Kate Stevens

November 20, 2019

IT FDN 100 A

Assignment07

Exception Handling & Pickling

Assignment07: Research and document your knowledge about the use and benefits of the Pickle module and exception handling in Python. The full assignment and further notes following this week’s discussions can be viewed or downloaded here: [Assigment07.pdf](https://canvas.uw.edu/courses/1342958/modules/items/9973247) (external link), [\_Mod7PythonProgrammingNotes.pdf](https://canvas.uw.edu/courses/1342958/modules/items/9973246) (external link).

Intro

The primary focus of week seven of Introduction to Python Programing, consisted of demonstrations on how to work with and configure error handling in Python which included the topics of try/except blocks, exception errors, built-in and custom error exceptions and exception classes, and Python’s Pickling Module. The secondary focus of this week was on learning how to use [basic writing and formatting syntax](https://help.github.com/en/github/writing-on-github/basic-writing-and-formatting-syntax) (external link ) using Jekyll. used to build and present personal and professional scripting projects with GitHub Webpages.

This weeks assignment consists of researching and documenting knowledge on the following bullet point topics and programming methods and presenting our research on GitHub Webpages using Jekyll’s writing and formatting syntax discussed in Mod07 YouTube tutorial found at [Module07 Course Video](https://youtu.be/4IkIdXJBC6o)(external link), starting at 1:30:00.

The following topics will be briefly discussed in this document containing the sources and conclusions of my research pertaining to Assignment07’s research exercise detailed in [Assigment07.pdf](https://canvas.uw.edu/courses/1342958/modules/items/9973247) (external link).

# Exception Handling in Python

# Pickling in Python

*Exception Handling in Python*

#### (Assignment Overview)

#### Search the web for examples of how to use Python’s exception handing features. Make note of the URL for any pages you feel are good at explaining the subject, and why you feel that way.

# Topic overview

Exception handling protocols are written/implemented by a script’s author(s) with the intention of making troubleshooting easier for users or other developers. This type of script editing helps to replace long, incomprehensible trace back error messages (for those who are not developers) with more specific and/or conducive instruction to follow if an error occurs while running a script/program.

These traceback errors can appear with basic user interaction, reading /writing data files and edits or manipulation to the code that prevent the program from running correctly. If traceback messages are not addressed and met with some form of error anticipation/moderation by the script author(s), it will cut back on intended efficiency and productivity for other developers as well as deter others from using, utilizing or running the program all together. To help guide or provide a user or fellow developer with a good experience using your program/script, many programmers will start by refining their code using “if/elif” Error Handling processes.

Error handling is also useful for developers because it provides useful insight to anticipate sections of our code that might throw an error or an exception for other future users down the road.

With this information developers can try to configure error blocks and handle them in the way that we want instead of allowing Python (and/or other programming languages) to auto generate and present a string of obscure and/or vague error messages to the end user.

I located external sources by searching ‘Exception Handling’ key phrases mentioned in [\_Mod7PythonProgrammingNotes.pdf](https://canvas.uw.edu/courses/1342958/modules/items/9973246) (external link, Listing 11-15). The Top two resources I found most useful (other than the [Module07 Course Video](https://youtu.be/4IkIdXJBC6o)(external link)) was in the format of a YouTube Tutorial, [Python Tutorial: Using Try/Except Blocks for Error Handling](https://www.youtube.com/watch?v=NIWwJbo-9_8) (external link) and it’s linked [GitHub](https://www.youtube.com/redirect?q=https%3A%2F%2Fgithub.com%2FCoreyMSchafer%2Fcode_snippets%2Ftree%2Fmaster%2FExceptions&event=video_description&v=NIWwJbo-9_8&redir_token=xVJvI7xsA5txpuMJMcr_Z2-gkJB8MTU3NDQ4MTIxM0AxNTc0Mzk0ODEz) Repository page (external link) posted in the authors information. I appreciated how closely related the examples were to the material discussed in the Mod07 tutorial, as many of the outside resources I come across tend to be either more advanced/specialized modifications or workarounds with limited description.

# Exception Handling Examples:

The Exception Handling examples below describes three ways to implement Python’s Exception handling functionalities modelled in the YouTube Tutorial, [Python Tutorial: Using Try/Except Blocks for Error Handling](https://www.youtube.com/watch?v=NIWwJbo-9_8) (external link).

This tutorial I choose to model the following examples from, stood out to me because it allowed the viewer to experience the narrators fluid and uncut thought process while evolving the code through several stages of performance and efficiency allowing the viewer to "Thought Shadow" every step of his development process and logic .

##### Example:7.1

Reference:

[*Python Tutorial: Using Try/Except Blocks for Error Handling*](https://www.youtube.com/watch?v=NIWwJbo-9_8) (external link), Timestamp: 00:50

Description:

Anticipate areas in your code that may throw an error that will cause a user to see a default traceback error message when ran. Use the reference tutorial to learn how we can handle exceptions in specific ways to look at the control flow of a try/except/else/finally statement to replace default traceback error messages with custom or built-in error messages to help increase the speed and productivity of troubleshooting.

7.1 Output Requirements:

Create a Try/Except code handling block with limited capabilities to notify the developer/user of a FileNotFound error.

##### *7.1 Example overview:*

Example 7.1, starts out with Starter-code (Figure 1) from the YouTube tutorial, referenced above, that has that has two problems to be addressed in order to achieve the desired output requirements (above).

Error 1:

In this case the code instructs to open a file, however the name of the file specified as ‘testfile.txt’ (Figure 1.1, Figure 1.3 red) which does not exist in the project’s directory, because the name of the file or path is wrong which result in a default traceback error message when the code is run.

Error 2:

Figures 1.1 & 1.3 also shows that a Try/except exception block has been accepted in the code but skipped over because the exception is read as a “pass” otherwise known as a placeholder. In order to implement an Exception handler to override Python’s default traceback error message shown in (Figures 1.2 & 1.3), the developer must replace “pass” (shown in Figures 1.1 & 1.3) with a valid exception handler.

7.1 Input “Starter-code” (without exception handler)

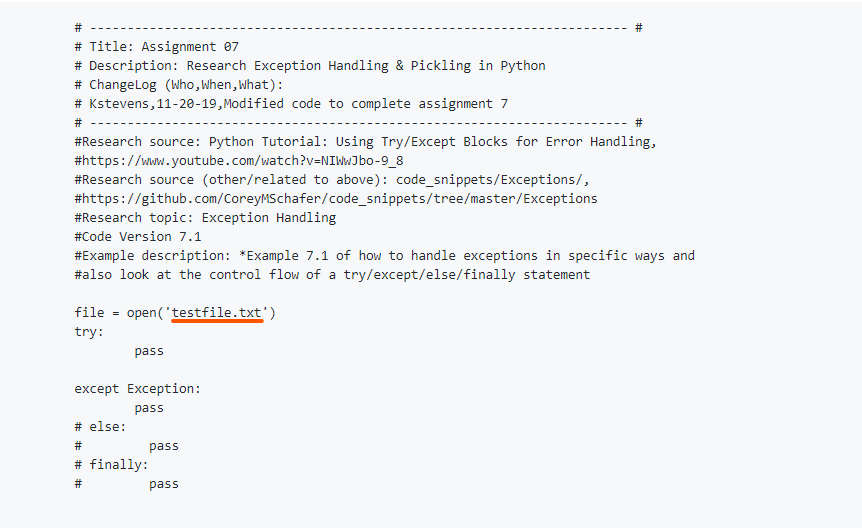


Figure 1.1: User Input with incorrect file name (red) without an active exception handler in place of the “pass” placeholder.

*7.1 Output “Starter-code” (without exception handler)*

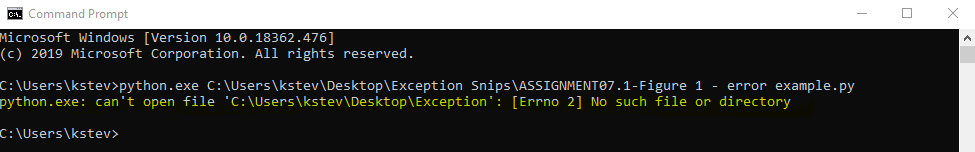


Figure 1.2: Default error message “output” displayed in CMD running code from Figure 1.1.

*7.1 Input / Output “Starter-code” viewed in PyCharm (without exception handler)*

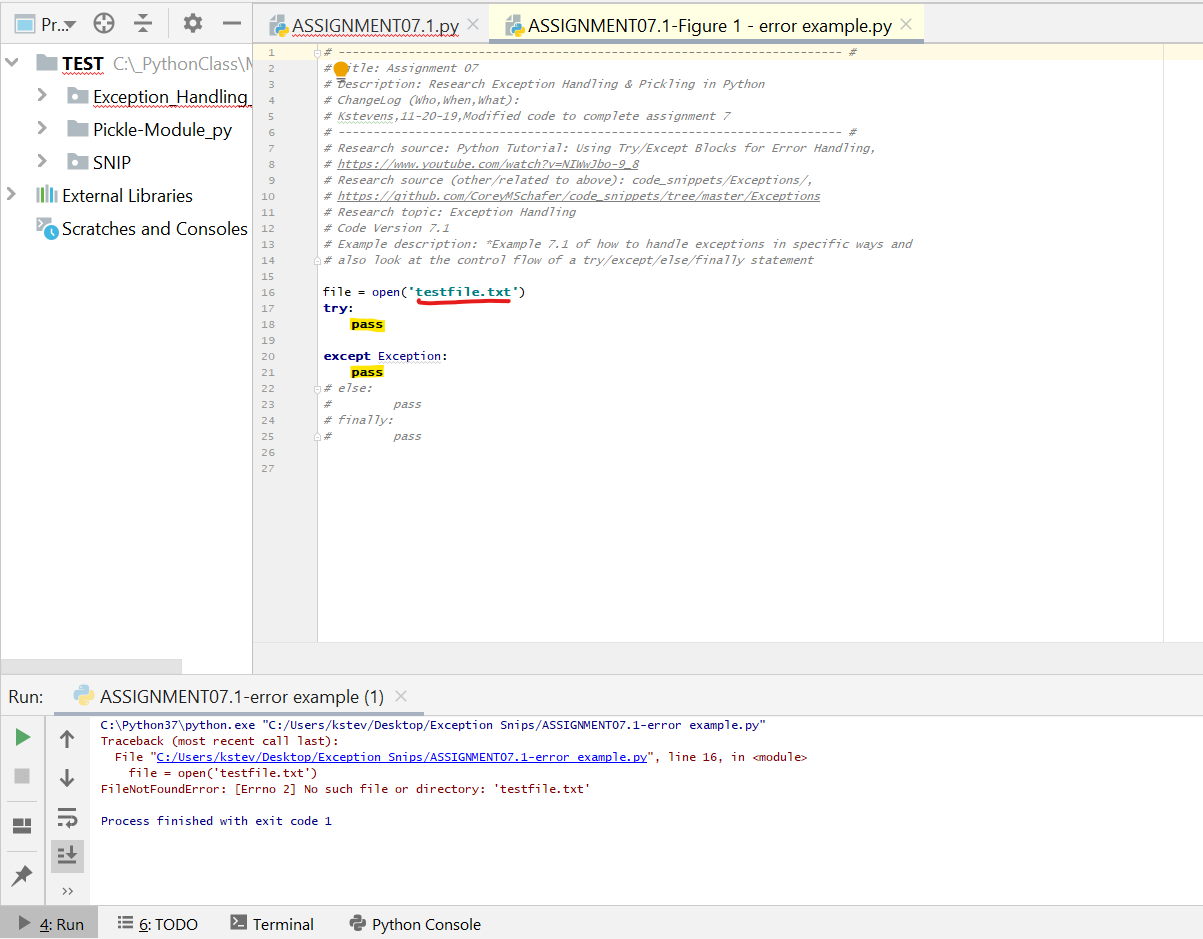
**

Figure 1.3: Displays for the flawed Input errors resulting in an Output with default traceback error message.

7.1 Input code: Basic Exception Handling

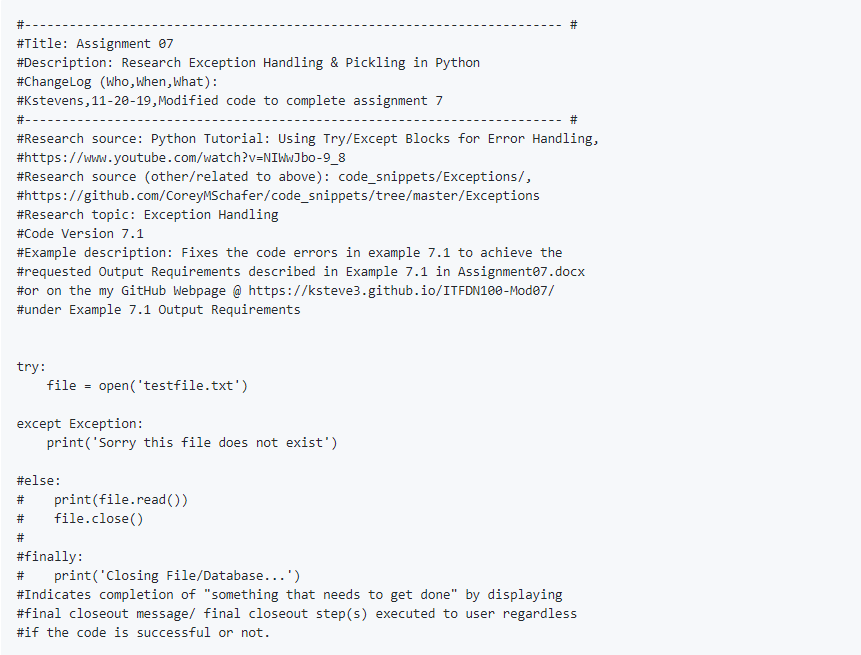


Figure 1.4: Shows how to manipulate the code from Figure 1.1 to successfully call an Exception handler to output a basic FileNotFound message back to the user whenever the defined exception occurs in the future. Output of Basic Exception Handler Example Solution (Figure1.4)

7.1 Output results: Basic Exception Handling

A screenshot of a cell phone

Description automatically generated

Figure 1.5: 7.1 Solution output od code found in Figure 1.4

##### Example:7.2

Reference:

[*Python Tutorial: Using Try/Except Blocks for Error Handling*](https://www.youtube.com/watch?v=NIWwJbo-9_8) (external link), Timestamp: 03:09

*Description:*

Anticipate areas in your code that may throw an error that will cause a user to see a default traceback error message when ran. Use the reference tutorial to learn how we can handle exceptions in specific ways to look at the control flow of a try/except/else/finally statement to replace default traceback error messages with custom or built-in error messages to help increase the speed and productivity of troubleshooting.

7.2 Output Requirements:

Create a Try/Except code handling block with multiple exception capabilities to notify the developer/user of a FileNotFound error using custom error messages.

# *7.2 Example Overview*

Figures 2.1, 2.2 & 2.3 show the successful implementation of a Try/Except exception handler using customized error handling messages for defined error occurrences.

Using the successful Try/except block code from Example 7.1, Figure 1.4 as “starter code”, manipulate the Try/except block to implement multiple exception handlers with custom error message descriptions.

7.2 Input Solution

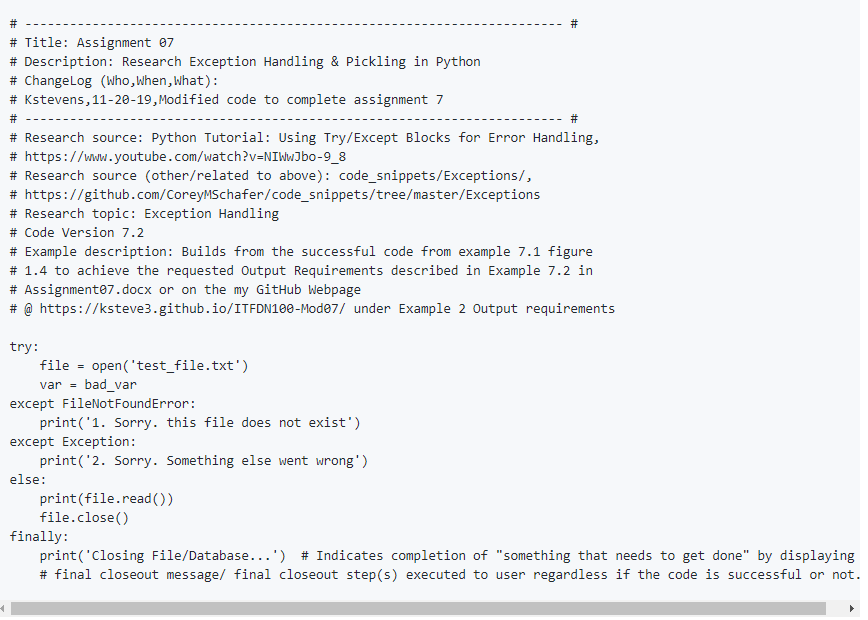


Figure 3.1: Shows new version of Try/except block after it has been manipulated using the “starter-code” from used in example 7.1, Figure 1.4. This new Try/except block uses the developer’s custom exception handler descriptions to display error messages for multiple errors that could occur.

7.2 Figure 2.1 (first exception output)

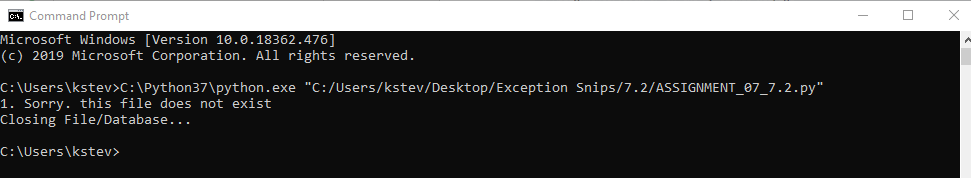


Figure 2.2: Output of first Exception handler, which displays a NameNotFound exception with a custom error message, “1. Sorry, this file does not exist” indicating the file name or the file path is wrong or does not exist in the project directory. Edit “open(‘<example>.py’)”.

7.2 Figure 2.3 (second exception output)

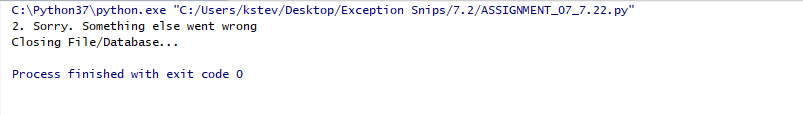


Figure 2.3: Output of second Exception handler, which displays the custom error message, “2. Sorry, something else went wrong” when a name is not defined in the code.

### Example:7.3

Reference:

[*Python Tutorial: Using Try/Except Blocks for Error Handling*](https://www.youtube.com/watch?v=NIWwJbo-9_8) (external link), Timestamp: 05:00

*Description:*

Anticipate areas in your code that may throw an error that will cause a user to see a default traceback error message when ran. Use the reference tutorial to learn how we can handle exceptions in specific ways to look at the control flow of a try/except/else/finally statement to replace default traceback error messages with custom or built-in error messages to help increase the speed and productivity of troubleshooting.

7.3 Output Requirements:

Create a Try/Except code handling block with multiple exception capabilities to notify the developer/user when an error occurs using Python’s built-in exception handlers.

Example 7.3 Overview

Figures 3.1, 3.2 & 3.3 show the successful implementation of a Try/Except block using Python’s built-in error handling messages.

Using the successful Try/except block from Example 7.2, Figure 2.1 as “starter code, manipulate the Try/except block code to implement multiple exception handlers with built-in error message descriptions.

7.3 Solution : Built in Exception Handlers

A screenshot of a social media post

Description automatically generated

Figure 3.1: Shows new version of Try/except block after it has been manipulated using the “starter-code” from used in example 7.2, Figure 2.1. This new Try/except block uses Python’s “Built-in” exception handlers to display multiple error message descriptions to display a variety of different errors that could occur.

7.3, Figure 3.1 output results

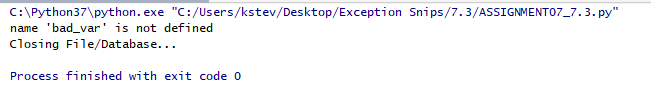


Figure 3.2: Output of first Exception handler, which displays a name error exception with a built-in error handling message, “name ‘bad\_var is not defined” indicating the file name or variable has not been defined in the code.

*7.3 Input / Output success: Built-in Exception Handling*

*A screenshot of a cell phone

Description automatically generated*

Figure 3.3: Shows both the corrected code input and its output results after the addressing the errored line of code that triggered a built-in error description to be displayed in Figure 3.2. After the errored line of code was commented out <var = bad\_var> and the wrong file name <’testfile.txt’> (responsible for error codes in displayed in Figures 1.4 & 1.5) was changed to <’test\_file.txt’>, the “try: file = open(‘<example>’)” statement finally was able to run successfully without error disruptions (Figure 3.4).

*7.3 Output results: Built-in Exception Handling*

A screenshot of a cell phone

Description automatically generated

Figure 3.4: Exception handling success!

*Pickling in Python*

(Assignment Overview)

Search the web for examples of how to use Python’s Pickling features. Make note of the URL for any pages you feel are good at explaining the subject, and why you feel that way.

*:mod:`pickle` --- Python object serialization* [26]

This documentation describes both the [pickle](https://docs.python.org/2/library/pickle.html#module-pickle) module and the [cPickle](https://docs.python.org/2/library/pickle.html#module-cPickle) module.

The [pickle](https://docs.python.org/2/library/pickle.html#module-pickle) module implements a fundamental, but powerful algorithm 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 is converted back into an object hierarchy. Pickling (and unpickling) is alternatively known as “serialization”, “marshalling,” [1](https://docs.python.org/2/library/pickle.html#id11) or “flattening”, however, to avoid confusion, the terms used here are “pickling” and “unpickling”.

 Warning :The [pickle](https://docs.python.org/2/library/pickle.html#module-pickle)module is not secure against erroneous or maliciously constructed data. Never unpickle data received from an untrusted or unauthenticated source. “Pickle — Python Object Serialization — Python 3.8.0 Documentation,” Python.org, 2019, [60]

Figure !: Security warning - [python.org](https://docs.python.org/3/library/pickle.html) [60](external link)

# Defined Functions of Pickle Module:

Apart from the Pickler and Unpickler classes, the module defines the following functions, and an exception:

**dump** (object, file[, bin])

Write a pickled representation of object to the open file object file. This is equivalent to "Pickler(file, bin).dump(object)". If the optional bin argument is present and nonzero, the binary pickle format is used; if it is zero or absent, the (less efficient) text pickle format is used.

**load** (file)

Read a pickled object from the open file object file. This is equivalent to "Unpickler(file).load()".

**dumps** (object[, bin])

Return the pickled representation of the object as a string, instead of writing it to a file. If the optional bin argument is present and nonzero, the binary pickle format is used; if it is zero or absent, the (less efficient) text pickle format is used.

**loads** (string)

Read a pickled object from a string instead of a file. Characters in the string past the pickled object's representation are ignored.

PicklingError

This exception is raised when an unpicklable object is passed to Pickler.dump().

Pickle Exceptions

Below are some of the common exceptions raised while dealing with pickle module −

Pickle.PicklingError: If the pickle object doesn’t support pickling, this exception is raised.

Pickle.UnpicklingError: In case the file contains bad or corrupted data.

EOFError: In case the end of file is detected, this exception is raised.

See Also:

Module [copyreg](https://docs.python.org/2.0/lib/module-copyreg.html) [32]

pickle interface constructor registration.

|  |
| --- |
| Module [shelve](https://docs.python.org/2.0/lib/module-shelve.html) [61] (external link) |

indexed databases of objects; uses pickle.

Module [copy](https://docs.python.org/2.0/lib/module-copy.html)[62]

shallow and deep object copying.

Module [marshal](https://docs.python.org/2.0/lib/module-marshal.html) [63](external link):

high-performance serialization of built-in types.

Advantages of using Pickle Module:

Comes handy to save complicated data.

Easy to use, lighter and doesn’t require several lines of code.

The pickled file generated is not easily readable and thus provide some security.

Recursive objects (objects containing references to themselves): Pickle keeps track of the objects it has already serialized, so later references to the same object won’t be serialized again. (The marshal module breaks for this.)

Object sharing (references to the same object in different places): This is similar to self- referencing objects; pickle stores the object once, and ensures that all other references point to the master copy. Shared objects remain shared, which can be very important for mutable objects.

User-defined classes and their instances: Marshal does not support these at all, but pickle can save and restore class instances transparently. The class definition must be importable and live in the same module as when the object was stored.

Cons:

Languages other than python may not able to reconstruct pickled python objects.

Risk of unpickling data from malicious sources ( see Figure ! )

**Warning**

The [pickle](https://docs.python.org/2/library/pickle.html) [17] (external link)  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) [65](external link) 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)[67] (external link)  may be more appropriate if you are processing untrusted data. See [Comparison with json](https://docs.python.org/3/library/pickle.html#comparison-with-json)[66]. (external link)

Figure !: Security warning - [python.org](https://docs.python.org/3/library/pickle.html) [60] (external link)

# 2.1 [copyreg](https://docs.python.org/2/library/copy_reg.html#module-copy_reg) — Register [pickle](https://docs.python.org/2/library/pickle.html#module-pickle) support functions [68]

The [copyreg](https://docs.python.org/2/library/copy_reg.html#module-copy_reg) module offers a way to define functions used while pickling specific objects.

The [pickle](https://docs.python.org/2/library/pickle.html#module-pickle), [cPickle](https://docs.python.org/2/library/pickle.html#module-cPickle), and [copy](https://docs.python.org/2/library/copy.html#module-copy) modules use those functions when pickling/copying those objects. The module provides configuration information about object constructors which are not classes. Such constructors may be factory functions or class instances.

#### copyreg.constructor(object)

* Declares object to be a valid constructor. If object is not callable (and hence not valid as a constructor), raises [TypeError](https://docs.python.org/2/library/exceptions.html#exceptions.TypeError).

copyreg.pickle(type, function[, constructor])

* Declares that function should be used as a “reduction” function for objects of type type; type must not be a “classic” class object. (Classic classes are handled differently; see the documentation for the [pickle](https://docs.python.org/2/library/pickle.html#module-pickle) module for details.) function should return either a string or a tuple containing two or three elements.
* The optional constructor parameter, if provided, is a callable object which can be used to reconstruct the object when called with the tuple of arguments returned by function at pickling time. [TypeError](https://docs.python.org/2/library/exceptions.html#exceptions.TypeError) will be raised if object is a class or constructor is not callable.
* See the [pickle](https://docs.python.org/2/library/pickle.html#module-pickle) module for more details on the interface expected of function and constructor.

# 2.1 copyreg – Register Pickle through Python.org

“Copyreg — Register Pickle Support Functions — Python 3.8.0 Documentation” (Python.org, 2019) <https://docs.python.org/3/library/copyreg.html> accessed November 25, 2019.

# 2.1 copyreg – Pickle Registration Example

A screenshot of a cell phone

Description automatically generated

Figure 2.1

# 2.2 Python Pickling & Unpickling [69]

Python pickle module is used for serializing and de-serializing python object structures. The process to converts any kind of python objects (list, dict, etc.) into byte streams (0s and 1s) is called pickling or serialization or flattening or marshalling. We can converts the byte stream (generated through pickling) back into python objects by a process called as unpickling.

Why Pickle?:

In real world scenario, the use pickling and unpickling are widespread as they allow us to easily transfer data from one server/system to another and then store it in a file or database.\

*Caution: It is advisable not to unpickle data received from an untrusted source as they may pose security threat. However, the pickle module has no way of knowing or raise alarm while pickling malicious data.*

Figure !: Security warning - [python.org](https://docs.python.org/3/library/pickle.html) [60] (external link)

Please note:

* Only after importing pickle module we can do pickling and unpickling.
* On running above script(unpickle) (below) we get our dictionary back as we initialized earlier.
* Also, please note because we are reading bytes here, we have used “rb” instead of “r”.

# *2.2 Input*

"C:\Users\kstev\Desktop\Exception Snips\Pickling\2.2\2.2 pickling unpickling.txt”

Figure 2.2a

# *2.2 Output*

{1: 'Zack', 2: '53050', 3: 'IT', 4: '38', 5: 'Flipkart'}

Figure 2.2b

# 2.3 Python Pickle Module for saving Objects by serialization [70]

Pickling is used to store python objects. This means things like lists, dictionaries, class objects, and more.

Generally, you will find pickling to be most useful with data analysis, where you are performing routine tasks on the data, such as pre-processing. Also, it makes a lot of sense when you're working with Python-specific data types, such as dictionaries.

For example, we use [pickling in the NLTK tutorial series to save our trained machine learning algorithm](https://pythonprogramming.net/pickle-classifier-save-nltk-tutorial/). This is so that, every time we want to use it, we do not need to constantly re-train it, which takes a while.

Instead, we just train the algorithm once, store it to a variable (an object), and then we pickle it. In the case of the NLTK module, generating the classifiers every time was taking 5-15+ minutes. With pickle, it was taking about 5 seconds.

If you have a large dataset, for example, and you're loading that massive data set into memory every time you run the program, it may make a lot of sense to just pickle it, and then load that instead, because it will be far faster, again by 50 - 100x, sometimes far more depending on the size.

Through saving the serialized object, it's nature is included, so we don't have to worry about loading things "as" strings, dictionaries, lists, etc.

"C:\Users\kstev\Desktop\Exception Snips\Pickling\2.3-pythonprogramming.net\2.3.1 Python Pickle Module for saving Objects by serialization.py"

Figure 2.3.1

# 2.3.2 Saving Classifiers with NLTK [71]

Training classifiers and machine learning algorithms can take a very long time, especially if you're training against a larger data set. Ours is actually pretty small. Can you imagine having to train the classifier every time you wanted to fire it up and use it? What horror! Instead, what we can do is use the Pickle module to go ahead and serialize our classifier object, so that all we need to do is load that file in real quick.

So, how do we do this? The first step is to save the object. To do this, first you need to import pickle at the top of your script, then, after you have trained with .train() the classifier, you can then call the following lines:

"C:\Users\kstev\Desktop\Exception Snips\Pickling\2.3-pythonprogramming.net\2.3.2 Saving Classifiers with NLTK.py"

Figure 2.3.2

# 2.4 Understanding Python Pickling [72]

Prerequisite: [Pickle Module](https://www.geeksforgeeks.org/pickle-python-object-serialization/) [72] (external link) , or See 2.7 examples

Python pickle module is used for serializing and de-serializing a Python object structure. Any object in Python can be pickled so that it can be saved on disk. What pickle does is that it “serializes” the object first before writing it to file. Pickling is a way to convert a python object (list, dict, etc.) into a character stream. The idea is that this character stream contains all the information necessary to reconstruct the object in another python script.

# 2.4.1 Input:

**"C:\Users\kstev\Desktop\Exception Snips\Pickling\2.4\2.4.1 Input - Understanding python picking with example .txt"**

Figure 2.4.1a

2.4.1 Output:



Figure 2.4.1b

##### 2.4.2 Pickling without a file

**"C:\Users\kstev\Desktop\Exception Snips\Pickling\2.4\2.4.2 Pickling without a file.txt"**

Figure 2.4.2

2.7 pickle—Python object serialization [72]

The pickle module is used for implementing binary protocols for serializing and de-serializing a Python object structure.

* Pickling: It is a process where a Python object hierarchy is converted into a byte stream.
* Unpickling: It is the inverse of Pickling process where a byte stream is converted into an object hierarchy.

*Module Interface :*

* dumps() – This function is called to serialize an object hierarchy.
* loads() – This function is called to de-serialize a data stream.

For more control over serialization and de-serialization, Pickler or an Unpickler objects are created respectively.

# Constants provided by the pickle module :

* pickle.HIGHEST\_PROTOCOL  
  This is an integer value representing the highest protocol version available. This is considered as the protocol value which is passed to the functions dump(), dumps().
* pickle.DEFAULT\_PROTOCOL  
  This is an integer value representing the default protocol used for pickling whose value may be less than the value of highest protocol.
* Functions provided by the pickle module :
* pickle.dump(obj, file, protocol = None, \*, fix\_imports = True)  
  This function is equivalent to Pickler(file, protocol).dump(obj). This is used to write a pickled representation of obj to the open file object file.
* The optional protocol argument is an integer that tells the pickler to use the given protocol.The supported protocols are 0 to HIGHEST\_PROTOCOL. If not specified, the default is DEFAULT\_PROTOCOL. If a negative number is specified, HIGHEST\_PROTOCOL is selected.
* If fix\_imports is true and protocol is less than 3, pickle will try to map the new Python 3 names to the old module names used in Python 2, so that the pickle data stream is readable with Python 2.

2.7.1 Input pickle.dump:

"C:\Users\kstev\Desktop\Exception Snips\Pickling\2.7\2.7.1 pickle.dump - input.txt"

Figure 2.7.1a

2.7.1 Output pickle.dump:

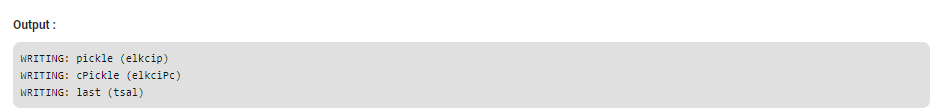
**

Figure 2.7.1b

2.7.2 Input pickle.load:

"C:\Users\kstev\Desktop\Exception Snips\Pickling\2.7\2.7.2pickle.load - input.txt"

Figure 2.7.2a

2.7.2 Output pickle.load:

*A close up of a logo

Description automatically generated*

Figure 2.7.2b

2.7.3 Input pickle.load:

"C:\Users\kstev\Desktop\Exception Snips\Pickling\2.7\2.7.3 pickle.load - input.txt"

Figure 2.7.3a

2.7.3 Output pickle.load:

*A close up of a logo

Description automatically generated*

Figure 2.7.3b

2.7.4 Input pickle.load:

"C:\Users\kstev\Desktop\Exception Snips\Pickling\2.7\2.7.4 pickle.loads.txt"

Figure 2.7.4a

2.7.4 Output pickle.load:

*A close up of a logo

Description automatically generated*

Figure 2.7.4b

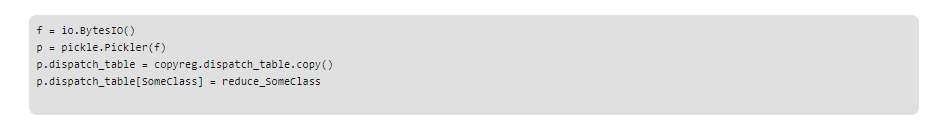
**

Figure 2.7.4c

Exceptions provided by the pickle module :

* 1.exception pickle.PickleError  
  This exception inherits Exception. It is the base class for all other exceptions raised in pickling.
* 2.exception pickle.PicklingError  
  This exception inherits PickleError. This exception is raised when an unpicklable object is encountered by Pickler.
* 3.exception pickle.UnpicklingError  
  This exception inherits PickleError. This exception is raised when there is a problem like data corruption or a security violation while unpickling an object.

# Classes exported by the pickle module:

1. class pickle.Pickler(file, protocol = None, \*, fix\_imports = True)  
This class takes a binary file for writing a pickle data stream.

* 1.dump(obj) – This function is used to write a pickled representation of obj to the open file object given in the constructor.
* *2.persistent\_id(obj) –* If persistent\_id() returns None, obj is pickled as usual. This does nothing by deafult and exists so that any subclass can override it.
* 3.Dispatch\_table – A pickler object’s dispatch table is a mapping whose keys are classes and whose values are reduction functions.  
  By default, a pickler object will not have a dispatch\_table attribute, and it will instead use the global dispatch table managed by the copyreg module.

Example : The below code creates an instance of pickle.Pickler with a private dispatch table whandles the SomeClass class specially.

A close up of a logo

Description automatically generated Figure 2.7.5a

*4.Fast –*The fast mode disables the usage of memo and speeds up the pickling process by not generating superfluous PUT opcodes.

*2.class pickle.Unpickler(file, \*, fix\_imports = True, encoding = “ASCII”, errors = “strict”)*  
 This class takes a binary file for reading a pickle data stream.

* *1.load() –*This function is used to read a pickled object representation from the open file object file and return the reconstituted object hierarchy specified.
* *2.persistent\_load(pid) –*This raises an UnpicklingError by default.
* *3.find\_class(module, name) –*This function imports module if required and returns the object called name from it, where the module and name arguments are str objects.

What can be pickled and unpickled?  
The following types can be pickled :

* None, True, and False
* integers, floating point numbers, complex numbers
* strings, bytes, bytearrays
* tuples, lists, sets, and dictionaries containing only picklable objects
* functions defined at the top level of a module (using def, not lambda)
* built-in functions defined at the top level of a module
* classes that are defined at the top level of a module
* instances of such classes whose \_\_dict\_\_ or the result of calling \_\_getstate\_\_() is picklable

Pickling Class Instances :  
This section explains the general mechanisms available to define, customize, and control how class instances are pickled and unpickled.  
No additional code is needed to make instances picklable. By default, pickle will retrieve the class and the attributes of an instance via introspection.

Classes can alter the default behaviour by providing one or several special methods :

* object.\_\_getnewargs\_ex\_\_()  
  This method dictates the values passed to the \_\_new\_\_() method upon unpickling. The method must return a pair (args, kwargs) where args is a tuple of positional arguments and kwargs a dictionary of named arguments for constructing the object.
* object.\_\_getnewargs\_\_()  
  This method supports only positive arguments. It must return a tuple of arguments args which will be passed to the \_\_new\_\_() method upon unpickling.
* object.\_\_getstate\_\_()  
  If this method is defined by classes, it is called and the returned object is pickled as the contents for the instance, instead of the contents of the instance’s dictionary.
* object.\_\_setstate\_\_(state)  
  If this method is defined by classes, it is called with the unpickled state. The pickled state must be a dictionary and its items are assigned to the new instance’s dictionary.
* object.\_\_reduce\_\_()  
  The \_\_reduce\_\_() method takes no argument and shall return either a string or preferably a tuple.
* object.\_\_reduce\_ex\_\_(protocol)  
  This method is similar to \_\_reduce\_\_ method.
* It takes a single integer argument. The main use for this method is to provide backwards-compatible reduce values for older Python releases.

Example : Handling Stateful Objects  
This example shows how to modify pickling behavior for a class. The TextReader class opens a text file, and returns the line number and line

* 1.If a TextReader instance is pickled, all attributes except the file object member are saved.
* 2.When the instance is unpickled, the file is reopened, and reading resumes from the last location.

2.7.5 Input Handling Stateful Objects:

"C:\Users\kstev\Desktop\Exception Snips\Pickling\2.7\2.7.5 Handling Stateful Objects.txt"

Figure 2.7.5b

2.7.5 Output Handling Stateful Objects:

*A close up of a logo

Description automatically generated*

Figure 2.7.5c

Bibliography

[1] C. Schafer, “Python Tutorial: Using Try/Except Blocks for Error Handling,” *YouTube*. 2019. YouTube. 2019. https://www.youtube.com/watch?v=NIWwJbo-9\_8.

[2] CoreyMSchafer, “CoreyMSchafer/code\_snippets,” *GitHub*, 17-Oct-2019. [Online]. Available: https://github.com/CoreyMSchafer/code\_snippets. [Accessed: 24-Nov-2019].

[3] Sentdex, “Python 3 Programming Tutorial - Try and Except error Handling,” *YouTube*. 12-Jul-2014.

[6] PythonProgramming.org, “Python Repository :: cpython/Lib/,” *python/cpython | GitHub*, 23-Nov-2019. [Online]. Available: https://github.com/python/cpython/tree/master/Lib. [Accessed: 24-Nov-2019].

[7] “CommonMark Spec,” *Commonmark.org*, 2019. [Online]. Available: https://spec.commonmark.org/0.29/. [Accessed: 24-Nov-2019].

[8] “commonmark.js demo,” *Commonmark.org*, 2019. [Online]. Available: https://spec.commonmark.org/dingus/?text=%09foo%09baz%09%09bim%0A. [Accessed: 24-Nov-2019].

[9] “copyreg — Register pickle support functions - GeeksforGeeks,” *GeeksforGeeks*, 07-Jun-2017. [Online]. Available: https://www.geeksforgeeks.org/copyreg-register-pickle-support-functions/. [Accessed: 24-Nov-2019].

[10] “Understanding Python Pickling with example - GeeksforGeeks,” *GeeksforGeeks*, 08-Jun-2017. [Online]. Available: https://www.geeksforgeeks.org/understanding-python-pickling-example/. [Accessed: 24-Nov-2019].

[11] PythonProgramming.org, “Python Repository :: cpython/Doc/library/pickle.rst,” *cpython/Doc/library/pickle.rst | GitHub*, 03-Nov-2019. [Online]. Available: https://github.com/python/cpython/blob/master/Doc/library/pickle.rst. [Accessed: 24-Nov-2019].

[12] “Python,” *GitHub*, 24-Nov-2019. [Online]. Available: https://github.com/python. [Accessed: 24-Nov-2019].

[13] Super User, “Try, Except, Else, Finally,” *Uci.edu*, 2019. [Online]. Available: http://tutors.ics.uci.edu/index.php/79-python-resources/104-try-except. [Accessed: 24-Nov-2019].

[14] “control flow of a try/except/else/finally statement - Google Search,” *Google.com*, 2015. [Online]. Available: https://www.google.com/search?q=control+flow+of+a+try%2Fexcept%2Felse%2Ffinally+statement&oq=control+flow+of+a+try%2Fexcept%2Felse%2Ffinally+statement&aqs=chrome..69i57j33.253j0j4&sourceid=chrome&ie=UTF-8. [Accessed: 24-Nov-2019].

[15] C. Schafer, “YouTube,” *YouTube*. 2019.

[16] “Python Pickle Module for saving Objects by serialization - Google Search,” *Google.com*, 2016. [Online]. Available: https://www.google.com/search?q=Python+Pickle+Module+for+saving+Objects+by+serialization&newwindow=1&sxsrf=ACYBGNQ09VGG3sOEeJVAaKC9tl\_Mr36xFg:1574474257447&source=lnms&tbm=vid&sa=X&ved=2ahUKEwiUuMrWnf\_lAhWDvJ4KHba0BvoQ\_AUoAnoECGgQBA&biw=1920&bih=893&google\_abuse=GOOGLE\_ABUSE\_EXEMPTION%3DID%3D92b1be5e5b19d7a0:TM%3D1574537361:C%3Dr:IP%3D2601:602:9d02:a130:2527:41bc:a3bd:1516-:S%3DAPGng0uGzsujrIUb0rhzIjIStAJ2lX4gkg%3B+path%3D/%3B+domain%3Dgoogle.com%3B+expires%3DSat,+23-Nov-2019+22:29:21+GMT. [Accessed: 24-Nov-2019].

[17] “11.1. pickle — Python object serialization — Python 2.7.17 documentation,” *Python.org*, 2019. [Online]. Available: https://docs.python.org/2/library/pickle.html#relationship-to-other-python-modules. [Accessed: 24-Nov-2019].

[18] N. Bowers, “Test::Cmd - Perl module for portable testing of commands and scripts - metacpan.org,” *Metacpan.org*, 25-Oct-2015. [Online]. Available: https://metacpan.org/pod/Test::Cmd. [Accessed: 24-Nov-2019].

[19] “Python Global, Local and Nonlocal variables (With Examples),” *Programiz.com*, 2019. [Online]. Available: https://www.programiz.com/python-programming/global-local-nonlocal-variables. [Accessed: 24-Nov-2019].

[20] datacamp, “datacamp/pythonwhat,” *GitHub*, 16-Sep-2019. [Online]. Available: https://github.com/datacamp/pythonwhat. [Accessed: 24-Nov-2019].

[21] “pythonwhat — pythonwhat 2.21.0 documentation,” *Readthedocs.io*, 2019. [Online]. Available: https://pythonwhat.readthedocs.io/en/latest/. [Accessed: 24-Nov-2019].

[22] “SCTs python - Google Search,” *Google.com*, 2019. [Online]. Available: https://www.google.com/search?newwindow=1&sxsrf=ACYBGNTaog76UITs3dol-M8EyHeyXbGz6w%3A1574596186347&ei=Wm7aXcXkFMTn-wSn5oXoBQ&q=SCTs+python&oq=SCTs+python&gs\_l=psy-ab.3...3244.5795..7643...0.2..0.79.471.7......0....1..gws-wiz.......0i71j0i67j0i20i263j0i10j0j0i22i30j0i22i10i30j33i160j0i13j0i13i30.6-ksYmYFEGA&ved=0ahUKEwjF-efy44LmAhXE854KHSdzAV0Q4dUDCAs&uact=5. [Accessed: 24-Nov-2019].

[23] google, “Python: API for verifying SCTs. · google/certificate-transparency@e9a274f,” *GitHub*, 13-Feb-2015. [Online]. Available: https://github.com/google/certificate-transparency/commit/e9a274f358702f634da2a67cc07113e33bdfff87. [Accessed: 24-Nov-2019].

[24] “ControLling mUltiple streams for tElepresence Wiki,” *Ietf.org*, 2015. [Online]. Available: https://trac.ietf.org/trac/clue. [Accessed: 24-Nov-2019].

[25] “pickle — Python object serialization - GeeksforGeeks,” *GeeksforGeeks*, 08-Jun-2017. [Online]. Available: https://www.geeksforgeeks.org/pickle-python-object-serialization/. [Accessed: 24-Nov-2019].

[26] “Python Programming Tutorials,” *Pythonprogramming.net*, 2019. [Online]. Available: https://pythonprogramming.net/pickling-scaling-machine-learning-tutorial/. [Accessed: 24-Nov-2019].

[27] “Python Programming Tutorials,” *Pythonprogramming.net*, 2019. [Online]. Available: https://pythonprogramming.net/machine-learning-tutorial-python-introduction/. [Accessed: 24-Nov-2019].

[28] “Python Programming Tutorials,” *Pythonprogramming.net*, 2019. [Online]. Available: https://pythonprogramming.net/pickle-classifier-save-nltk-tutorial/. [Accessed: 24-Nov-2019].

[29] python.org, :“mod:`pickle` --- Python object serialization,” *https://docs.python.org/*, 2019. [Online]. Available: https://docs.python.org/2/\_sources/library/pickle.rst.txt. [Accessed: 24-Nov-2019].

[30] python, “python/cpython,” *GitHub*, 23-Nov-2019. [Online]. Available: https://github.com/python/cpython/tree/master/Doc/library. [Accessed: 24-Nov-2019].

[31] “Python Module Index — Python 3.8.0 documentation,” *Python.org*, 2019. [Online]. Available: https://docs.python.org/3/py-modindex.html. [Accessed: 24-Nov-2019].

[32] “Index — Python 3.8.0 documentation,” *Python.org*, 2019. [Online]. Available: https://docs.python.org/3/genindex-all.html. [Accessed: 24-Nov-2019].

[33] “Index of Menu Items - Help | PyCharm,” *Jetbrains.com*, 2019. [Online]. Available: https://www.jetbrains.com/help/pycharm/index-of-menu-items.html. [Accessed: 24-Nov-2019].

[34] “Query results - Help | PyCharm,” *Jetbrains.com*, 2019. [Online]. Available: https://www.jetbrains.com/help/pycharm/viewing-query-results.html#export-data. [Accessed: 24-Nov-2019].

[35] “Full-stack Web Development - Features | PyCharm,” *JetBrains*, 2019. [Online]. Available: https://www.jetbrains.com/pycharm/features/web\_development.html. [Accessed: 24-Nov-2019].

[36] “Regular Expression Syntax Reference - Help | PyCharm,” *Jetbrains.com*, 2019. [Online]. Available: https://www.jetbrains.com/help/pycharm/regular-expression-syntax-reference.html. [Accessed: 24-Nov-2019].

[37] “Managing data sources - Help | PyCharm,” *Jetbrains.com*, 2019. [Online]. Available: https://www.jetbrains.com/help/pycharm/managing-data-sources.html. [Accessed: 24-Nov-2019].

[38] “Scope Language Syntax Reference - Help | PyCharm,” *Jetbrains.com*, 2019. [Online]. Available: https://www.jetbrains.com/help/pycharm/scope-language-syntax-reference.html. [Accessed: 24-Nov-2019].

[39] “Install EduTools Plugin - Help | Educational Products,” *Jetbrains.com*, 2019. [Online]. Available: https://www.jetbrains.com/help/education/install-edutools-plugin.html?section=IntelliJ%20IDEA. [Accessed: 24-Nov-2019].

[40] “Regular Expression Syntax Reference - Help | PyCharm,” *Jetbrains.com*, 2019. [Online]. Available: https://www.jetbrains.com/help/pycharm/regular-expression-syntax-reference.html. [Accessed: 24-Nov-2019].

[41] “Import/Export options from CSV to database - Features | DataGrip,” *JetBrains*, 2019. [Online]. Available: https://www.jetbrains.com/datagrip/features/importexport.html. [Accessed: 24-Nov-2019].

[42] “Built-in Developer Tools - Features | PyCharm,” *JetBrains*, 2019. [Online]. Available: https://www.jetbrains.com/pycharm/features/tools.html. [Accessed: 24-Nov-2019].

[43] “Download DataGrip: Cross-Platform IDE for Databases & SQL,” *JetBrains*, 2019. [Online]. Available: https://www.jetbrains.com/datagrip/download/#section=windows. [Accessed: 24-Nov-2019].

[44] “Free downloads of Ablebits.com Excel add-ins and Outlook plug-ins,” *Ablebits.com*, 2019. [Online]. Available: https://www.ablebits.com/downloads/index.php. [Accessed: 24-Nov-2019].

[45] “Python Programming Tutorials,” *Pythonprogramming.net*, 2019. [Online]. Available: https://pythonprogramming.net/python-pickle-module-save-objects-serialization/. [Accessed: 24-Nov-2019].

[46] “copyreg — Register pickle support functions - GeeksforGeeks,” *GeeksforGeeks*, 07-Jun-2017. [Online]. Available: https://www.geeksforgeeks.org/copyreg-register-pickle-support-functions/. [Accessed: 24-Nov-2019].

[47] “25.4. 2to3 - Automated Python 2 to 3 code translation — Python 2.7.17 documentation,” *Python.org*, 2019. [Online]. Available: https://docs.python.org/2/library/2to3.html#module-lib2to3. [Accessed: 24-Nov-2019].

[48] “Python Pickling,” *Tutorialspoint.com*, 2019. [Online]. Available: https://www.tutorialspoint.com/python-pickling. [Accessed: 24-Nov-2019].

[49] “11.1. pickle — Python object serialization — Python 2.7.17 documentation,” *Python.org*, 2019. [Online]. Available: https://docs.python.org/2/library/pickle.html#module-pickle. [Accessed: 25-Nov-2019].

[50] “copyreg — Register pickle support functions — Python 3.8.0 documentation,” *Python.org*, 2019. [Online]. Available: https://docs.python.org/3/library/copyreg.html. [Accessed: 25-Nov-2019].

[51] “copyreg — Register pickle support functions — Python 3.8.0 documentation,” *Python.org*, 2019. [Online]. Available: https://docs.python.org/3/library/copyreg.html. [Accessed: 25-Nov-2019].

[52] “pickle — Python object serialization — Python 3.8.0 documentation,” *Python.org*, 2019. [Online]. Available: https://docs.python.org/3/library/pickle.html. [Accessed: 25-Nov-2019].

[53] “3.11.1 Example,” *Python.org*, 2019. [Online]. Available: https://docs.python.org/2.0/lib/pickle-example.html. [Accessed: 25-Nov-2019].

[54] “Module Index,” *Python.org*, 2019. [Online]. Available: https://docs.python.org/2.0/lib/modindex.html. [Accessed: 25-Nov-2019].

[55] “Python Library Reference,” *Python.org*, 2019. [Online]. Available: https://docs.python.org/2.0/lib/lib.html. [Accessed: 25-Nov-2019].

[56] “Extending and Embedding the Python Interpreter,” *Python.org*, 2019. [Online]. Available: https://docs.python.org/2.0/ext/ext.html. [Accessed: 25-Nov-2019].

[57] “11. Data Persistence — Python 2.7.17 documentation,” *Python.org*, 2019. [Online]. Available: https://docs.python.org/2/library/persistence.html. [Accessed: 25-Nov-2019].

[58] “pickle — Python object serialization — Python 3.8.0 documentation,” *Python.org*, 2019. [Online]. Available: https://docs.python.org/3/library/pickle.html. [Accessed: 25-Nov-2019].

[59] python.org, “pickle — Python object serialization — Python 3.7.3 documentation,” *Python.org*, 2019. [Online]. Available: https://docs.python.org/3/library/pickle.html.

[60] “Stateful - APIDesign,” *Apidesign.org*, 2010. [Online]. Available: http://wiki.apidesign.org/wiki/Stateful. [Accessed: 25-Nov-2019].

[61] “3.14 shelve -- Python object persistence,” *Python.org*, 2019. [Online]. Available: https://docs.python.org/2.0/lib/module-shelve.html. [Accessed: 25-Nov-2019].

[62] “3.15 copy -- Shallow and deep copy operations,” *Python.org*, 2019. [Online]. Available: https://docs.python.org/2.0/lib/module-copy.html. [Accessed: 25-Nov-2019].

[63] “3.16 marshal -- Alternate Python object serialization,” *Python.org*, 2019. [Online]. Available: https://docs.python.org/2.0/lib/module-marshal.html. [Accessed: 25-Nov-2019].

[64] “hmac — Keyed-Hashing for Message Authentication — Python 3.8.0 documentation,” *Python.org*, 2019. [Online]. Available: https://docs.python.org/3/library/hmac.html#module-hmac. [Accessed: 25-Nov-2019].

[65] “pickle — Python object serialization — Python 3.8.0 documentation,” *Python.org*, 2019. [Online]. Available: https://docs.python.org/3/library/pickle.html#comparison-with-json. [Accessed: 25-Nov-2019].

[66] “json — JSON encoder and decoder — Python 3.8.0 documentation,” *Python.org*, 2019. [Online]. Available: https://docs.python.org/3/library/json.html#module-json. [Accessed: 25-Nov-2019].

[67] “Research Guides: Microsoft Word for Dissertations: Cross-References,” *Umich.edu*, 2019. [Online]. Available: https://guides.lib.umich.edu/c.php?g=283073&p=1888264. [Accessed: 25-Nov-2019].

[68] “LibGuides: Citation Styles: APA, MLA, Chicago, Turabian, IEEE: IEEE Style,” *Libguides.com*, 2019. [Online]. Available: https://pitt.libguides.com/citationhelp/ieee. [Accessed: 25-Nov-2019].

[69] I. Periodicals, “IEEE EDITORIAL STYLE MANUAL FOR AUTHORS.”

[70] University of Washington, “Canvas Login,” *https://canvas.uw.edu/*, 2019. [Online]. Available: https://canvas.uw.edu/courses/1342958/modules/items/9973247. [Accessed: 25-Nov-2019].

[71] PythonMod7Project, “PythonMod7Project,” *itfdn100 YouTube*, 13-Nov-2019. [Online]. Available: https://youtu.be/4IkIdXJBC6o. [Accessed: 25-Nov-2019].

[72] “Assignment07.pdf: IT FDN 100 A Au 19: Foundations Of Programming: Python,” *Uw.edu*, 2019. [Online]. Available: https://canvas.uw.edu/courses/1342958/files/59791641?module\_item\_id=9973247. [Accessed: 25-Nov-2019].

[73] “Lab7-1\_Starter.py: IT FDN 100 A Au 19: Foundations Of Programming: Python,” *Uw.edu*, 2019. [Online]. Available: https://canvas.uw.edu/courses/1342958/files/59729143?module\_item\_id=9973388. [Accessed: 25-Nov-2019].

[74] “\_Mod7PythonProgrammingNotes.pdf: IT FDN 100 A Au 19: Foundations Of Programming: Python,” *Uw.edu*, 2019. [Online]. Available: https://canvas.uw.edu/courses/1342958/files/59801217?module\_item\_id=9973246. [Accessed: 25-Nov-2019].

[75] R. Root, “YouTube,” *YouTube*. 2019.

[76] “Assignment07.pdf: IT FDN 100 A Au 19: Foundations Of Programming: Python,” *Uw.edu*, 2019. [Online]. Available: https://canvas.uw.edu/courses/1342958/files/59791641?module\_item\_id=9973247. [Accessed: 25-Nov-2019].

[77] “\_Mod7PythonProgrammingNotes.pdf: IT FDN 100 A Au 19: Foundations Of Programming: Python,” *Uw.edu*, 2019. [Online]. Available: https://canvas.uw.edu/courses/1342958/files/59801217?module\_item\_id=9973246. [Accessed: 25-Nov-2019].

Compiled Recourse Lists:

[4] Kstevens, “Compiled List of Resources and Tutorials Related to Exception Handling in Python 3,” *Google.com*, 20-Nov-2019. [Online]. Available: https://docs.google.com/spreadsheets/d/e/2PACX-1vRnad3aZB7\_j9aKOajRgzOf3bGkSlcJ\_NSVobVJnApOc\_f7yzTMFPAHcjIhD6IxhiaIhZpEK6UEiXn1WBTG3sg/pubhtml.

[5] Kstevens, “Compiled List of Resources and Tutorials Related to Pickle Module Resources- Python Programming,” *Google.com*, 20-Nov-2019. [Online]. Available: https://docs.google.com/spreadsheets/d/e/2PACX-1vTFel2-8hzvknNPtJF\_e\_WGJuCEDRUhxEj-0LKL5En0fUX8QQTvouHaENlUEVZDDAnRQ427D\_W6cxDJUEIVZFU/pubhtml.