Emmanuel Ihejirika
June 3 , 2023
IT FDN 100 A
Assignment 07

INTRO TO PROGRAMMING (PYTHON)

Intro

In the 7th week of the Foundations of Programming: Python course, we learned how to create our own functions, use the pickle function and its accessories (.load and .dump), and the try-except function and use them to create a program. With this new knowledge we were supposed to implement the last hw assignment into this current one, using the def function tool. We also applied what we learned the previous week to this assignment. The assignment attached to this lecture asks us to create a program that will display a program of our own choosing as long as it contains the functions: pickle and try-except. Pickle is a way to modify files just like we have been for the past few weeks. But it can store the information in a data file or text file, as a binary so it is difficult to read. The try-except function will allow users to enter inputs that normally cause python to throw back a built in error message, but instead of the program crashing, you can send your own custom error message telling the user to what the error is and what input type to enter instead and allow them to try again without restarting the code. I was out of town for the past week so this information is a bit behind what we are currently on.

Creating the Program

After reading the assignment prompt, I went back to the notes the professor used to create the youtube lecture videos and read the different functions and how they are used. I saw he also had lab 7-1 starter code and I decided to use that to start my program. From what I learned in assignment 6 about setting up custom functions, I took another shot at them and applied it to this program. I created 3 functions: def get_user_data, def save_data_to_file, and def read_data_from_file shown in *figure 1*.

Part of the assignment was to also implement the try-except function. So I decided to use it in the main menu section of my code so that if the user entered anything other than the option numbers it will tell them to correct that and continue the program, see *figure 2*. After that I, better than assignment 6, able to see each function and understand what they needed like a puzzle and then make each fit together in the end. Which is what I struggled with last time, I could see the trees but not the forest so to speak. *Figures 3*-5 will show what each custom function contains.Get_user_data will ask the user for the information that will go into the file, def save_data_to_file will save the data the user entered and put it in the file, and def

read_data_from_file will take the information that is in the file and read it out to the user.

```
12

    2 usages
13  def get_user_data():...
19  # Processing ------------------------------------ #
    1 usage
20  def save_data_to_file(file_name, lstcustomer):...
26

    1 usage
27  def read_data_from_file(file):...
32
```

Figure 1. Custom Functions

```
while True:
    try:
        print('Hello, welcome to the file saver')
        choice = int(input('''Make a choice:
        1. Get user data
        2. save data from file
        3. read data from the file
        4. Close the program
        '''))
        if choice == 1:
            get_user_data()
        elif choice == 2:
            lstcustomer = get_user_data() #not sure why code bugs out when this line isnt here
            save_data_to_file(file_name, lstcustomer)
        elif choice == 3:
            read_data_from_file(file_name)
        elif choice == 4:
            break
    except ValueError:
        print('INVALID CHOICE! Enter 1, 2 or 3!')
```

Figure 2. Try-except block and menu

```
2 usages
def get_user_data():
    intId = int(input('Enter an ID:'))
    strName = str(input('Enter a name: '))
    lstcustomer = [intId, strName]
    print(lstcustomer)
    return lstcustomerS
```

Figure 3. Def get_user_data()

```
def save_data_to_file(file_name, lstcustomer):
    with open(file_name, 'ab') as file:
        pickle.dump(lstcustomer, file)
    print(file)

    pass  # TODO: Add code here
```

Figure 4. Def save_data_to_file ()

```
1 usage
def read_data_from_file(file):
    with open('AppData.dat', 'rb') as file:
        objFiledata = pickle.load(file)
        print(objFiledata)
    pass  # TODO: Add code here
```

Figure 5. Def read_data_from_file()

The program, when done correctly, will provide the user with a menu of options to choose from, shown in figure 2, get user data, save data from file, read data from file, and end program. *Figure 6* will show the program also works in the command line. I followed along with the lecture video and did my best to comprehend each step and attempt them myself.

```
Make a choice:
        1. Get user data
        2. save data from file
        3. read data from the file
        4. Close the program
        w
INVALID CHOICE! Enter 1, 2 or 3!
Hello, welcome to the file saver
Make a choice:
        1. Get user data
        2. save data from file
        3. read data from the file
        4. Close the program
        1
Enter an ID:4
Enter a name: Bob
[4, 'Bob']
Hello, welcome to the file saver
Make a choice:
        1. Get user data
        2. save data from file
        3. read data from the file
        4. Close the program
        2
Enter an ID:4
Enter a name: sarah
[4, 'sarah']
<_io.BufferedWriter name='AppData.dat'>
Hello, welcome to the file saver
Make a choice:
```

Figure 6. Script in CMD

Summary

I create a script that will display a menu to the user and ask the user what options they want to choose. They will have the option to add data to a binary file, read the data, save the data, or end the program. The main focus of this program is to demonstrate the pickle and try-except function and how they work. What I took from this was how to use the pickle function, try-except function, and outside of the assignment scope make the program more efficient by using the **with** open (Appdata.dat) as file function which will open perform an action and close the file without the having to tell it to. I was able to understand def functions better this time around and hope that it is a sign of a brighter future in programming.