Ryan L. Blake

24 August 2021

IT FDN 100

Assignment 07

https://github.com/rblake50/IntroToProg-Python/tree/main/HW07

**Pickling and Error Handling**

# Introduction

Limitations exist to the previously learned method of file input and output using raw text. For example, what if a developer wants to save data that is more complex than strings, such as lists or dictionaries? The technique of pickling offers a way to preserve data as objects instead of raw text. This technique is explored in Assignment 07 alongside more advanced error handling in Python. A developer code with simple analysis of baseball statistics will highlight some of the key functions and steps for both pickling and error handling.

# Topics

* Pickling
* Error handling

# Summary

At this point in the course, we have learned how to perform basic operations with Python and deal with data collections like tuples, lists, and dictionaries within the Python development environment and simple text files. While this approach offers near-infinite possibilities for user design and data saving, there is a more effective way to save objects with information compared to simple text files. The technique is known as “pickling.”

The textbook provides an excellent definition, copied here for reference: “*Pickling* means to preserve—and that’s just what it means in Python. You can pickle a complex piece of data, like a list or dictionary, and save it in its entirety to a file. Best of all, your hands won’t smell like vinegar when you’re done.” The technique can avoid some common hassles related to simple text interpretation such as if delimiting or formatting conventions change throughout a document.

The following online resources were found to be helpful:

* “Pickle in Python: Object Serialization” ([link](https://www.datacamp.com/community/tutorials/pickle-python-tutorial))
* “What is Pickling in Python?” ([link](https://www.afternerd.com/blog/python-pickle/))

All pickling requires the pickle library to be imported.

import pickle

Data can be read using load().

# Deserialize data

with open(strFileToRead,'rb') as f:

e = pickle.load(f)

Data can be written using dump(). Like text input and output, pickling can read, write, or append as commanded.

# Serialize data (overwite)

with open(strFileToSave,'wb') as file:

pickle.dump(someData,file)

Error handling is another valuable function within Python. Basic error handling can be wrapped into a straightforward block of (pseudo)code:

try:

some\_action

except:

print(“Custom error message”)

else:

next\_action # if there is no error

finally:

another\_action # in success or error case

While Python has default error messaging, it is designed for developers and not users. Adding custom error handling can improve user experience and clarify errors, especially regarding user input.

The except block offers a few added tricks for error handling. One involves the interpretation of different error codes.

try:

some\_action

except ValueError:

print(“You have a value error!”)

except IOError as e:

print(“You have an IO error!”)

print(e) # Default Python error message for IOError

else:

next\_action

As shown, specific errors can be addressed, and the default error handling can even be captured for controlled usage.

# Code Description

A code can be developed on Python (Assignment07.py) to explore basic pickling and error handling methods. The code looks at baseball statistics for players to calculate the number of home runs per a standard 162 game season.

The code begins by prompting a user for a file name of a pickled object. The code first checks to see if the file exists and prints the current data if it does (and is properly formatted).

The menu presents (3) options:

1. Add a player
2. Calculate home runs per season
3. Exit (with save option)

Adding a player results in the following dictionary data structure:

|  |  |  |
| --- | --- | --- |
| **Name** | **hrs** | **gms** |

A few examples of real data (found at [Baseball Reference](https://www.baseball-reference.com/)) are copied below:

|  |  |  |
| --- | --- | --- |
| **Name** | **hrs** | **gms** |
| Carlos Pena | 286 | 1493 |
| Aramis Ramirez | 386 | 2194 |
| Jason Kendall | 75 | 2085 |

Each dictionary row is appended to a list.

When the calculation option is selected, the code asks for the player to calculate and then retrieves the [hrs] and [gms] values to calculate the average number of home runs per 162 game season.

When finished, a user can exit and choose to save the pickled object back to the same file name. Along the way, there are numerous functions and methods was basic error handling using the try-except-else logic.

The code can be tested in PyCharm. This example will look at unpickling an existing object that has (3) players’ sample data and some of the basic error handling for common mistakes.

C:/\_PythonClass/Assignment07/Assignment07.py

What file would you like to open? (Hint: try data.pickle) data.pickle

Here are the players in your list:

================================

Albert Pujols

Ken Griffey Jr.

Mike Trout

=========================================

1. Add player

2. Calculate average home runs per season

3. Exit program (with save option)

What is your selection? 1

Who do you want to add? Carlos Pena

How many home runs? ninety

D'oh! You need to put a number for both. Let's start over.

Here are the players in your list:

================================

Albert Pujols

Ken Griffey Jr.

Mike Trout

=========================================

1. Add player

2. Calculate average home runs per season

3. Exit program (with save option)

What is your selection? 1

Who do you want to add? Carlos pena

How many home runs? 286

In how many games? 1493

Here are the players in your list:

================================

Albert Pujols

Ken Griffey Jr.

Mike Trout

Carlos Pena

=========================================

1. Add player

2. Calculate average home runs per season

3. Exit program (with save option)

What is your selection? 2

Who do you want to calculate? Barry Bonds

Name not found on list. Try again!

Press ENTER

Here are the players in your list:

================================

Albert Pujols

Ken Griffey Jr.

Mike Trout

Carlos Pena

=========================================

1. Add player

2. Calculate average home runs per season

3. Exit program (with save option)

What is your selection? 2

Who do you want to calculate? Albert Pujols

Albert Pujols hit 37.20 home runs per season.

Press ENTER to return to menu.

Here are the players in your list:

================================

Albert Pujols

Ken Griffey Jr.

Mike Trout

Carlos Pena

=========================================

1. Add player

2. Calculate average home runs per season

3. Exit program (with save option)

What is your selection? 3

Do you want to save? [y]es or [n]o

yes

Data successfully pickled to data.pickle. Press ENTER to exit.

Process finished with exit code 0

The script can also be tested in the Python command shell. This example will look at opening and writing to a file that does not exist.

Text

Description automatically generated

Text

Description automatically generated

Figure : Example code with new pickled file.

Notice in this example that the data was saved to a file called something.dat. The script can be used again with something.dat to confirm the pickled list with Barry Bonds as an item is there.

Text

Description automatically generated

Figure : Deserialization of data from previous code in something.dat.

The file something.dat is also seen in the directory:

Table

Description automatically generated

Figure . Data file as seen in the Windows Explorer directory.

When opened with a text editor, the result is apparent gibberish:

€•; ]”}”(ŒName”Œ  
Barry Bonds”Œhrs”G@‡Ð Œgms”G@§T ua.

However, some clue to the contents of the pickled object are seen with substrings like “Name,” “Barry Bonds,” “hrs,” and “gms.”

In conclusion, pickling opens a new door for data structures in Python. Further, the data can be ensured through powerful error handling. The combination of these two techniques will surely play an important role in future code development in this course.

# Appendix: Python Script

*# ---------------------------------------------- #  
# Title: Pickling and Error Handling  
# Dev: rblake50  
# ChangeLog:  
# rblake50, 08.17.2021, created script  
# rblake50, 08.24.2021, removed "Edit Player"  
# option (can be developed in the  
# future)  
# ---------------------------------------------- #  
# DESCRIPTION:  
# This script reads a pickled object from a user-  
# specified file and allows users to add player  
# statistics (home runs and games) to a list  
# of dictionary objects. The user can calculate  
# the average number of home runs per 162 game  
# season. All data can be saved (pickled) back  
# to the same user-specified file.  
# ---------------------------------------------- #  
# ASSUMPTIONS:  
# This script assumes that the pickled object  
# being read is in the appropriate dictionary  
# format (Name,hrs,gms). Reading an existing  
# pickled object in a different format will  
# present unhandled errors.***import** pickle  
  
*# -- Declare data types -- #*lstData = []  
dicPlayers = {}  
  
**class** FileIO:  
 *"""This class defines functions for file input and output"""* **def** readFile(strFileName):  
 *"""Reads a serialized file"""* **with** open(strFileName, **'rb'**) **as** fileIncoming:  
 lstData = pickle.load(fileIncoming)  
 **return** lstData  
  
 **def** overwriteFile(strFileName,lstData):  
 *"""  
 This function pickles a list and overwrites it to a file.* **:param** *strFileName (string) of file to save* **:param** *lstData (list) to pickle and save* **:return** *(string) of success message  
 """* **with** open(strFileName, **'wb'**) **as** file:  
 pickle.dump(lstData, file)  
 **return "Success"  
  
class** Present:  
 *"""This class presents information as text."""* @staticmethod  
 **def** printMenu():  
 *"""  
 This method displays the user menu.  
 """* print(**"""  
 =========================================  
 1. Add player  
 2. Calculate average home runs per season  
 3. Exit program (with save option)  
 """**)  
  
 **def** printList(lstName):  
 *"""  
 Prints formatted elements of a list* **:param** *lstName (list) with items  
 """* msg = **"Here are the players in your list:\n"**+**"="**\*32+**"\n"  
 for** item **in** lstName:  
 msg = msg + item[**"Name"**] + **"\n"** print(msg)  
  
**class** Calculate:  
 *"""This class performs calculations of statistics."""* **def** avgHR(name,lst):  
 *"""  
 Calculates the average number of home runs per 162 game season.* **:param** *name (string) of player in the list* **:return** *(float) average HRs per 162 game season  
 """* **for** player **in** lst:  
 **if** name.lower() == player[**"Name"**].lower():  
 index = lst.index(player)  
  
 *# HRs per game* numHRs = lst[index][**"hrs"**] / lst[index][**"gms"**]  
  
 *# HRs per season* numHRs = numHRs \* 162  
  
 **return** numHRs  
  
*# -- Input -- #*strFileName = input(**"What file would you like to open? (Hint: try data.pickle) "**)  
  
*# -- Deserializing data -- #***try**:  
 lstData = FileIO.readFile(strFileName)  
**except** FileNotFoundError:  
 print(**"Ugh! Your file was not found in the immediate directory."**)  
 input(**"You'll be starting from scratch. Otherwise, exit and start over using data.pickle."**)  
  
*# -- Operations -- #***while True**:  
 *# Show current players in list* Present.printList(lstData)  
  
 *# Display menu* Present.printMenu()  
 choice = input(**"What is your selection? "**)  
  
 *# Add player* **if** choice == **"1"**:  
 player = input(**"Who do you want to add? "**)  
 **try**:  
 homeRuns = float(input(**"How many home runs? "**))  
 games = float(input(**"In how many games? "**))  
 **except** ValueError:  
 input(**"D'oh! You need to put a number for both. Let's start over."**)  
 **continue  
 else**:  
 dicPlayer = {**"Name"**:player.title(),**"hrs"**:homeRuns,**"gms"**:games}  
 lstData.append(dicPlayer)  
  
 *# Calculate average* **elif** choice == **"2"**:  
 **try**:  
 choiceCalc = input(**"Who do you want to calculate? "**)  
 HRperSeason = Calculate.avgHR(choiceCalc, lstData)  
 print(choiceCalc.title() + **" hit %.2f home runs per season."** % HRperSeason)  
 input(**"Press ENTER to return to menu."**)  
 **except** UnboundLocalError:  
 print(**"Name not found on list. Try again!"**)  
 input(**"Press ENTER"**)  
  
 *# Exit (and save if desired)* **elif** choice == **"3"**:  
 choiceSave = input(**"Do you want to save? [y]es or [n]o\n"** ).lower()  
 **if** choiceSave == **"y" or** choiceSave == **"yes"**:  
 **try**:  
 FileIO.overwriteFile(strFileName, lstData)  
 **except**:  
 print(**"Where's the file??"**)  
 **else**:  
 input(**"Data successfully pickled to "** + strFileName + **". Press ENTER to exit."**)  
 **break  
 elif** choiceSave == **"n" or** choiceSave == **"no"**:  
 input(**"Good-bye! Press ENTER to exit."**)  
 **break  
 else**:  
 input(**"You entered nonsense... Let's go back to the menu."**)  
 **continue** *# Invalid selection* **else**:  
 input(**"Invalid choice. Press ENTER to return to the menu."**)