Alan Martin
02/18/2024
Foundations of Python
Assignment_06

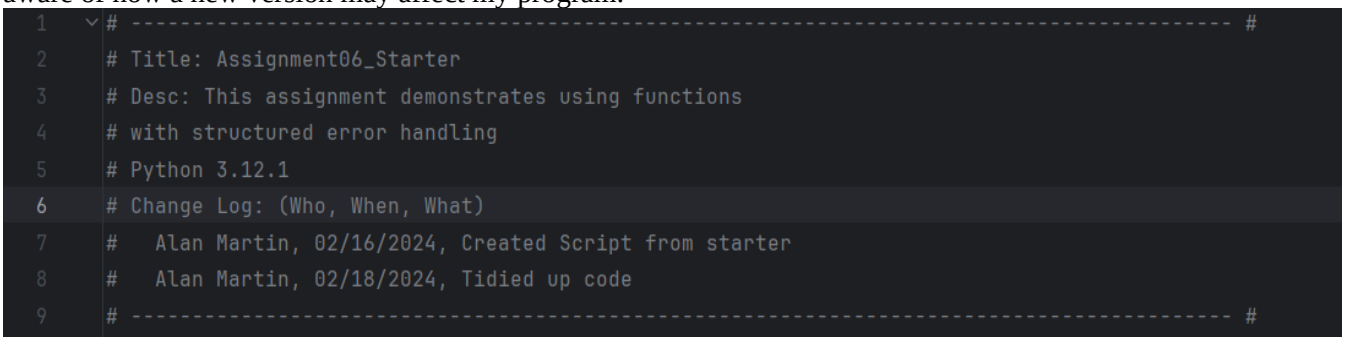
Classes, Functions (Methods), and Structured Error Handling

Introduction

In this document I'll be walking though creating python script that uses a loop function to cycle through selections on our menu until we decide to quit our program, reaching the break statement. We'll first use selection #1 to take user input, selection #2 will print the input receive from user, selection #3 will open-save-close file with user data, selection #4 will break the loop and exit the program.
This week we were introduced to classes and functions(methods), and the benefits they serve in programming (encapsulation, modularity, reusability, etc.). We'll create a doc string to describe each class and function(method) to better explain its usage, purpose and behavior. We'll continue to use JSON files to read data into our program and write data from our program into our JSON file.

Setting python script header

Setting the script header from the start helps document your work. Included I have my title, a description, python version, change log describing who, what, when. I added Python version so I can become more aware of how a new version may affect my program.



```
# ------------------------------------------------------------------------------------------ #
# Title: Assignment06_Starter
# Desc: This assignment demonstrates using functions
# with structured error handling
# Python 3.12.1
# Change Log: (Who, When, What)
#   Alan Martin, 02/16/2024, Created Script from starter
#   Alan Martin, 02/18/2024, Tidied up code
# ------------------------------------------------------------------------------------------ #
```

Import libraries

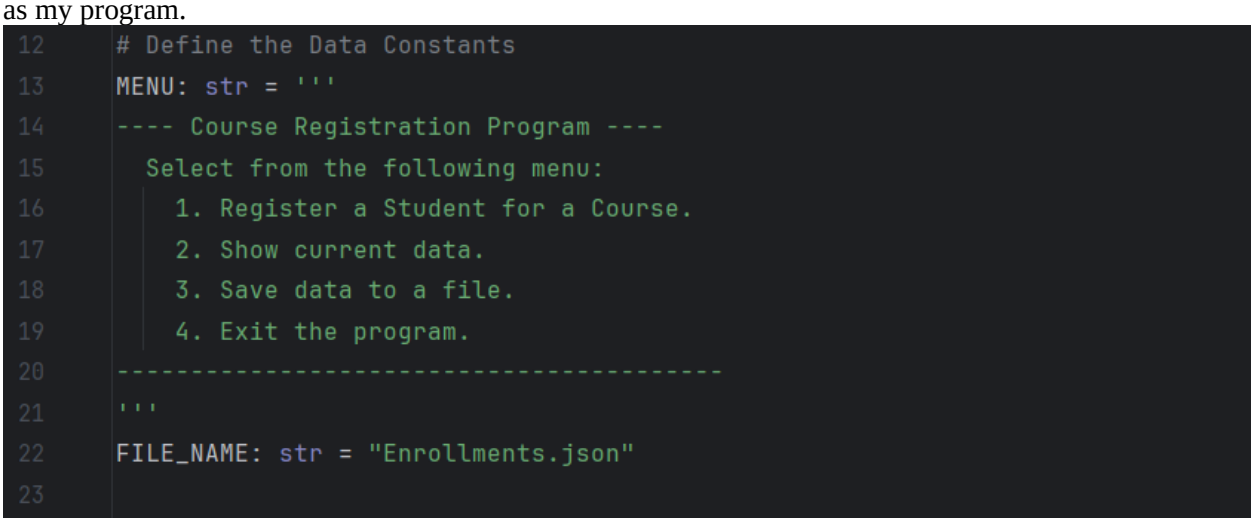
Since we'll be dealing with Json files, we needed to import the Json library.

```
# ------------------------------------------------------------------------------------------ #
import json

```

Defining the Constants

The constants in this program were predefined. The **MENU** constant will become the visual instruction for the user. The **FILE_NAME** constant is set to **Enrollments.json** and will end up saved in the same folder as my program.



```
# Define the Data Constants
MENU: str = '''
---- Course Registration Program ----
  Select from the following menu:
    1. Register a Student for a Course.
    2. Show current data.
    3. Save data to a file.
    4. Exit the program.
----------------------------------------
'''
FILE_NAME: str = "Enrollments.json"

```

Defining the Variables

These variables names have been predefined, but their values have not been set. **menu_choice** holds the choice made by the user and directs the loop. **Students variable holds a table of student data**

```
24    # Define the Data Variables
25    menu_choice: str = " "  # Hold the choice made by the user.
26    students: list = []  # a table of student data
27
28
```

Identifying our classes and Methods

Our first class is **FileProcessor.** The triple double-quotes (” ” ”) signals the docstring, which is where you want to define the classes’ purpose, usage, and behavior, along with your change log.

```
29    class FileProcessor:
30        """
31        Processing layer functions designed to interact with JSON files.
32
33        Alan Martin, 02/16/2024, created class FileProcessor
34        """
35
```

Functions/Methods

Example

```
def write_data_to_file(file_name: str, student_data: list):
```

Def = define the function/method
write_data_to_file = function name (best to use lower case)
(file_name:str, student_data:list) = parameters, arguments will be provided when methods are called
:str and :list are type hints

@staticmethod, class FileProcessor, Method read_data_from_file

Method is the term used when a function sits inside a class, otherwise we'll call it a function. In this foundations class it may be used interchangeably, but let it be known that there is a difference. @staticmethod means that the method can be called from the class without an instance of the class. In this method, we're opening our json file while using some error handling seen in the try except, finally functions. b **IO.output_error_messages** in the except function will call the method. Inside the parenthesis on the method are parameters (**file_name**: str, **student_data**: list), arguments for these parameters will be seen later when the program runs.

```
36      @staticmethod
37      def read_data_from_file(file_name: str, student_data: list):
38          """
39          A FileProcessing function designed to read JSON files into program.
40
41          Alan Martin, 02/16/2024, created method read_data_from_file
42          """
43          file = " "
44          try:
45              file = open(file_name, "r")
46              student_data = json.load(file)
47              file.close()
48          except FileNotFoundError as e:
49              IO.output_error_messages( message: "Text file must exist before running this script!", e)
50          except Exception as e:
51              IO.output_error_messages( message: "There Was A Non-Specific Error!", e)
52          finally:
53              if not file.closed:
54                  file.close()
55          return student_data
56
```

@staticmethod, class FileProcessor, Method write_data_to_file

Reference @staticmethod, class FileProcessor, Method read_data_from_file. This class method writes student_data table to file using json.dump. This file should be saved in the same folder as your program. Error handling is used here as well, along with calls to other methods.

```
57      @staticmethod
58      def write_data_to_file(file_name: str, student_data: list):
59          """
60          A FileProcessing function designed write program data into JSON files.
61
62          Alan Martin, 02/16/2024, created method write_data_from_file
63          """
64          file = " "
65          try:
66              file = open(file_name, "w")
67              json.dump(student_data, file)
68              file.close()
69              print("\n-----Your Data has been saved-----\n")
70          except Exception as e:
71              IO.output_error_messages( message: "Error: There was a problem with writing to the file", e)
72          finally:
73              if not file.closed:
74                  file.close()
```

Identifying our classes and Methods. Class IO

This class manages user input and output.

```
77  class IO:
78      """
79      A collection of presentation layer functions that manage user input and output
80
81      Alan Martin, 02/16/2024, Created IO Class
82      """
83
```

@staticmethod, class IO, Method output_error_messages

This class method helps with error handling messages. In class **FileProcessor**, you'll see both methods **read_data_from_file** and **write_data_to_file** have calls to **IO.output_error_messages.** The **output_error_messages** method is called when the **except** function is triggered in the **FileProcessor** class methods, printing messages specific to the error.

```
84      @staticmethod
85      def output_error_messages(message: str, error: Exception = None):
86          """
87          This function displays custom error messages to user
88
89          Alan Martin, 02/16/2024, Created output_error_messages function
90          """
91          print(message, end="\n\n")
92          if error is not None:
93              print("-- Technical Error Message -- ")
94              print(error, error.__doc__, type(error), sep='\n')
95
```

@staticmethod, class IO, Method output_menu

This method displays our MENU constant for the user.

```python
    @staticmethod
    def output_menu(menu: str):
        """
        This function displays our constant MENU

        Alan Martin, 02/16/2024, Created output_menu function
        """
        print(menu)

```

@staticmethod, class IO, Method input_menu_choice

This method takes the user input and returns it to the user. If the user input isn't string(1, 2 ,3, 4) **raise Exception** print a reminder to the user that only 1, 2, 3, 4 are accepted. If any other errors occur, **IO.output_error_messages** method is called. Again, the user choice is **returned**

```python
    @staticmethod
    def input_menu_choice():
        """
        This function gets user input for the MENU of choices

        Alan Martin, 02/16/2024, Created input_menu_choice function
        """
        choice = "0"
        try:
            choice = input("Enter your menu choice number: ")
            if choice not in ("1", "2", "3", "4"):  # Note these are strings
                raise Exception("Please, choose only 1, 2, 3, or 4")
        except Exception as e:
            IO.output_error_messages( message: "Error: There was a Non-Specific Error", e)
        return choice

```

@staticmethod, class IO, Method input_student_data

This method takes input for a student's first name, last name, and course. Error handling can be seen in **if not student_first_name.isalpha()** and **if not student_last_name.isalpha().** This line of code is flagged if the user enters anything other than alphabetic letters. After first name, last name, and course have been input by the user, that data is stored into the **student: dict,** and then appended to the **student_data: table.** Specific error handling messages can be seen on line 141 and 143. **Return student_data**

```
121     @staticmethod
122     def input_student_data(student_data):
123         """
124         This function that takes student first name, last name, and course
125 
126         Alan Martin, 02/16/2024, Created input_student_data function
127         """
128         try:
129             student_first_name = input("Enter the student's first name: ")
130             if not student_first_name.isalpha():
131                 raise ValueError("The last name should not contain numbers.")
132             student_last_name = input("Enter the student's last name: ")
133             if not student_last_name.isalpha():
134                 raise ValueError("The last name should not contain numbers.")
135             course_name = input("Please enter the name of the course: ")
136             student = {"FirstName": student_first_name,
137                        "LastName": student_last_name,
138                        "CourseName": course_name}
139             student_data.append(student)
140             print(f"You have registered {student_first_name} {student_last_name} for {course_name}.")
141         except ValueError as e:
142             IO.output_error_messages( message: "Error: There was a Value Error", e)
143         except Exception as e:
144             IO.output_error_messages( message: "Error: There was a problem with your entered data.", e)
145         return student_data
146 
```

@staticmethod, class IO, Method output_student_courses

This function displays all student data including their first name, last name, and course. It uses a **for loop, r**eading and printing each student for the user to see.

```
147     @staticmethod
148     def output_student_courses(student_data: list):
149         """
150         This function displays student first name, last name, and course in list
151 
152         Alan Martin, 02/16/2024, Created output_student_data function
153         """
154         print("-" * 50)
155         for student in student_data:
156             print(f'Student {student["FirstName"]} '
157                   f'{student["LastName"]} is enrolled in {student["CourseName"]}')
158         print("-" * 50)
159 
160 # End of Classes and their Methods
161 
```

Read the JSON file data into a list of lists

Students variable now holds the JSON file data from the **read_data_into_file** method, using arguments **FILE_NAME, students.**

```
162     # Read the file data into a list of lists (table)
163
164
165     students = FileProcessor.read_data_from_file(file_name=FILE_NAME, student_data=students)
166
167
```

While loop and present MENU

The **while loop** starts and **IO.output_menu** presents the **MENU** constant shown a the arguement

```
168     # Start the loop
169     while True:
170
171         # Present the MENU
172         IO.output_menu(menu=MENU)
173
```

If conditional statement and menu_choice #1

If menu_choice == 1, **students** variable stores data collected by the **IO.input_student_data method. Continue** starts the **while loop** over

```
177         # Input user data
178         if menu_choice == "1":  # This will not work if it is an integer!
179             students = IO.input_student_data(student_data=students)
180             continue
181
```

elif conditional statement and menu_choice #2

elif menu_choice == 2, output **students** data to the user. **Students** is the argument for the **student_data** parameter.

```
182     # Present the current data
183     elif menu_choice == "2":
184         IO.output_student_courses(student_data=students)
185         continue
186
```

elif conditional statement and menu_choice #3

elif menu_choice == 3, save data to JSON file. The **FileProcessor.write_data_to_file** call uses **FILE_NAME** and **students** arguments. In the method, it takes the data stored in **students** variable and stores it into the **FILE_NAME** constant**.**

```
187     # Save the data to a file
188     elif menu_choice == "3":
189         FileProcessor.write_data_to_file(file_name=FILE_NAME, student_data=students)
190         continue
191
```

elif conditional statement and menu_choice #4

elif menu_choice == 4, break out of the loop.

```
191
192     # Stop the loop
193     elif menu_choice == "4":
194         break
195
```

Print to show program has ended

This print statement let the user know that the program has ended.

```
195
196 print("Program Ended")
197
```

Summery

In this assignment our constants were predefined, and our variable names were given but not assigned. We read a json file into a list in our program, allowing us to build onto an existing file, we also read stored data back into our json file to be stored. Option 1 allowed use to receive user input and collect data that we used to append our list. Option 2 read the students list back to us, ensuring that data was collected. Option 3 allowed us to save data that was collected and option 4 allowed us to save and close the program. We learned how to load and dump json files and exception handling using the try, except, and finally statements. We did all this while using classes and functions/methods. By utilizing functions, we have much cleaner code that is easier to read, more modular, better for testing, and is more efficient .

References

Common Header Format in Python | Delft Stack
Reading and Writing to text files in Python - GeeksforGeeks
Python if-else Statement (tutorialspoint.com)
Writing Professional Papers (youtube.com)
What Is JSON | Explained (external site)
Exceptions in Python - Python Tutorial (external site)
GitHub Tutorial - Beginner's Training Guide (external site)
Python Exceptions: An Introduction (external site)
Functions - Learn Python - Free Interactive Python Tutorial
Let's Learn Python - Basics #6 of 8 - Functions - YouTube