David Petkov

May 31, 2023

IT FDN 100 A

Assignment 7

Module 7 - GitHub

Creating a Guessing Game Using Python Exceptions and Pickling

Preliminary Comment Before Beginning the Report

I requested a 2-day extension for this assignment because I wasn't able to finish it together with Module 6 by Wednesday, May 31. I returned from vacation last week and was swamped with work on top of classwork. Thank you for your understanding.

Introduction

In this assignment, we are tasked with creating a script to demonstrate what we have learned about exception handling and pickling in Python from sources outside the classroom. The script I developed is a simple demonstration that combines both methods into one interactive program that I built a simple guessing game around.

Building the Code

The first concept I focused on was exception handling. After referencing multiple sources, I quickly realized that there are multiple methods and strategies that can be used with exception handling depending on what the programmer is trying to achieve. The most basic of all is the simple try-except strategy, try this code, except if it doesn't work, run this code instead.

There are also many different types of errors you can get from a program, and the exception method allows you to have a different output depending on the type of error. This was something I wanted to demonstrate when building my code, to make it interactive depending on the type of output received from the try portion.

My program is a simple game of guessing the lucky number. In the try section, the user is asked to pick any number. There are four ways to get an exception, an unlucky guess resulting in an index error, a type error due to an incorrect input, an out-of-range guess by guessing beyond 10, and a general exception

for when the user finds a way to break the game. Due to my inexperience, I have not been able to break my own game yet, only when changing the code to fail on purpose. The try-exception portion is shown in Figure 1.

```
try:
    guess = abs(int(input("Gambler: Bet $10 you can pick my lucky number? Select any number between 1 and 10: "))) - 1
    if guess > 9:
        raise Exception
    elif guess == 2:
        print("\nGambler: Annnnd...", list_of_things[guess])
        SoreLoser()
        trick = input("The Game: Would you like to perform a jedi mind trick? [y/n]: ")
        if trick.lower() == 'y':
            JediMindTrick()
        elif trick.lower() == 'n':
            print("\nThe Developer: You are truly a noble one, may the force be with you")
        else:
            print("\nThe Game: You had a mishap and jedi mind tricked yourself!\n")
            CounterProductive()
    else:
        print("\nGambler: Annnnd...", list_of_things[guess])

except IndexError as IndErr:
    print("\nGambler: Tough luck pal, next time...")

except ValueError as ValErr:
    print("""
    The Game:
    You were supposed to pick a number buddy
    You never had a chance :(
    """)

except TypeError as TypErr:
    print("Gambler: Oh, why thank you for giving me $20! Bet was only $10 but I'll take it!")

except Exception as ExcErr:
    print("\nGambler: ... Everything ok in your head? I said 1 to 10 and you give me", guess+1, "?")

except:
    print("The Developer: Oh you really broke the game...")
```

Figure 1: The try-exception portion of the program

The pickling portion of my script is rather simple. Once the lucky guess is made, the SoreLoser() function is run which creates a binary file and stores a string inside it using the dump() function. The opening and closing of binary files are similar to text files, but the mode with which you edit them is slightly modified, you need to indicate that you are editing a binary file, in my case, by adding the letter ‘b’ to the append mode to get ‘ab’.

We use pickling for data serialization between network applications that need to exchange data with other programs. It is useful to use when saving Python objects between sessions and processes. It is important to note, however, that it is not very secure. It may be unreadable to humans, but it can be easily decoded by a program and if used for malicious purposes if the wrong files are unpickled. The pickling portion of the program is shown in Figure 2.

```python
guess = None
import pickle
list_of_things = ['Nope', 'So Close', 'Winner!...oh no...(dramatic pause)', 'Loser', 'Ouch']


1 usage
def SoreLoser():
    print("\nGambler: This can't be.. I simply cannot lose, if you want my money you will need to")
    print("use a jedi mind trick to guess my ultra secret password to get my $10\n")
    file = open('SecretFile', 'ab')
    password = "Password"
    pickle.dump(password, file)
    file.close()
    return

1 usage
def CounterProductive():
    raise TypeError


1 usage
def JediMindTrick():
    file = open('SecretFile', 'rb')
    read = pickle.load(file)
    print("\nGambler: I will give you the password. My password is:", read)
    file.close()
    return
```

Figure 2: The pickling portion of the program

Running the Program

The program begins with the gambler asking you to pick a number between 1 and 10. As stated above, the user will either pick the wrong number (IndexError or pick an item from a list that isn't the lucky number), an out-of-range number (Exception), a letter (ValueError), or the correct number. Figure 3 shows the various losing options which are accessed by either raising an exception or creating an exception for a specific fault error.

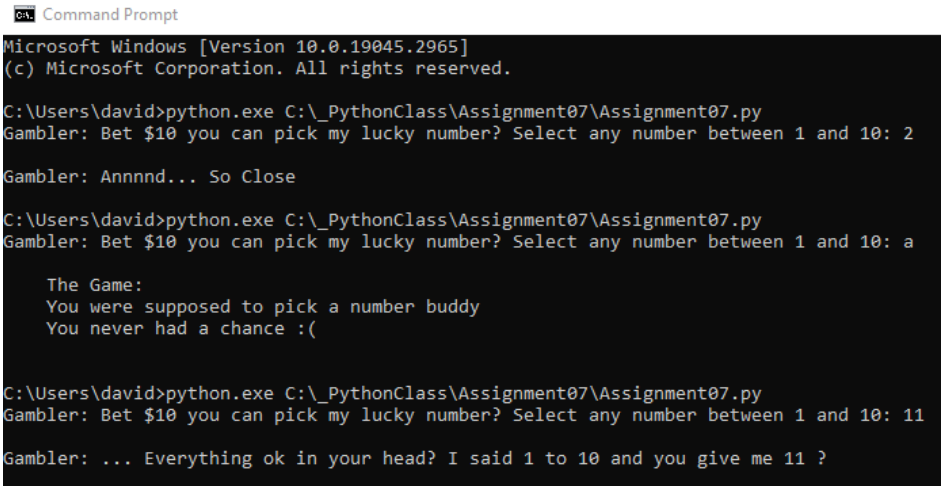

Figure 3: The exceptions running when the user guesses the wrong number

If the user makes the lucky guess, the SoreLoser() function is run which creates a binary file and appends a “secret” password to it. The user now has new options. If the user wants to win the gamble, they can mind tricking the gambler by running the JediMindTrick() function. The function reads the binary file to get the password and their money. The user can also choose to be noble and not trick the gambler. (Figure 4)

```
C:\Users\david\AppData\Local\Programs\Python\Python311\python.exe "C:\Python\_Python (
Gambler: Bet $10 you can pick my lucky number? Select any number between 1 and 10: 3

Gambler: Annnnd... Winner!...oh no...(dramatic pause)

Gambler: This can't be.. I simply cannot lose, if you want my money you will need to
use a jedi mind trick to guess my ultra secret password to get my $10

The Game: Would you like to perform a jedi mind trick? [y/n]: y

Gambler: I will give you the password. My password is: Password
```

Figure 4: The user makes the lucky guess and selecting to get the password, which runs the JediMindTrick() function to read the binary file for data

However, if the user makes an incorrect selection by selecting anything that isn’t the letters ‘y’ or ‘n’, the CounterProductive() function is run, which raises a TypeError and runs that exception. In reality, it isn’t a type error since any user input will automatically be a string which is what the program expects, but this was just to demonstrate the concept of raising an exception outside the try-except portion of the script. (Figure 5)

```
Gambler: Bet $10 you can pick my lucky number? Select any number between 1 and 10: 3

Gambler: Annnnd... Winner!...oh no...(dramatic pause)

Gambler: This can't be.. I simply cannot lose, if you want my money you will need to
use a jedi mind trick to guess my ultra secret password to get my $10

The Game: Would you like to perform a jedi mind trick? [y/n]: h

The Game: You had a mishap and jedi mind tricked yourself!

Gambler: Oh, why thank you for giving me $20! Bet was only $10 but I'll take it!
```

Figure 5: The user makes a mistake, which runs the CounterProductive()

Because the pickle module is imported, the name must be included before the load and dump functions so that the program works, otherwise, you would actually “break” the game. (Figure 6)

```
Gambler: Bet $10 you can pick my lucky number? Select any number between 1 and 10: 3

Gambler: Annnnd... Winner!...oh no...(dramatic pause)

Gambler: This can't be.. I simply cannot lose, if you want my money you will need to
use a jedi mind trick to guess my ultra secret password to get my $10

The Developer: Oh you really broke the game...
```

Figure 6: Breaking the game is much easier when the code isn’t correct

Helpful Resources

Apart from the textbook and class resources, I was able to find a few helpful resources online that helped me better understand exceptions and pickling in Python. Reference [1] was excellent in demonstrating a wide variety of methods for using exceptions. The website provided sample code and thorough explanations for different types of examples and it was a very useful resource, but it didn't have all the information I needed.

Reference [2] provided information on the different types of exceptions that are built into Python. This helped complete my understanding of exceptions to create a more encompassing program to include more types of errors.

Reference [3] provided some useful descriptions of what pickling does and the different types of functions that exist and what they do. The website as a whole is very useful but not always the best resource, particularly when working with a new concept or idea for the first time.

Reference [4] provided simpler examples and explanations of pickling which was what I needed to complete my program for Module 7.

Conclusion

The final script of the program helped to demonstrate various ways of using the try-exception method and how helpful it could be in certain circumstances. The program also made use of the pickling method, albeit using simpler examples because I wanted to refrain from making my code look like a copy and paste from the internet. The method frankly is very straightforward.

Resources

[1]: *8. Errors and Exceptions*. (n.d.). Python Documentation. https://docs.python.org/3/tutorial/errors.html

[2]: GeeksforGeeks. (2023). Python Exception Handling. *GeeksforGeeks*. https://www.geeksforgeeks.org/python-exception-handling/

[3]: *pickle — Python object serialization*. (n.d.). Python Documentation. https://docs.python.org/3/library/pickle.html#:~:text=%E2%80%9CPickling%E2%80%9D%20is%20the%2%200process%20whereby,back%20into%20an%20object%20hierarchy.

[4]: *Python Pickling*. (n.d.). https://www.tutorialspoint.com/python-pickling