**Theresa Ward**

December 1, 2020

IT FDN 110 A

Assignment 07

# <https://github.com/ScrappySnacks/IntroToProg-Python-Mod07>

# <https://scrappysnacks.github.io/IntroToProg-Python-Mod07/>

## **A “pickled” Latin dictionary with exception handling**

## Introduction

For assignment #7, we are asked to create a demo that demonstrates how pickling and structured error handling work. To accomplish this, I have created a “pickled” Latin dictionary demo that also includes exception handling code. This document will describe the steps taken to create the demo and ensure it works as expected. As with past assignments PyCharm is used here for script development. Additionally, assignment documents can be found in the Mod07 repository and the text, images and links from this document can be found at <https://scrappysnacks.github.io/IntroToProg-Python-Mod07/>.

## Resources

Before script development commenced, internet research was performed on pickling and exception handling resources. I found that the same sources had very helpful instructional material on both topics. These sources are listed below along with reasoning for their recommendation:

### Helpful Pickling Resources

* <https://www.datacamp.com/community/tutorials/pickle-python-tutorial>
  + In addition to containing well-written instructional material, Datacamp provides an opportunity for additional practice. This site was the most helpful to the pickling portion of demo development.
* <https://realpython.com/python-pickle-module/>
  + This external resource puts pickling in a larger context and introduces concepts that were not introduced thus far in the course (e.g., serialization and deserialization).
* <https://docs.python.org/2/library/pickle.html>
  + This website succinctly covers important concepts on pickling.

### Helpful Exception Handling Resources

* <https://www.datacamp.com/community/tutorials/exception-handling-python>
  + This well-written site introduces basic concepts and additional opportunities for practice.
* <https://realpython.com/lessons/introduction-exceptions/>
  + This site has a 101 class on exceptions that was very well done and easy to understand.
* <https://docs.python.org/3/tutorial/errors.html>
  + This site was the most helpful to the exception handling portion of demo development.

## Demo Development

The “pickled” Latin dictionary program takes three pre-defined dictionaries with English to Latin translation (Latin colors, Latin Animals, Latin Objects), pickles them and writes them to file. The user can also recall the dictionaries from file (i.e., “unpickling”). In addition to pickling, the program also handles three exceptions through existing exception classes and one custom created class. The remainder of this assignment document describes the functionality of the code organized by Separation of Concerns. The detailed python script can be found in Appendix A.

### Demo Variables and a Custom Exception Class

After importing the pickle module, I first declare a list object, lstTable, which will store the dictionary contents as rows. I also define a custom exception class. This class will be used to raise an error if the user chooses an invalid selection. It also returns the instruction, “Please enter a valid option: 1, 2, or 3.”

### Data Dictionaries

The next portion of code defines the three data dictionaries to be pickled. They are Latin colors, Latin animals and Latin objects. Each key is the English word and each value is the Latin translation.

### Processing

In the processing section, I define two functions: PickleToFile() and UnpickleFromFile(). PickleToFile() takes four arguments which are the three dictionaries and binary filename. This function opens the binary file and pickles the data dictionaries to file. If the file does not exist, the open method will create it. UnpickleFromFile() does the unpickling . However, it does not assume the user is unpickling from the same file location. The user can pickle the data in several file locations, so the unpickling function requires the user to be specific about the file to unpickle. The function does assume, however, that the *same* Latin dictionaries are pickled each time. UnpickleFromFile() also returns the dictionary contents.

### Presentation

PrintDataFromFile() takes the unpickled data dictionaries as arguments and unpacks them for presentation. A local variable is defined that is used to loop through the list table object.

### Main Script Body

In the main body of the script, a menu of options is presented within a *while* loop that is set to Boolean True. A try / except block is also started. Within the try / except block, the user is requested to choose an option to pickle, unpickle or exit. The input is assigned to intChoice as an integer data type. An *if* statement is used to run the script based on the value of intChoice. If the user enters a value that is not integer data type, the script will raise an error via the ValueError class and print “That was not a number! Try again.” If the user enters an integer that is not 1, 2, or 3, the script will raise the NotValidChoice() exception, as discussed previously (Figure 1). The script also catches all non-specific exceptions using the Exception class at the very end of the try / except block.

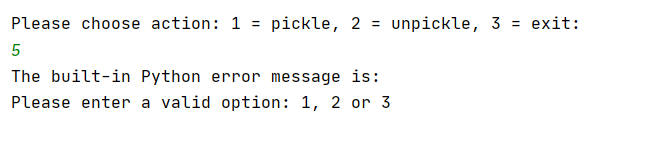


Figure 1. Output in PyCharm when an invalid selection is entered.

* If the user enters “1,” the script prompts the user to enter a file name. This is followed by printing the dictionary contents using PrintDataFromFile() and pickling using the PickleToFile() function. An example of PyCharm output when “1” is chosen is shown in Figure 2.
* If the user enters “2,” the script prompts the user to enter a file name to unpickle. Then the UnpickleFromFile() function does the unpickling and prints the results using PrintDataFromFile(). If the user enters a pickled file name that does not exist, the script will raise an error via the FileNotFoundError class. An example of PyCharm output when “2” is chosen is shown in Figure 3. Figure 3 also shows the result when the error is raised.
* If the user enters “3,” the program ends.

Lastly, the script can be run successfully from the Windows command shell as shown in Figure 4.

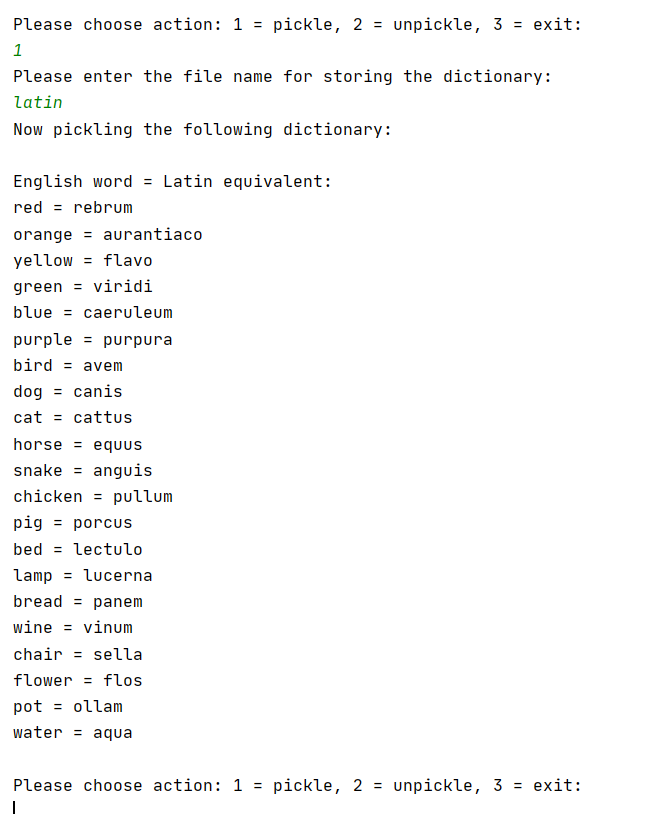


Figure 2. Output in PyCharm: Menu option 1

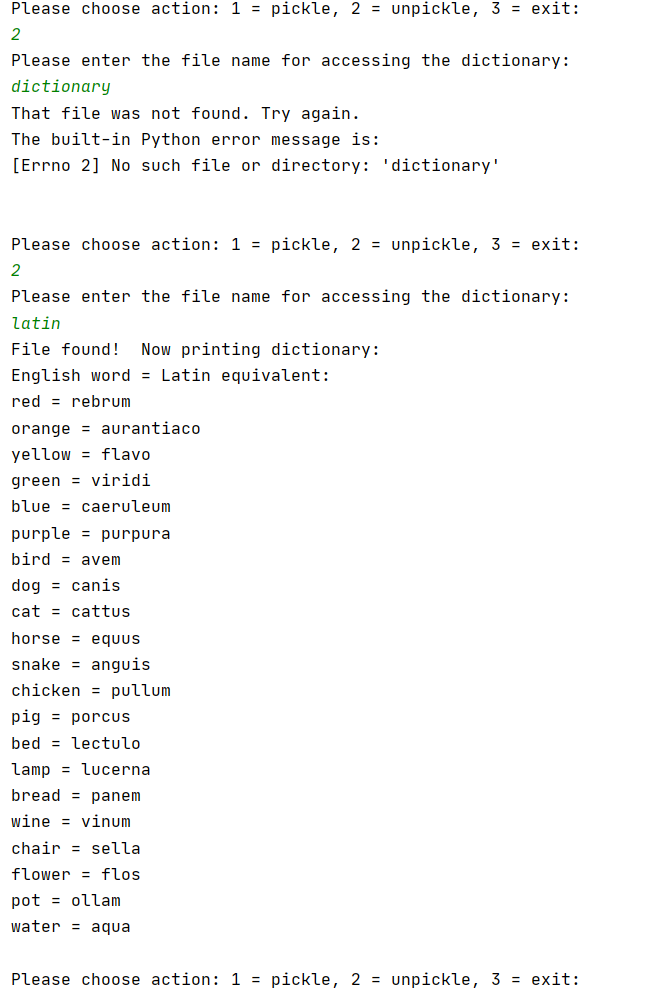


Figure 3. Output in PyCharm: Menu option 2 and example of exception handling.

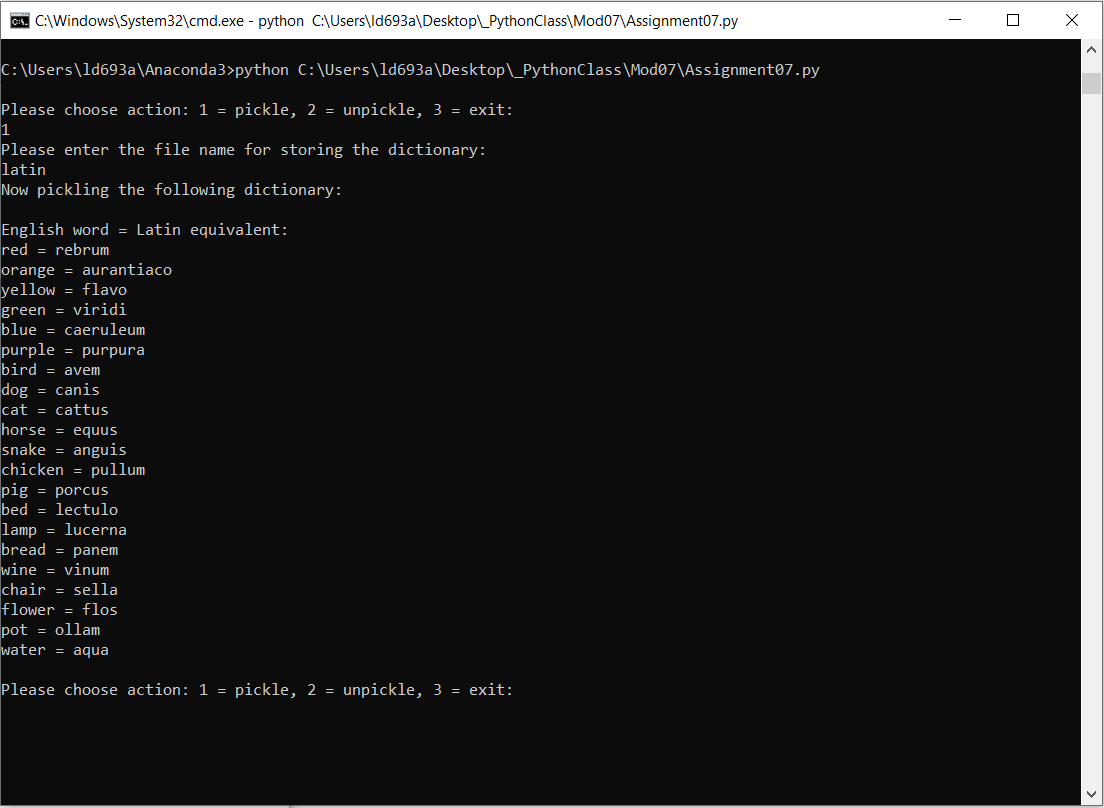


Figure 4. Output via command window.

## Summary

In summary, I was successful in completing a script that meets the requirements of Module 07. The script demo can pickle and unpickle the contents of a list table object. The script also demonstrates several examples of exception handling. However, an improvement on this script would be to allow for flexibility in modifying dictionary contents.

## Appendix A – Python code

*#-------------------------------------------------#  
# Title: A "pickled" Latin dictionary  
# Description: Demonstrates pickling and exception handling in one script  
# ChangeLog: N/A, Rev New  
# T. Ward Dev, 12/01/2020 , created script  
#-------------------------------------------------#  
# Pickle resources: See demo write-up for more information  
# https://www.datacamp.com/community/tutorials/pickle-python-tutorial  
# https://realpython.com/python-pickle-module/  
# https://docs.python.org/2/library/pickle.html  
  
# Exception handling resources: See demo write-up for more information  
# https://www.datacamp.com/community/tutorials/exception-handling-python  
# https://docs.python.org/3/tutorial/errors.html  
# https://realpython.com/lessons/introduction-exceptions/  
  
#--- Import pickle module --- #***import** pickle  
  
*#--- Declare variables and custom classes --- #*lstTable = []  
  
**class** NotValidChoice(Exception): *# Custom exception class for reminding the user that there are only 3 choices.* **def** \_\_str\_\_(self):  
 **return 'Please enter a valid option: 1, 2 or 3'** + **"\n"***# -- Define data dictionaries -- #*LatColors = {**"red"**:**"rebrum"**, **"orange"**:**"aurantiaco"**, **"yellow"**:**"flavo"**,**"green"**: **"viridi"**, **"blue"**:**"caeruleum"**, **"purple"**: **"purpura"**}  
  
LatAnimals = {**"bird"**:**"avem"**, **"dog"**:**"canis"**, **"cat"**:**"cattus"**, **"horse"**:**"equus"**, **"snake"**:**"anguis"**, **"chicken"**:**"pullum"**, **"pig"**:**"porcus"**}  
  
LatObjects = {**"bed"**: **"lectulo"**, **"lamp"**:**"lucerna"**, **"bread"**:**"panem"**, **"wine"**:**"vinum"**, **"chair"**:**"sella"**, **"flower"**: **"flos"**, **"pot"**:**"ollam"**, **"water"**:**"aqua"**}  
  
*# -- Processing -- #***def** PickleToFile(dic1,dic2,dic3, filename): *# This function opens the file pickles the data dictionaries to file.  
 # If the file does not exist, this function will create it.* outfile = open(filename,**'wb'**)  
 pickle.dump(dic1,outfile)  
 pickle.dump(dic2,outfile)  
 pickle.dump(dic3,outfile)  
 outfile.close()  
  
**def** UnpickleFromFile(r1, r2, r3, filename): *# This function unpickles the data dictionaries to file.* infile = open(filename, **"rb"**)  
 r1 = pickle.load(infile)  
 r2 = pickle.load(infile)  
 r3 = pickle.load(infile)  
 infile.close()  
 **return** r1, r2, r3  
  
*# -- Presentation (I/O) -- #***def** PrintDataFromFile(r1,r2, r3): *# This function unpacks displays data in a more appealing way.* lstTable = [r1, r2, r3]  
 print(**"English word = Latin equivalent:"**)  
 i = 0 *# Declare local variable* **while** i < int(len(lstTable)):  
 **for** english, latin **in** lstTable[i].items():  
 print(english, **"="**, latin)  
 i += 1  
  
*# -- Main Script Body -- #***while True**: *# This while loop contains if / then to run functions base on selection and includes error handling and  
 # raising a custom exception class, if the user tries to enter a choice other than 1, 2 or 3.* **try**:  
 intChoice = int(input(**"\n"**+ **"Please choose action: 1 = pickle, 2 = unpickle, 3 = exit: "**+ **"\n"**))  
 **if** intChoice == 1:  
 filename = str(input(**"Please enter the file name for storing the dictionary: "** + **"\n"**)).lower()  
 print(**"Now pickling the following dictionary: "** + **"\n"**)  
 PrintDataFromFile(LatColors, LatAnimals, LatObjects)  
 PickleToFile(LatColors, LatAnimals, LatObjects, filename.strip())  
 **elif** intChoice == 2:  
 filename = str(input(**"Please enter the file name for accessing the dictionary: "** + **"\n"**)).lower()  
 UnpickleFromFile(LatColors, LatAnimals, LatObjects, filename.strip())  
 print(**"File found! Now printing dictionary: "**)  
 PrintDataFromFile(LatColors, LatAnimals, LatObjects)  
 **elif** intChoice == 3:  
 **break** *# and exit the program* **else**:  
 **raise** NotValidChoice()  
 **except** ValueError **as** e:  
 print(**"That was not a number! Try again."**)  
 print(**"The built-in Python error message is: "**)  
 print(e, **"\n"**)  
 **except** FileNotFoundError **as** e:  
 print(**"That file was not found. Try again."**)  
 print(**"The built-in Python error message is: "**)  
 print(e, **"\n"**)  
 **except** Exception **as** e:  
 print(**"The built-in Python error message is: "**)  
 print(e, **"\n"**)