Nathan Castek

February 27th, 2024

IT FDN 110 A

Assignment07

https://github.com/trinathlon/IntroToProg-Python-Mod07

Classes and Objects

Introduction

In this assignment, I reviewed various articles and videos detailing the usage of classes and objects. Then, I created a simple python script that prompts a user to make a selection from a menu of options. Depending on the selection, the program will execute the appropriate command. In this assignment, I utilized: constants, variables, conditional operators, the input function, the print function, type hints, while loops, lists, dictionaries, error handling, functions, classes, objects and worked with JSON files. Furthermore, I practiced testing my code from the integrated development environment PyCharm and from the terminal.

Research

I started out the assignment by reading the various articles and watching the videos for this week's module and assignment. I learned about classes, objects, error handling, working with JSON files, and hot to utilize GitHub for configuration management.

Programming

In the next part of the assignment I created a simple python program that prompts a user to make a selection from a menu of options and then performs a specific action based on the selection the user made.

Header

I started out my program by revising the header file provided in the assignment starter file. I removed the professors name and replaced it with my name and date. The resultant header can be seen below.

```
# ------------------------------------------------------------------------------
-------------- #
# Title: Assignment07
# Desc: This assignment demonstrates using data classes
# with structured error handling
# Change Log: (Who, When, What)
#   NCastek,2/26/2024,Created Script
# ------------------------------------------------------------------------------
-------------- #
```

Defining Imports, Constants, and Variables

I then defined the imports, constants, and variables for the program. In this assignment I knew I had to read and write to a JSON file. Therefore, I needed to import json in order to use the json functions. Next, I defined the constants and variables. The constants and variables were detailed in the assignment, so I ensured the names were identical to the assignment instructions. The resultant code can be seen below.

```
# 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"
```

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

Define Classes

After revising the header, defining the constants, and variables, I set up the code to define the Person and Student class defined in the assignement instructions. I created each class, made a docstring describing the class, and then used the “pass” keyword as a placeholder.

Define Functions

After creating placeholder for the two classes, I began creating functions under each class. I was able to create functions by leveraging code from the starter file and from the module notes. The first function I defined was the constructor method for the Person class. The resultant function can be seen below:

```
def __init__(self, student_first_name: str = "", student_last_name: str =
""):
    """
    constructor of a person object with a first name and last name set to
empty strings

    ChangeLog: (Who, When, What)
    NCastek,2/26/2024,Created function
    """
    self.student_first_name = student_first_name
    self.student_last_name = student_last_name
```

Next I created getters and setters for the student_first_name and student_last_name variables. I also validated the inputs of each of these in the setter functions. The resultant code can be seen below:

```
@property
def student_first_name(self):
    """
    returns the students first name

    ChangeLog: (Who, When, What)
    NCastek,2/26/2024,Created function
    """
    return self._student_first_name.title()

@student_first_name.setter
def student_first_name(self, value: str):
    """
    sets the students first name and checks to see if first name is valid

    ChangeLog: (Who, When, What)
    NCastek,2/26/2024,Created function
    """
    if value.isalpha() or value == "":  # allow characters or the default
empty string
        self._student_first_name = value
    else:
        raise ValueError("The first name should not contain numbers.")

@property
def student_last_name(self):
    """
    returns the students last name
```

```
    ChangeLog: (Who, When, What)
    NCastek,2/26/2024,Created function
    """
    return self._student_last_name.title()

@student_last_name.setter
def student_last_name(self, value: str):
    """
    sets the students last name and checks to see if the last name is valid

    ChangeLog: (Who, When, What)
    NCastek,2/26/2024,Created function
    """
    if value.isalpha() or value == "":  # allow characters or the default
empty string
        self._student_last_name = value
    else:
        raise ValueError("The last name should not contain numbers.")
```

Lastly, I revised the string function to return a formatted string of the person object. The resultant code can be seen below:

```
def __str__(self):
    """
    The string function for the person class
    returns formatted string of the person

    ChangeLog: (Who, When, What)
    NCastek,2/26/2024,Created function
    """
    return f"{self.student_first_name},{self.student_last_name}"
```

I then created the Student class. From the assignment instructions I knew I wanted to inherit the Person class, so Person was used as an input to the Student class. Furthermore, I created a constructor for the Student class, by calling the person constructor and adding the input course name. The resultant code can be seen below:

```
class Student(Person):
    """
    A collection of student functions, inheriting the person class, and
adding course name.

    ChangeLog: (Who, When, What)
    NCastek,2/26/2024,Created Class
    """
    def __init__(self, student_first_name: str = "", student_last_name: str =
"", course_name: str = ""):
        """
        constructor of a student object with first name, last name, and
course name set to empty strings
```

```
        ChangeLog: (Who, When, What)
        NCastek,2/26/2024,Created function
        """
        super().__init__(student_first_name, student_last_name)
        self.course_name = course_name
```

Then, I created a getter and setter for course name and revised the string function to return a formatted string of the student object. The resultant code can be seen below.

```
@property
def course_name(self):
    """
    returns the students course name

    ChangeLog: (Who, When, What)
    NCastek,2/26/2024,Created function
    """
    return self._course_name

@course_name.setter
def course_name(self, value: str):
    """
    sets the students course name and checks to see if it is valid

    ChangeLog: (Who, When, What)
    NCastek,2/26/2024,Created function
    """
    if value.isalnum() or value == "":  # allow characters or the default
empty string
        self._course_name = value
    else:
        raise ValueError("The course name should be alphanumeric
characters.")

def __str__(self):
    """
    The string function for the student class
    returns formatted string of the student

    ChangeLog: (Who, When, What)
    NCastek,2/26/2024,Created function
    """
    return
f"{self.student_first_name},{self.student_last_name},{self.course_name}"
```

The FileProcessor and IO classes already existed in the starter file, so for those I just had to make some revisions to use student objects instead of a list of dictionaries. I did this by changing all instances where student_data dictionaries were used to student objects.

Read JSON File

The read_data_from_file function needed to be revised to use student objects. I did this by storing the results of json.load(file) into the variable file_data. Then, I looped through the file data to generate student objects for each line in the file data. The resultant code can be seen below:

```
@staticmethod
def read_data_from_file(file_name: str, student_data: list[Student]):
    """ This function reads data from a json file and loads it into a list of
student object rows

    ChangeLog: (Who, When, What)
    NCastek,2/26/2024,Created function

    :param file_name: string data with name of file to read from
    :param student_data: list[Student] student objects to be filled with file
data

    :return: list[Student]
    """
    file_data = []
    file = None
    try:
        file = open(file_name, "r")
        file_data = json.load(file)
        file.close()
    except Exception as e:
        IO.output_error_messages(message="Error: There was a problem with
reading the file.", error=e)

    finally:
        if file is not None and file.closed is False:
            file.close()
    for line in file_data:
        student_data.append(
            Student(line["student_first_name"], line["student_last_name"],
line["course_name"])
        )

    return student_data
```

Writing to JSON File

The write_data_to_file function needed to be revised to take the student objects, convert them back to dictionaries, so they could be written to the JSON file. I did this by looping through the student objects and converting them to dictionaries and then appending them to the file. The resultant code can be seen below:

```
@staticmethod
def write_data_to_file(file_name: str, student_data: list[Student]):
    """ This function writes data to a json file with data from a list of
student object rows
```

```
    ChangeLog: (Who, When, What)
    NCastek,2/26/2024,Created function

    :param file_name: string data with name of file to write to
    :param student_data: list[Student] of student object rows to be writen to
the file

    :return: None
    """

    file_data = []
    for student in student_data:
        file_data.append({"student_first_name": student.student_first_name,
                          "student_last_name": student.student_last_name,
                           "course_name": student.course_name})
    file = None

    try:
        file = open(file_name, "w")
        json.dump(file_data, file)
        file.close()
        IO.output_student_and_course_names(student_data=student_data)
    except Exception as e:
        message = "Error: There was a problem with writing to the file.\n"
        message += "Please check that the file is not open by another
program."
        IO.output_error_messages(message=message, error=e)
    finally:
        if file is not None and not file.closed:
            file.close()
```

Input/Output Data

Next, I had to revise the output_student_and_course_names function to utilize the student objects. The resultant code can be seen below:

```
@staticmethod
def output_student_and_course_names(student_data: list[Student]):
    """ This function displays the student and course names to the user

    ChangeLog: (Who, When, What)
    NCastek,2/26/2024,Created function

    :param student_data: list[Student] of student object rows to be displayed

    :return: None
    """

    print("-" * 50)
    for student in student_data:
        print(f'Student {student.student_first_name} '
              f'{student.student_last_name} is enrolled in
```

```
{student.course_name}')
    print("-" * 50)
```

Next, I revised the input_student_data function to use student objects. The resultant code can be seen below:

```
@staticmethod
def input_student_data(student_data: list[Student]):
    """ This function gets the student's first name and last name, with a
course name from the user

    ChangeLog: (Who, When, What)
    NCastek,2/26/2024,Created function

    :param student_data: list[Student] of student object rows to be filled
with input data

    :return: list
    """

    try:
        student_first_name = input("Enter the student's first name: ")
        student_last_name = input("Enter the student's last name: ")
        course_name = input("Please enter the name of the course: ")
        student = Student(student_first_name, student_last_name, course_name)
        student_data.append(student)
        print()
        print(f"You have registered {student_first_name} {student_last_name}
for {course_name}.")
    except ValueError as e:
        IO.output_error_messages(message="One of the values was the correct
type of data!", error=e)
    except Exception as e:
        IO.output_error_messages(message="Error: There was a problem with
your entered data.", error=e)
    return student_data
```

Testing

After completing my program I tested it by running it from PyCharn and the terminal. I first ran my program from PyCharm and below you can see my user input's and what was printed to the screen.

Note: I first made the entry:

```
[{"student_first_name": "Bob", "student_last_name": "Smith", "course_name":
"Python 100"}, {"student_first_name": "Sue", "student_last_name": "Jones",
"course_name": "Python 100"}, {"student_first_name": "Dan",
"student_last_name": "Lewis", "course_name": "Python 101"}]
```

in the Enrollments.json file so the program could read in the file. Then, I tested the various inputs, see below for the terminal I/O.

```
---- 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.
----------------------------------------


Enter your menu choice number: 1
Enter the student's first name: Hugo
Enter the student's last name: Lop
Please enter the name of the course: Python 203

You have registered Hugo Lop for Python 203.


---- 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.
----------------------------------------


Enter your menu choice number: 2
------------------------------------------------
Student Bob Smith is enrolled in Python 100
Student Sue Jones is enrolled in Python 100
Student Dan Lewis is enrolled in Python 101
Student Hugo Lop is enrolled in Python 203
------------------------------------------------


---- 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.
----------------------------------------
```

```
Enter your menu choice number: 3
-------------------------------------------------
Student Bob Smith is enrolled in Python 100
Student Sue Jones is enrolled in Python 100
Student Dan Lewis is enrolled in Python 101
Student Hugo Lop is enrolled in Python 203
-------------------------------------------------


---- 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.
----------------------------------------


Enter your menu choice number: 1
Enter the student's first name: Polly
Enter the student's last name: Pockets
Please enter the name of the course: Python 101

You have registered Polly Pockets for Python 101.


---- 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.
----------------------------------------


Enter your menu choice number: 2
-------------------------------------------------
Student Bob Smith is enrolled in Python 100
Student Sue Jones is enrolled in Python 100
Student Dan Lewis is enrolled in Python 101
Student Hugo Lop is enrolled in Python 203
Student Polly Pockets is enrolled in Python 101
-------------------------------------------------
```

```
---- 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.
----------------------------------------


Enter your menu choice number: 3
-------------------------------------------------
Student Bob Smith is enrolled in Python 100
Student Sue Jones is enrolled in Python 100
Student Dan Lewis is enrolled in Python 101
Student Hugo Lop is enrolled in Python 203
Student Polly Pockets is enrolled in Python 101
-------------------------------------------------


---- 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.
----------------------------------------


Enter your menu choice number: 1
Enter the student's first name: 7
Enter the student's last name: 3
Please enter the name of the course: python 100
One of the values was the correct type of data!

-- Technical Error Message --
The first name should not contain numbers.
Inappropriate argument value (of correct type).
<class 'ValueError'>


---- 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.

Next, I tested my program in the terminal and saw identical results to when I used PyCharm.

Summary

In this assignment, I reviewed various articles and videos detailing the usage of classes and objects. Then, I created a simple python script that prompts a user to make a selection from a menu of options. Depending on the selection, the program will execute the appropriate command. In this assignment, I utilized: constants, variables, conditional operators, the input function, the print function, type hints, while loops, lists, dictionaries, error handling, functions, classes, objects and worked with JSON files. Furthermore, I practiced testing my code from the integrated development environment PyCharm and from the terminal.