Paula Shoup

November 20, 2019

Foundations of Programming: Python

Assignment 07

https://github.com/eldonsdata/IntroToPython-Python-Mod07

Pickling & Exception Handling

Pickling

Pickling is the process of serialization or converting a file into a byte stream. To learn more about pickling I referenced a Pickle tutorial produced by Datacamp. Datacamp recommends pickling a file to save for later use or if it needs to be transferred over TCP or saved to a database. It is not recommended to pickle files that will be used between different platforms or different versions of Python as data may convert incompletely or not at all or it may require additional steps that make pickling no longer useful. The Datacamp article did a good job of explaining why and when to pickle files as well as providing helpful examples of how to pickle a file including when dealing with data from other Python friendly platforms.

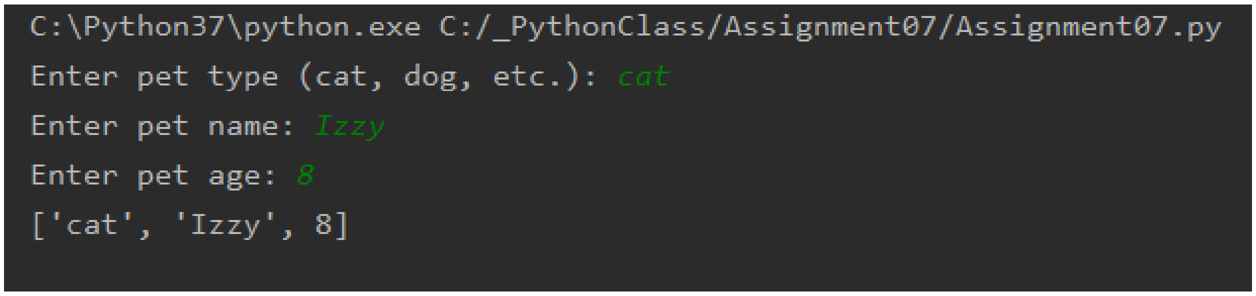
https://www.datacamp.com/community/tutorials/pickle-python-tutorial

The follow example shows how to pickle a file by converting user input regarding pet type, name, and age.

The first step to pickle a file in Python is to import the pickle module. After importing pickle, the most important part of pickling a file is using pickle.dump to convert a file and pickle.load to unpickle data back into Python. In the below example, pickle.dump is used in the save_data_to_file function to convert user data and write it to file. Pickle.load is used in the read_data_from_file function to unpickle the user data. The file and list variables (strFileName and lstPet) are initialized at the beginning of the script to create the ‘PetData.dat’ file for storage. The user input variables (strType, strName, and intAge) allow the user to input pet type, name and age, which will ultimately be saved to the list object lstPet. The end of the script converts the data and displays what has been saved to file.

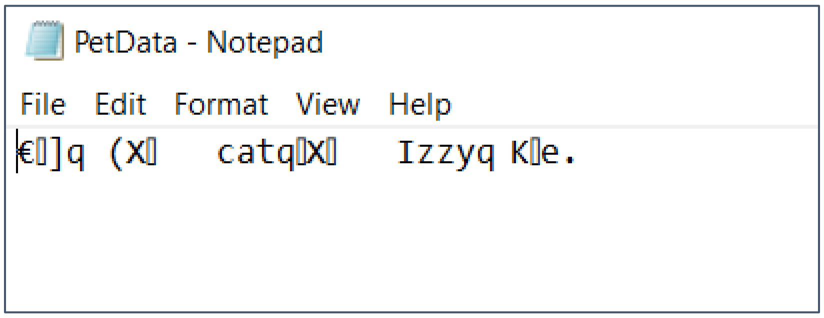
```
import pickle

strFileName = 'PetData.dat'
lstPet = []

def save_data_to_file(file_name, list_of_data):
    objFile = open(file_name, "ab")
    pickle.dump(list_of_data, objFile)
    objFile.close()


def read_data_from_file(file_name):
    file = open(file_name, "rb")
    list_of_data = pickle.load(file)
    file.close()
    return list_of_data


strType = str(input("Enter pet type (cat, dog, etc.): "))
strName = str(input("Enter pet name: "))
intAge = int(input("Enter pet age: "))
lstPet = [strType, strName, intAge]

save_data_to_file(strFileName, lstPet)

print(read_data_from_file(strFileName))
```

Running Assignment07.py in Pycharm displays as follows:



```
C:\Python37\python.exe C:/_PythonClass/Assignment07/Assignment07.py
Enter pet type (cat, dog, etc.): cat
Enter pet name: Izzy
Enter pet age: 8
['cat', 'Izzy', 8]
```

Below is how the pickled input from the PetData.dat file looks like in Notepad.



```
€]q (X    catqX    Izzyq Ke.
```

Exception Handling

If Python encounters an error it causes the current python script to stop and raise an exception message. Exception handling allows the developer to anticipate and intercept errors to prevent a program from crashing. Additionally, through exception handling the developer can control the error message displayed and craft a message that is more user friendly than the default program language intended for developers. The below article from the webpage Real Python has a number of examples that are helpful in understanding how to use and create custom exceptions.

https://realpython.com/python-exceptions/

A good way to anticipate what errors can benefit from error handling is to try to make errors occur when testing your code. In the above example regarding pet data, if the user inputs character data into the pet's age variable (intAge) a ValueError exception will occur. To avoid this happening, a try-except clause can be used to raise a ValueError exception, so the user will know the correct data format. Example below:

```python
try:
    strType = str(input("Enter pet type (cat, dog, etc.): "))
    strName = str(input("Enter pet name: "))
    intAge = int(input("Enter pet age: "))
except ValueError:
    print('Only use numbers for pet age')
else:
    print('Pet data pickled and saved to file')
```