**Name:** Woody Smith

**Date:** 7/27/2024

**Course:** Python Fundamentals

**Assignment:** Assignment05 – This assignment involves creating a Python script that

demonstrates how to use lists, dictionaries, and Error Handing inputs and files to manage data. The primary goal of the script is to create a course registration program that allows users to register students for courses, display current registration data, and save the data to a file. This program features a menu-driven interface that allows users to easily register

students, view current registrations, and save data. It effectively manages student

registration while demonstrates using dictionaries, files, and exception handling

## Header Comments

## 1. \*\*`#!/usr/bin/python`\*\*: This shebang line indicates that the script should be run using the Python interpreter. 2. \*\*`# -\*- coding: utf-8 -\*-`\*\*: This line specifies the encoding used in the script, which is UTF-8. 3. \*\*Title: Assignment05\*\*: Title of the script. 4. \*\*Desc: This assignment demonstrates using lists and files to work with data\*\*: A brief description of the script. 5. \*\*Change Log\*\*: - \*\*Wsmith,7/27/2024,Created Script\*\*: Initial creation of the script by Wsmith.

## Code Description

### import json

This imports the JSON module, which allows for reading and writing JSON data.

### # Define the Data Constants

### MENU: str = '''

Defines a constant MENU, which holds a string representing the main menu of the program.

### ---- 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"

Defines a constant FILE\_NAME to store the name of the file where data will be saved.

### # Define the Data Variables and constants

### student\_first\_name: str = ''

Holds the first name of a student entered by the user.

### student\_last\_name: str = ''

Holds the last name of a student entered by the user.

### course\_name: str = ''

Holds the name of a course entered by the user.

### student\_data: dict = {}

A dictionary to hold one row of student data.

### students: list = []

A list to hold multiple rows of student data.

### json\_data: str = ''

A string to hold combined data in JSON format.

### file = None

A variable to hold a reference to an opened file.

### menu\_choice: str

A variable to hold the menu choice made by the user.

### # Populate with test data

### students = [{'student\_first\_name': 'Woody', 'student\_last\_name': 'Smith', 'course\_name': 'Python 101'}]

Populates the students list with test data.

### try:

### with open(FILE\_NAME, 'w') as file:

### json.dump({'students': students}, file)

Creates and writes test data to a file in JSON format.

### except IOError as err:

### print(f'An I/O error occurred: {err}')

Handles any I/O error that occurs during file operations.

### else:

### print(f'Write to file {FILE\_NAME} successful')

Executes if the try block does not raise an exception.

### finally:

### print(json.dumps({'students': students}, indent=4))

### The json.dumps method converts our students dictionary into a JSON formatted string. The indent=4 argument ensures that the JSON string is formatted with an indentation of four spaces, making it easier to read.

### # When the program starts, read the file data into a list of lists (table)

### try:

### with open(FILE\_NAME, 'r') as file:

### data = json.load(file)

Reads data from the file into the students list.

### students = data.get('students', [])

### except FileNotFoundError:

### print(f'The file {FILE\_NAME} does not exist.')

Handles the case where the file does not exist.

### except Exception as e:

### print(f'An error occurred: {e}')

Handles any other exception that may occur.

### else:

### print(f'Read file successful')

Executes if the try block does not raise an exception.

### # Present and Process the data

### while True:

### # Present the menu of choices

### print(MENU)

Displays the main menu.

### menu\_choice = input('What would you like to do: ')

Gets the user's menu choice.

### # Input user data

### if menu\_choice == '1':

Checks if the user chose to register a student for a course.

### student\_first\_name = input("Enter the student's first name: ")

Gets the student's first name.

### if not student\_first\_name.isalpha():

### raise ValueError('Student First Name needs to be characters only')

Validates that the first name contains only alphabetic characters.

### student\_last\_name = input("Enter the student's last name: ")

Gets the student's last name.

### if not student\_last\_name.isalpha():

### raise ValueError('Student Last Name needs to be characters only')

Validates that the last name contains only alphabetic characters.

### course\_name = input('Please enter the name of the course: ')

Gets the course name.

### if len(course\_name) < 5 or course\_name.isnumeric():

### raise ValueError('Course name length or value is not correct. Please try again')

Validates the course name length and that it is not numeric.

### student\_data = {'student\_first\_name': student\_first\_name,

Creates a dictionary with the student data.

### 'student\_last\_name': student\_last\_name,

### 'course\_name': course\_name}

### students.append(student\_data)

Adds the student data to the students list.

### print(f'You have registered {student\_first\_name} {student\_last\_name} for {course\_name}.')

Confirms the registration to the user.

### elif menu\_choice == '2':

Checks if the user chose to show current data.

### print(json.dumps({'students': students}, indent=4))

Displays the current student data in a formatted JSON string by implementing the. dump() method converts our students dictionary into a JSON formatted string. The indent=4 argument ensures that the JSON string is formatted with an indentation of four spaces, making it easier to read.

### elif menu\_choice == '3':

Checks if the user chose to save data to a JSON file.

### try:

### with open(FILE\_NAME, 'w') as file:

### json.dump({'students': students}, file)

Writes the student data to the file in JSON format using the .dump() method.

### except IOError as err:

### print(f'An I/O error occurred: {err}')

Handles any I/O error that occurs during file operations.

### else:

### print('The following data was saved to file!')

### for student in students:

### print(f"Student {student['student\_first\_name']} {student['student\_last\_name']} is enrolled in {student['course\_name']}")

Confirms the data was saved and displays it.

### elif menu\_choice == '4':

Checks if the user chose to exit the program.

### break

Exits the loop.

### else:

### print('Please only choose option 1, 2, or 3')

Prompts the user to choose a valid option.

### print('Program Ended')

Indicates the program has ended.

**Challenges:** I didn't encounter any difficult challenges with this assignment, as we are now delving into some of the more advanced features of the language. Coming from a background in C++/C# and SQL programming as a Data Engineer, I can freely experiment with the code without getting ahead of myself. Although my documentation process might seem tedious and boring, it aligns with my professional experience, where I routinely document and store code in Confluence and Bitbucket.

**Summary:** Assignment05 involves creating a Python script to manage student course registrations using lists, dictionaries, and error handling. The program features a menu-driven interface for registering students, displaying current data, and saving data to a file, demonstrating effective data management and exception handling.