

Dmitriy Penzin

February 27, 2022

Foundations Of Programming: Python

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

## Assignment 07 – Exception handling and Pickling mode

The purpose of this assignment is to do research and provide examples on how to use Python's exception handling and Python's Pickling features..

### Exceptions in Python:

- Python stops the current program and displays an error message when runs into an error. Purpose of handling exceptions is to trap potential errors, provide user friendly message and graceful exit.
- Try-Except allows to capture the exception object
- Beneficial at any interaction of user with the program to handle wrong type input, division by zero etc.
- Developer can throw (Raise) exception with "raise" at any time
- "finally" allows to run execute sections of code that should always run, with or without encountered exceptions

### Exception Types:

- IOError – open non-existing file
- IndexError- nonexistent element
- KeyError- dictionary key is not found
- NameError- name (or variable) not found
- SyntaxError- syntax error
- TypeErroroperation or function applied to inappropriate type object
- ValueError- inappropriate value
- ZeroDivisionError- division by zero

### Web Search on Exception Handling in Python:

I found <https://realpython.com/python-exceptions/>

to be the most comprehensive source of the subject. This web page contains visual diagrams of the code to be performed

### Example from previous assignment:

I needed to have a number from user's input in Assignment 04, so I used exception handling as follows:

```
elif menuChoice == '1':
    strItem = input('input item: ')
    try:
        floatValue = float(input('input value: '))
    except ValueError:
        print("\nINVALID ENTRY, please enter number for value")
    lstRow = [strItem.title(), floatValue]
    tbl.append(lstRow)
```

### Pickling data:

- Better way to save complex data than converting to text
- Can pickle numbers, strings, tuples, lists, dictionaries
- Functions: dump, load, shelve

### Web search on data pickling:

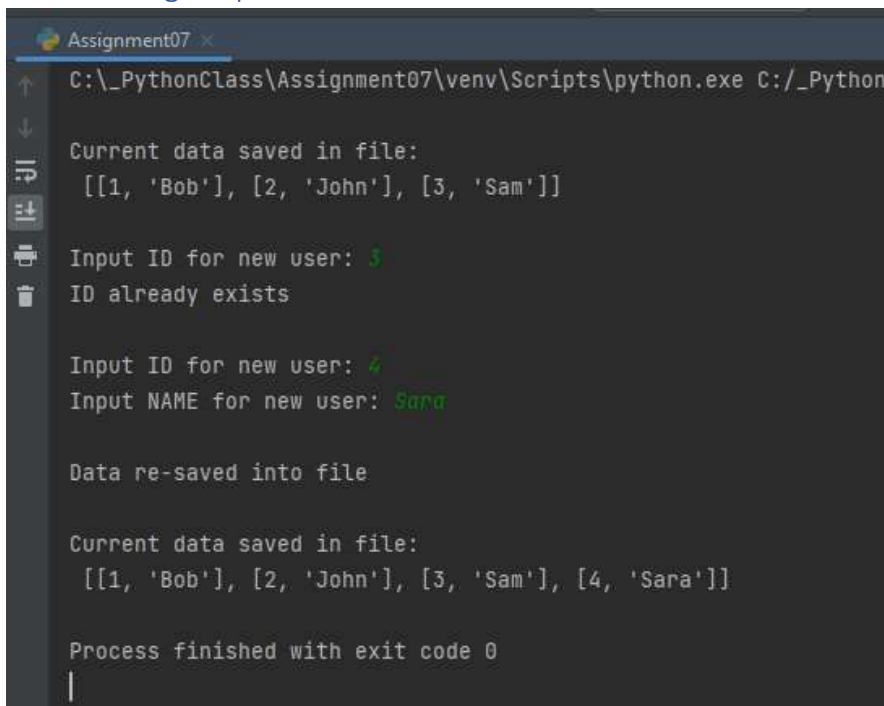
<https://www.youtube.com/watch?v=Pl4Hp8qwwes>

this video appears to be concise and clear on the subject. It also shows “pythonic” way to work with files without the need to close them.

### Steps I took in performing Assignment 07:

1. I watched the module video
2. I read Chapter 7 of the book
3. I searched web for information on pickling and exceptions
4. I created a simple script. It has some hard-coded data I am saving into a file and reading from it. It also allows one input from the user in order to demonstrate re-writing into the file and exception handling. However, I am not using “append” mode. Unpickling after data was pickled in “append” mode returns a separate list and requires additional work to structure it. I chose to overwrite data after the addition of new item as an acceptable method for this simple example.

### Running script:



```
Assignment07 x
C:\_PythonClass\Assignment07\venv\Scripts\python.exe C:/_Python
Current data saved in file:
[[1, 'Bob'], [2, 'John'], [3, 'Sam']]
Input ID for new user: 3
ID already exists

Input ID for new user: 4
Input NAME for new user: Sara

Data re-saved into file

Current data saved in file:
[[1, 'Bob'], [2, 'John'], [3, 'Sam'], [4, 'Sara']]

Process finished with exit code 0
|
```

## Examples of Exceptions in Python:

I added exception into writing data into file function in case of "read-only" file when IO error is generated:

```
17 def write_file(list, file): # Function to write into binary file. Overwrite
18     try: # Handling exception if file is "Read-only"
19         with open(file, "wb") as obj_file:
20             pickle.dump(list, obj_file)
21     except IOError:
22         print("\nNo access to file. Program is terminated")
23         exit()
```

Also, ID from the user should be integer number, so it was handled with ValueError exception:

```
30     try:
31         new_id = int(input("\nInput ID for new user: "))
32         if new_id > 3:
33             break
34         else:
35             print("ID already exists")
36             continue
37     except ValueError: # raised when value for ID is not integer
38         print("\nID should be an integer")
```

## Running from CMD:

```
cmd Command Prompt
c:\_PythonClass\Assignment07>Python Assignment07.py

Current data saved in file:
[[1, 'Bob'], [2, 'John'], [3, 'Sam']]

Input ID for new user: 4
Input NAME for new user: Sara

Data re-saved into file

Current data saved in file:
[[1, 'Bob'], [2, 'John'], [3, 'Sam'], [4, 'Sara']]

c:\_PythonClass\Assignment07>
```

## Summary:

During the work on this assignment, I created a simple script with writing and reading data from a binary file. This script also demonstrated basic exceptions. Module07 video and notes were very helpful.