Patrick Chuoy

August 23, 2023

Foundations of Programming: Python

Assignment 07

https://github.com/trickyUW/IntroToProg-Python-Mod07

Pickling and Exception Handling

Introduction

For this week's assignment, we were tasked with searching for resources to learn pickling and exception handling in Python. Then we have to create and document our own examples of how each work. Below are my demonstrations and how I'd explain each topic.

Pickling

I googled "Python pickling" and one of the first sites listed was *Understanding Python Pickling with example*. It looked short and easy to understand, so I decided to use it as my example.

In Python, there is a built-in library called Pickle. Pickling serializes a Python object before writing it to a file (Understanding Python Pickling with example, 2023). It is a way to convert objects like lists into a character stream and then be able to unpickle it to reconstruct the object. In Module 07, it was explained that pickling saves data in a binary format, which obscures the content and may reduce the size.

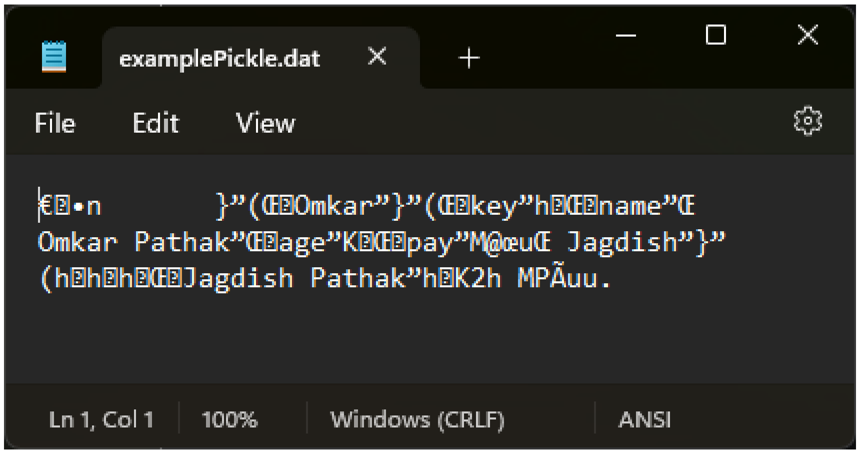
Below is the example code from the website (Listing 1). The only thing I changed was the file. The original was 'examplePickle', which was just a file. I added .dat to because in Module 07 and our textbook, we used DAT files for binary files.

```python
import pickle


def storeData():
    # initializing data to be stored in db
    Omkar = {'key': 'Omkar', 'name': 'Omkar Pathak',
             'age': 21, 'pay': 40000}
    Jagdish = {'key': 'Jagdish', 'name': 'Jagdish Pathak',
               'age': 50, 'pay': 50000}

    # database
    db = {}
    db['Omkar'] = Omkar
    db['Jagdish'] = Jagdish

    # Its important to use binary mode
    dbfile = open('examplePickle.dat', 'ab')

    # source, destination
    pickle.dump(db, dbfile)
    dbfile.close()
```

```python
def loadData():
    # for reading also binary mode is important
    dbfile = open('examplePickle.dat', 'rb')
    db = pickle.load(dbfile)
    for keys in db:
        print(keys, '=>', db[keys])
    dbfile.close()


if __name__ == '__main__':
    storeData()
    loadData()
```

Listing 1: Example Script (Understanding Python Pickling with example, 2023)

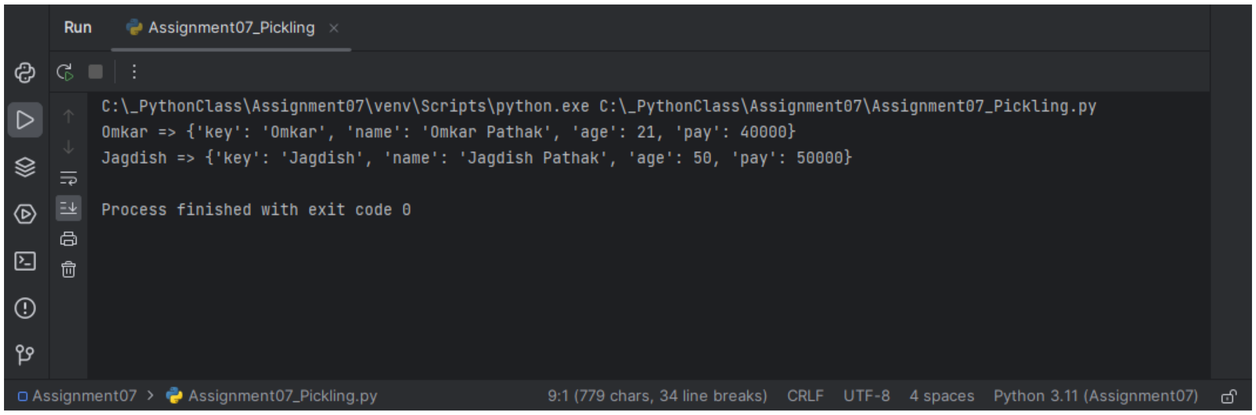
The website gives a short and sweet demonstration of pickling. To use pickling, you have to have, import pickle, like the first statement. The storeData() function uses pickle.dump() to write the data. In the example, a dictionary database (db) that contains two lists of dictionaries is initialized. They use ‘ab’ when opening the file to append a binary file, or create one if it doesn’t exist. The pickle.dump() function writes the dictionary objects into the binary object. The binary file (Figure 1) is mostly obscured and only a couple things are readable by humans.



```
€•n       }”(Œ Omkar”}”(Œkey”hŒname”Œ
Omkar Pathak”Œage”KŒpay”M@œuŒ Jagdish”}”
(hhhŒJagdish Pathak”hK2h MPÃuu.
```

Figure 1: examplePickle.dat

The loadData() function shows an example of using the pickle.load() function to load the dictionary object, unpickle, and then print them. Below is what the output looks like (Figure 2):


Run
Assignment07_Pickling

```
C:\_PythonClass\Assignment07\venv\Scripts\python.exe C:\_PythonClass\Assignment07\Assignment07_Pickling.py
Omkar => {'key': 'Omkar', 'name': 'Omkar Pathak', 'age': 21, 'pay': 40000}
Jagdish => {'key': 'Jagdish', 'name': 'Jagdish Pathak', 'age': 50, 'pay': 50000}

Process finished with exit code 0
```

Assignment07 > Assignment07_Pickling.py
9:1 (779 chars, 34 line breaks) CRLF UTF-8 4 spaces Python 3.11 (Assignment07)

Figure 2: PyCharm output from example script

Exception Handling

When researching exception handling in Python, I found *Python Exception Handling* (Python Exception Handling, n.d.). The tutorial site is neatly organized and easy to read.

Exceptions are unexpected events that occur when running a program. When an exception is found, the program stops running and gives an error message that usually does not help the user. By using exception handling, you can try to catch errors ahead of time and give a more meaningful error message.

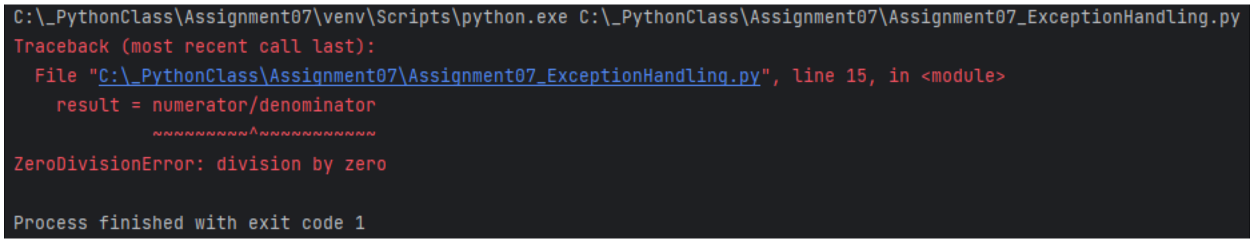
Try-Except Block

The Try-Except Block is used to handle exceptions. In the try block is the code that might give an exception and in the except block is code that runs when an exception does occur.

```python
try:
    numerator = 10
    denominator = 0

    result = numerator/denominator

    print(result)
except:
    print("Error: Denominator cannot be 0.")
```

Listing 2: Try-Except block example (Python Exception Handling, n.d.)

Listing 2 shows a basic example. You cannot divide by zero, so the code generates an exception. In the except block, the example prints a message to say the denominator cannot be 0. Without the Try-Except blocks, Python would output less user friendly (figure 3):



```
C:\_PythonClass\Assignment07\venv\Scripts\python.exe C:\_PythonClass\Assignment07\Assignment07_ExceptionHandling.py
Traceback (most recent call last):
  File "C:\_PythonClass\Assignment07\Assignment07_ExceptionHandling.py", line 15, in <module>
    result = numerator/denominator
             ~~~~~~~~~^~~~~~~~~~~~
ZeroDivisionError: division by zero

Process finished with exit code 1
```

Figure 3: Divide by 0

Catching Specific Exceptions

You can have multiple except blocks to handle different types of exceptions. In the website's example (Listing 3), the code tries to print what is in index [5] of a list that only goes to index [3]. They have specific blocks of built-in Python exceptions: one for dividing by zero and one for when the index is out of bound. Since the code will generate an IndexError exception, the ZeroDivisionError block is skipped and the print statement in the IndexError block is executed.

```python
try:

    even_numbers = [2, 4, 6, 8]
    print(even_numbers[5])

except ZeroDivisionError:
    print("Denominator cannot be 0.")

except IndexError:
    print("Index Out of Bound.")
```

Listing 3: Catching Specific Exceptions example (Python Exception Handling, n.d.)

Try with Else Clause

A Try-Except block can also have an else block at the end. If the code in the try block runs without error and no exceptions are raised, the code in the else block will run.

```python
try:
    num = int(input("Enter a number: "))
    assert num % 2 == 0
except:
    print("Not an even number!")
else:
    reciprocal = 1/num
    print(reciprocal)
```

Listing 4: Try with Else Clause example (Python Exception Handling, n.d.)

The website's example (Listing 4) has the user enter an integer and checks if it's an even number. If it's not even, the exception block will run and print a message. If it is even, the else block runs and prints the number's reciprocal.

Try-Finally

If you have a finally block, it is always executed, whether an exception was raised or not. There can only be one finally block for each try block.

```python
try:
    numerator = 10
    denominator = 0

    result = numerator / denominator

    print(result)
except:
    print("Error: Denominator cannot be 0.")

finally:
    print("This is finally block.")
```

Listing 5: Try-Finally example (Python Exception Handling, n.d.)

In the example given (Listing 5), the code tries to divide by zero and the exception block is run. Whether an exception was raised or not, the finally block would still run and print.

Summary

After completing Module 07, I did supplemental research to find more examples of pickling and exception handling. I used examples I found online and tried to explain how they work in my own words. I copied their code into PyCharm to make sure they work and get a better understanding of pickling and exception handling.

References

Python Exception Handling. (n.d.). Retrieved from Programiz: https://www.programiz.com/python-programming/exception-handling

Understanding Python Pickling with example. (2023, July 15). Retrieved from geeksforgeeks: https://www.geeksforgeeks.org/understanding-python-pickling-example/