in a binary file**



As the script will store data in binary file, we ‘import pickle’ at the start.

Pickling will convert the python object(list) into a byte stream.

Declaring variables and Data

lstTable = [] # or lstCustomer

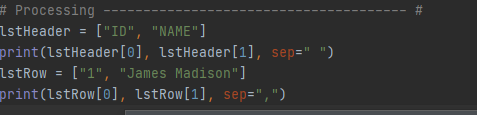
file = ‘AppData.txt’

strID = “

strName = “

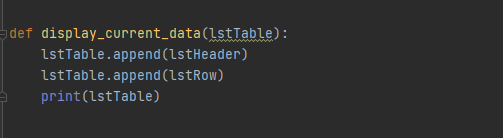
objFile = “AppData.data”

objFileData = []



lstHeader for the lstTable with two values, and printing values using subscript.

first lstRow of data to be included in the lstTable, again printed using subscript or index placeholder.



def display\_current\_data with one parameter lstTable, append lstHeader and each lstRow to lstTable and print lstTable

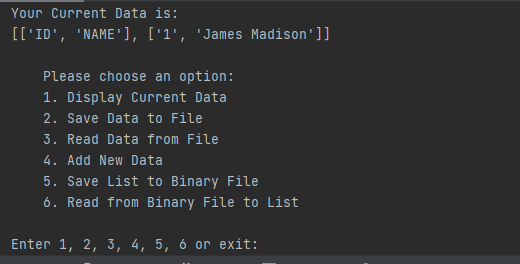
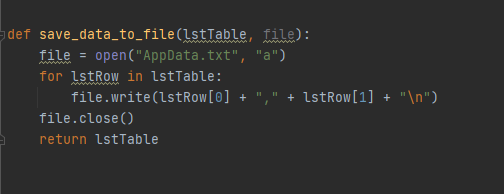


Image 1: Output for def display\_current\_data(lstTable)



The first function used in processing, to save data to a text file,’AppData.txt’, with two parameters, ‘lstTable’ and ‘file’.

After opening a file, for each lstRow in the lstTable, write each value in the lstRow based on index position, into the file. Close the file and return lstTable as a variable.

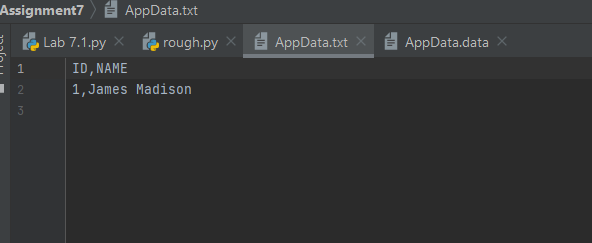


Image 2: Output for def save\_data\_to\_file(lstTable, file)

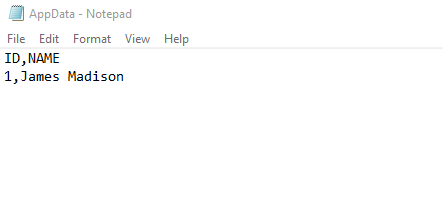
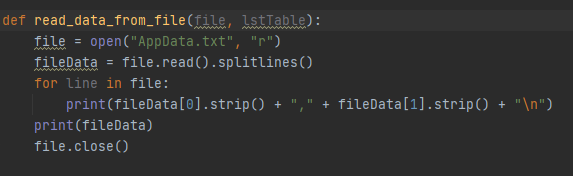


Image 3; Data saved in the text file ‘AppData.txt’



def read\_data\_from\_file(file, lstTable) opens the text file. File Data, on reading lines, for each line in file, print into a list Row, value at [0] first, followed by value at [1].

Print all file data in lstRows and close the file.

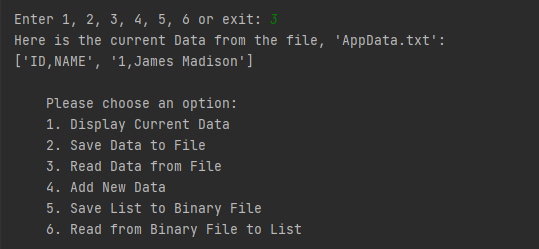
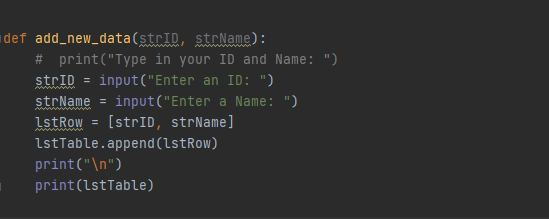


Image 4: Output for def read\_data\_from\_file(file,lstTable)



Function def add\_new\_data with two parameters, strID and strName obtains user input for an ID followed by a Name. The input is displayed in a lstRow and the lstRow appended to the lstTable to print.

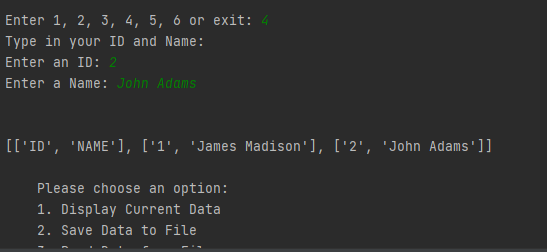
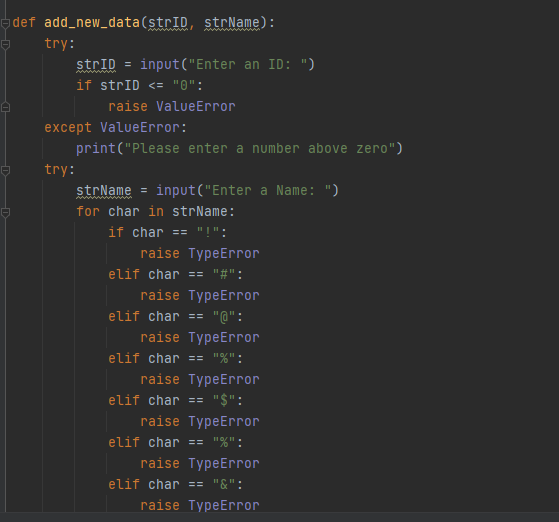


Image 5: Output for def add\_new\_data(strID, strName)



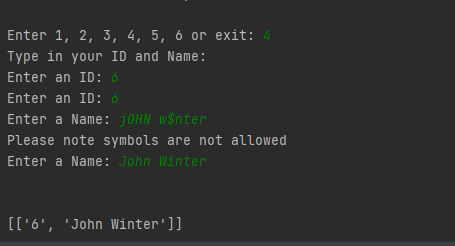
Tried to incorporate structure error handling for user input to enter new data

try: for user input, but if user input equals or is lesser than zero, raise a ValueError subclass.

except: for ValueError, accept user input.

For strName, if any char in strName == (the above symbols), raise a TypeError class. It would be much better to have the symbols as a tuple or list and have one ‘if’ statement,

But for now, used individual statements to let the user know symbols are not allowed.



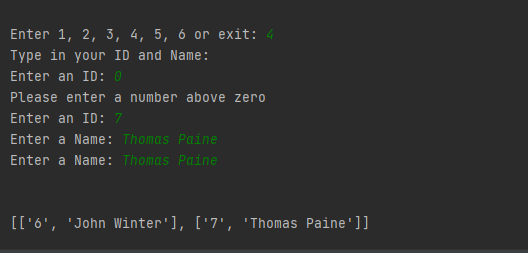
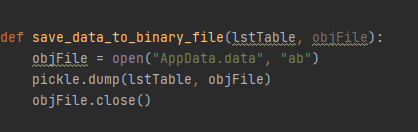


Image 6 & 7: Output for error handling, if a zero is entered as strID or a symbol included in strName.

Would need to work a little more on the script, so either Enter an ID or Enter a Name is not printed twice when the other is entered incorrectly.



def save\_data\_to\_binary\_file with two parameters, the object ‘lstTable’ which must be converted into a byte stream stored in ‘objFile’.

After opening an objFile ‘AppData.data’, mark ‘**ab**’ to append/write binary data, the command line involves pickle to dump the lstTable into objFile, then close objFile.

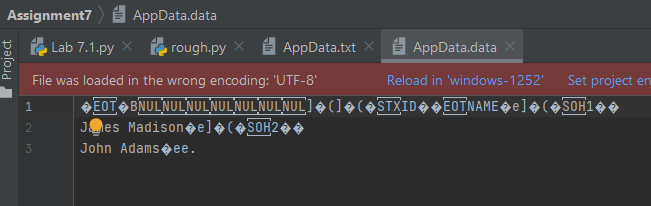
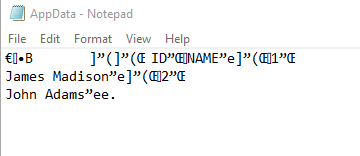
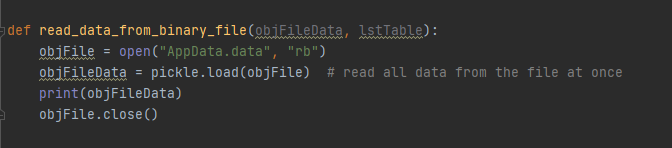


Image 8 & 9: Output for def save\_data\_to\_binary\_file



def read\_data\_from\_binary\_file has two parameters, objFileData to be read from the binary file and converted into a list object, lstTable at output.

Open ‘AppData.data’ , ‘**rb**’ read binary file. Pickle to load data from the parameter ‘objFile’ to a list object, objFileData and print in lstRows. Then close the binary file.

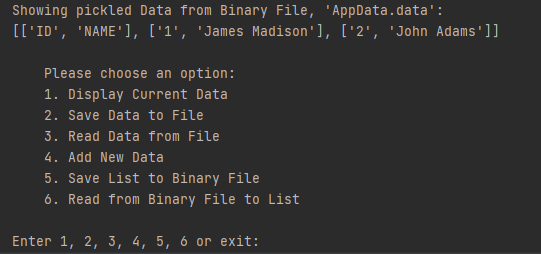


Image 10: Output for def read\_data\_from\_binary\_file(objFileData, lsttable)

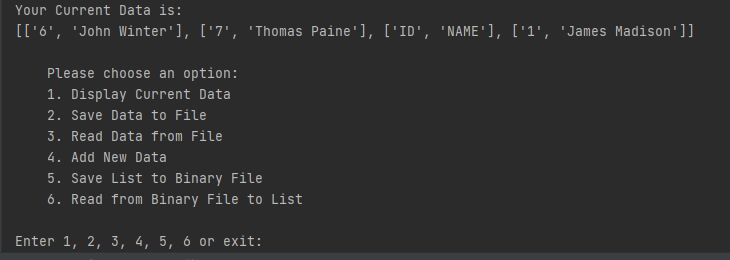


Image 11: Final Output before exit. The order in which lstData is entered into lstTable needs to be rectified in the script.

Conclusion

While mostly successful, the script needs more work in placing lstRows in the order eneterd in the lstTable as well as ‘try’ ‘ except’ ‘finally’ statements in error handling, so if user input is entered correctly, it does not ask for user input again. However, was successful in placing pickling and unpickling code inside functions as well as saving and reading data from a text file and binary file.

**Sources**

1 [What are the Benefits and Limitations of Using Python? | edu CBA](https://www.educba.com/benefits-and-limitations-of-using-python/)

2 [Built-in Functions — Python 3.9.2 documentation](https://docs.python.org/3/library/functions.html)

3 [Built in Function in Python with Examples - Intellipaat](https://intellipaat.com/blog/tutorial/python-tutorial/python-built-in-functions/)

4 [Exception Handling in Python - PythonForBeginners.com](https://www.pythonforbeginners.com/error-handling/exception-handling-in-python)

5 [(Tutorial) Exception and Error Handling in Python - DataCamp](https://www.datacamp.com/community/tutorials/exception-handling-python)

6 [Difference between Text File and Binary File - The Crazy Programmer](https://www.thecrazyprogrammer.com/2018/05/difference-between-text-file-and-binary-file.html)

7 [What are binary and text files? (nayuki.io)](https://www.nayuki.io/page/what-are-binary-and-text-files)

8 [Python - Exceptions Handling - Tutorialspoint](https://www.tutorialspoint.com/python/python_exceptions.htm)

9 [How to Define Custom Exceptions in Python? (With Examples) (programiz.com)](https://www.programiz.com/python-programming/user-defined-exception)

10 [What Is Markdown and How Do You Use It? (howtogeek.com)](https://www.howtogeek.com/448323/what-is-markdown-and-how-do-you-use-it/)

11 [Mastering Markdown · GitHub Guides](https://guides.github.com/features/mastering-markdown/)

12 Module 7 Programming Notes