Ronald Mursewick

November 19, 2023

Foundations of Programming:

Python Assignment 06

https://github.com/Visual-Library/IntroToProg-Python

Functions w and w/o parameters and return values, and github.

# Introduction

The purpose of this writing is to document some enhancements to our prior program. In this iteration of our program we will define and add function calls with and without parameters. Functions can be used in a variety of ways. One use of a function we can pass a value to the function and have the function use the variable or variables to operate on. In another function we may not pass information to the function but expect the function to return data to us. One of these extensions is to use a list to hold our temporary student data before we write it to a .json file. The change here is writing to a json file. JSON stands for Java Script Object Notation. JSON is a more extensible format structure. It is used widely in web applications. In assignment 04 we incorporated pandas methods for managing out in program data structure. In this lesson we will also use try-raise-except to verify that we are getting clean data into our class file.

# Functions

# Our first function definition. We pass text string to the function for printing to the console.

# A black background with text Description automatically generated

# ***Figure 1. Function definition with parameter passing and no return***

# Our next function we do not pass any parameters to the function but we expect the function to return data to us.

# A screen shot of a computer Description automatically generated

# ***Figure 2. Function definition with only data return from function***

# Our third function example. This function print output to the console.

# A screen shot of a computer Description automatically generated

# ***Figure 3. Function to output data to the console***

# Pandas

# A screen shot of a computer Description automatically generated

# ***Figure 4. Importing Pandas methods and creating the student data DataFrame***

# Our DataFrame will be 5 columns wide and able to have rows added indefinitely. In the DataFrame creation above we are creating the header row for the data.

# A computer screen with white text Description automatically generated

# ***Figure 5. Appending the newly added student to the DataFrame***

# One of the changes is using pandas over variables. Notice the “0” just to the left of the last name “Harrison”. When the operator selects option “2”, we print the DataFrame. With indexing the “zero” row is the first row of the array. Since Harrison was our first student entered he is saves in the table in row zero

# A screen shot of a computer Description automatically generated

# ***Figure 6. Shows how data is stored in the DataFrame***

# Moving the student data from the DataFrame we build with the information the operator entered during the program run. Another reason for using pandas is ease of data handling. Here we see that with a one line command (method) we are able to write the entire contents of the DataFrame to the .json file. We choose the mode=”a” so we append our new data to any other data that has been entered into the file.

# We will also use the orient=”records” to keep our data in a tabular format.

# 

# ***Figure 7. Use of the DataFrame to .json file method.***

# In our program we will also show the use of the list and dictionary operations with output to a second .json file.

# A screen shot of a computer Description automatically generated

# ***Figure 8. List creation utilizing dictionary format***

# A screen shot of a computer Description automatically generated

# ***Figure 9. Appending the data just entered to our running list.***

# Finally in option 3 we will create or append our data to a second .json file. Note the format varies slightly. We will use the json dump method to perform this operation.

# A screen shot of a computer Description automatically generated

# ***Figure 10. Output to file using json dump.***

# Another process introduced in this lesson is validating data with the try-raise-except. The way we use the check in out code is to verify that the first and last names are alphabetical. We use the .isalpha method to test the operators input to confirm that it is a name.

# A computer screen shot of a computer code Description automatically generated

# ***Figure 11. using try-raise-except example***

# Program Operation

# First we will choose operation 1 to register a student.

# A screenshot of a computer program Description automatically generated

***Figure 12. Menu selection 1***

Now the operator chooses select 2, so we show the operator what data has been entered.

A computer screen shot of a program

Description automatically generated

***Figure 13. Menu selection 2***

In menu select 3, we save the data to the .csv file. In this case another operator has entered two other students. In the beginning of the program we checked to see if a student data file existed. If the file existed we opened the for “append” to we could add data to the already registered student list. After we write the latest student to the file we show all of the students that have been registered for the class. Another change in the output for selection 3, was to change $stdout to a file and print the enrollment list to a file. Once the output is complete change $stdout back to the console for dialogue to the operator.

A computer screen shot of a program

Description automatically generated

***Figure 14. Menu selection 3***

Menu selection ends the program and returns us to the command prompt.

A screen shot of a computer program

Description automatically generated

***Figure 9. Menu selection 15***

We put some error processing code in so that if the operator tried to show the current data before a student was registered, we would let them know that they needed to enter student data.

A screen shot of a computer program

Description automatically generated

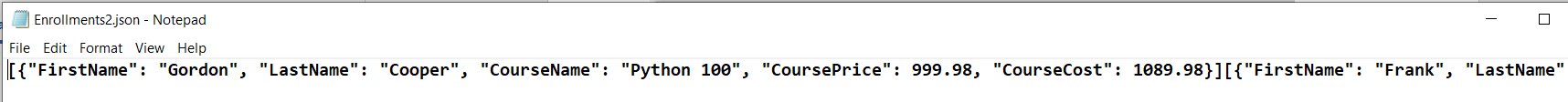
***Figure 16. Menu selection 2, when no student has been registered.***

The Enrollment.json data file as displayed in Notepad.

A screenshot of a computer

Description automatically generated

***Figure 17. Enrollments.json data file***



***Figure 18. The Enrollments2.json data file***

A screenshot of a computer

Description automatically generated

***Figure 19 Reassigning $stdout to a .txt file for printing***

# Summary

We continue to build upon our first assignment. Now we add operations to perform repetitive tasks until a specified condition is met. We are creating function for repetitive procedures in our code. The function be used in a variety of forms: Parameter(s) requires, no parameters, data returned, only output. With adding complexity we want to look for methods that enhance our ability to manage larger amounts of data as efficiently as possible. We show how data can be used in lists with dictionary and Pandas is a powerful tool to aid us with data management and manipulation. We change out output file type to json since it is widely is for data manipulation today. We also add data checking to ensure that the data we write to out file is clean or normalized to minimize the chance for errors or misinterpretation of data later in the processing.