Johnathan Luu
Intro to Python
Assignment 7
5/24/2022
https://github.com/LofiLogan/

Pickling and Structured Error Handling

Intro

Pickling and structured error handling were the main focus of this week's module. Simply put, pickling is the process of converting information into byte or binary form. Exception handling can be used in many ways. At the simplest level, the Python language has some exceptions implemented into the system by default to pick out specific errors the script may run across. However, exception handling also becomes more customized when the script writer creates custom exceptions if the script performs a certain way outside the parameters of what is expected.

Pickling in Python

As defined by many online sources, pickling "serializes" the intended object first before writing it to a file. It is the method used to convert a python object such as a list or dictionary into a character stream, which will contain all the information needed to put the object or information back together in another python script.

Source: https://www.geeksforgeeks.org/understanding-python-pickling-example/

Structured Error Handling

Structured error handling is a great way to define and identify errors as objects. Python by default as some set up. These errors can assist in identifying reasons a script is not running as expected. In addition, this also allows the script writer to define their own error types. For example, in the script created in this assignment, a range of numbers was used. When the person running the script inputs a number outside the defined range, however, an error will show, telling the user to input a number inside that range.

Source: https://docs.progress.com/bundle/openedge-abl-error-handling-117/page/What-is-structured-error-handling.html#:~:text=Structured%20error%20handling%20is%20an,your%20own%20error%20types%20(classes)

Creating the Program

Pickling:

The example of pickling in this script is shown by a function created to write to a file in binary, as well as a second function used to read the information from the file.

```
#Save data to binary file
def SaveData(FileName, DataLst):
    pFile = open(FileName, "wb")
    pickle.dump(DataLst, pFile)
    pFile.close()

#Read data from binary file
def ReadData(FileName):
    pFile = open(FileName, "rb")
    DataLst = pickle.load(pFile)
    pFile.close()
    return DataLst
```

Exception Error Handling:

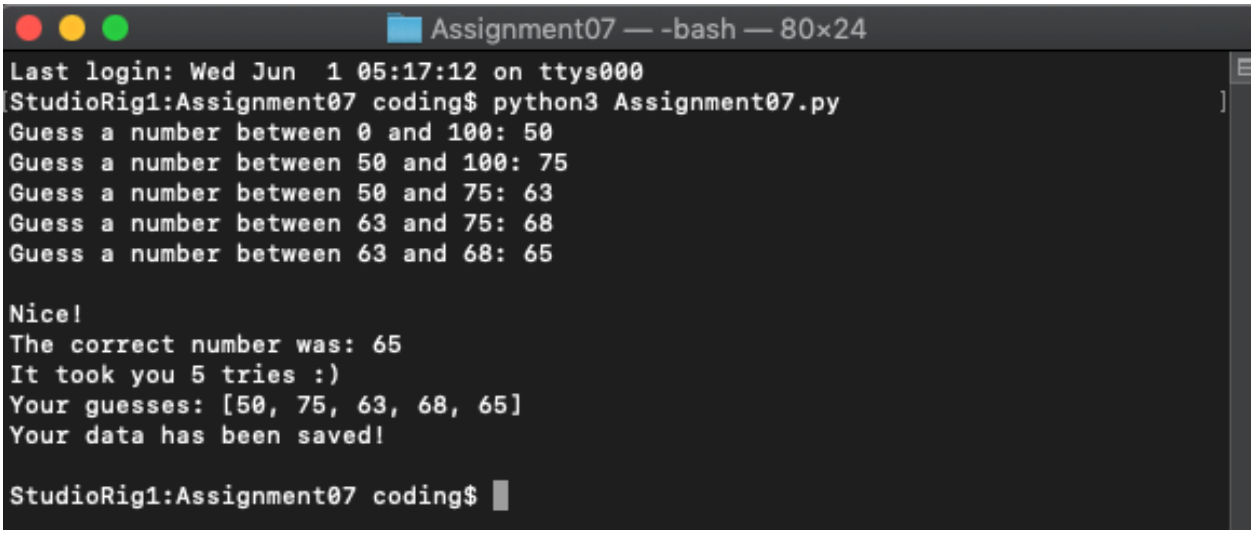
Exception Error Handling is show in this script, first with a custom exception called ‘RangeError’, which will show if the user. Inputs a number outside the given range.

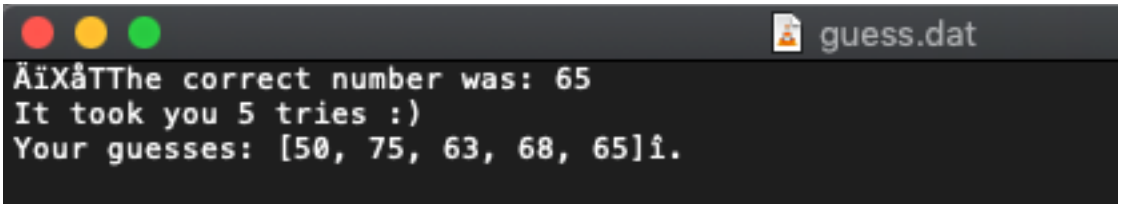
```
#If user guesses a number outside of range, RangeError exception will occur
class RangeError(Exception):
    """ Number must be between Minimum and Maximum"""
    def __str__(self):
        return "Number is not in range"
```

It is followed later in the script by a try except that defines the situation in which a RangeError() will be activated. After the conditions are defined, the errors also have to be decided how they will appear on the screen in the situation that they are raised.

```
try:
    Guess = int(input("Guess a number between " + str(Low) + " and " + str(High) + ": "))
    if Guess >= High or Guess <= Low: #Reference to RangeError exception class
        raise RangeError()

#Determines how RangeError exception is to be displayed
except RangeError as e:
    print()
    print(e, e.__doc__, type(e), sep="\n")
    print()

#Determines how ValueError exception is displayed if user does not type in an integer
except ValueError as e:
    print()
    print("A whole number was not inputed")
    print(e, e.__doc__, type(e), sep="\n")
    print()

#To catch any other errors by default Python system
except Exception as e:
    print()
    print("There was a non-specific error!")
    print("Built-in Python error info: ")
    print(e, e.__doc__, type(e), sep="\n")
    print()

#If no exceptions, proceed with decreasing range until user guess the number
else:
    High = Numbers.Higher(rGuess=Guess, rHigh=High, rNumber=TheNumber)
    Low = Numbers.Lower(rGuess=Guess, rLow=Low, rNumber=TheNumber)
    Counter += 1
    GuessesLst.append(Guess)
```

Results


Assignment07 — -bash — 80×24
Last login: Wed Jun 1 05:17:12 on ttys000
StudioRig1:Assignment07 coding$ python3 Assignment07.py
Guess a number between 0 and 100: 50
Guess a number between 50 and 100: 75
Guess a number between 50 and 75: 63
Guess a number between 63 and 75: 68
Guess a number between 63 and 68: 65
Nice!
The correct number was: 65
It took you 5 tries :)
Your guesses: [50, 75, 63, 68, 65]
Your data has been saved!
StudioRig1:Assignment07 coding$


guess.dat
ÄïXåTThe correct number was: 65
It took you 5 tries :)
Your guesses: [50, 75, 63, 68, 65]î.