Andrew Yarberry

11/17/2024

ITD FDN 110B Foundations of Programming: Python

Module 06

https://github.com/Finviel314/Python110

Assignment 06 Classes and Functions

Introduction

For assignment six, we are shifting to programming by creating custom classes and functions. This requires reordering how we think about processing our programs. The core function will remain unchanged, but rather than writing the code linearly, we will create a class with functions to read and write to the file and a separate class to cover inputs/outputs to the console.

Functions

A function is a script that only runs when it is called by the main body of the program. Previously were writing our code so that it would execute line by line. With a function, you can write a block of code that can be reused whenever it is called. "You can pass data, known as parameters, into a function" (1) and it will return a result.

Example

```
def my_function():
  print("Hello from a function")

my_function()
```

ex

Figure 1: Function Example (1)

To further streamline using functions we must familiarize ourselves with global and local variables. A local variable is declared inside a function where "global variables can be used by everyone, both inside of functions and outside" (2). It is worth noting that if you use the same name of a global variable as a local variable the local variable will only exist inside the function and the global variable will be unchanged. (Figure 2)

```python
x = "awesome"

def myfunc():
  x = "fantastic"
  print("Python is " + x)

myfunc()

print("Python is " + x)
```

```
Python is fantastic
Python is awesome
```

Figure 2: Global Local Variable Example (2)

While compiling my code I ran into a PyCharm warning about 'shadow variables' which "occurs when a variable defined in the inner scope has the same name as a variable in the outer scope" (3). The program was still able to compile but I suspect shadow variables could reduce the robustness of the code. The solution was to properly implement parameters for the functions.

In Figure 3 we can see we are calling the 'input_student_data' function and inside the parenthesis, we are passing the parameter that 'students is equal to 'student_data' where students is the parameter for the function shown in Figure 4.

```python
if menu_choice == '1':
    IO.input_student_data(student_data=students)
    print('\nHere is the current data:')
    IO.output_student_courses(student_data=students)
```

Figure 3: Passing to Local Variable

```python
@staticmethod  1 usage
def input_student_data(student_data: list):
    """Prompts the user to enter student information and adds it to the list...."""
```

Figure 4: Function Parameter Example

The use of arguments or parameters further increases the usefulness of the functions we create because we are not limited to global variables but can instead have local variables that can pass different information each time the function is called.

Class

A Class is a type of object in Python that can be used like a “constructor, or a "blueprint" for creating objects” (4). “Methods in objects are functions that belong to the object” (5). An example of a object method can be seen in Figure

Example

Insert a function that prints a greeting, and execute it on the p1 object:

```python
class Person:
  def __init__(self, name, age):
    self.name = name
    self.age = age

  def myfunc(self):
    print("Hello my name is " + self.name)

p1 = Person("John", 36)
p1.myfunc()
```

Figure 5: Object Method Example (5)

For this program, we are using classes to group our functions into two distinct groups for handling processes. The first is anything that involves reading and writing to the file, which is called ‘FileProcessor’, and the second handles inputs and outputs to the console which is called ‘IO’.

```python
# File Processing Class
class FileProcessor:  2 usages
    """Handles file operations for reading and writing student data to a JSON file...."""

    @staticmethod  1 usage
    def read_data_from_file(file_name: str, student_data=None):...

    @staticmethod  1 usage
    def write_data_to_file(file_name: str, student_data: list):...
# End of Class Definition
```

Figure 6: File Processor Class

```python
# Input/Output Class
class IO:  11 usages
    """
    A class for input/output operations in the course registration program.
    """
    @staticmethod  6 usages
    def output_error_messages(message: str, error: Exception = None):...

    @staticmethod  2 usages
    def output_student_courses(student_data: list):...

    @staticmethod  1 usage
    def output_menu():...

    @staticmethod  1 usage
    def input_menu_choice():...

    @staticmethod  1 usage
    def input_student_data(student_data: list):...

#End of class definition
```

Figure 7: Input-Output Class

JSON

Java-Script object notation (JSON) "is a syntax for storing and exchanging data" (6) and Python has a built-in package for working with this file type. Because of this built-in functionality code to read and write to JSON files is simpler than code we have seen for working with CSV files as shown in figures 8 and 9 respectively.

```python
with open(file_name, 'r') as file:
    return json.load(file)
```

Figure 8: JSON Read Example

```python
with open(file_name, 'w') as file:
    json.dump(student_data, file)
```

Figure 9: JSON Write Example

Testing the Program

I was able to successfully open and run the program in both PyCharm and CMD. Error handling works can be seen in Figures 10, 11, and 12 respectively.

```
---- 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 students first name: Vic
Enter the students last name: Vu
Enter the course name: Python 100

Here is the current data:
--------------------------------------------------
Student Andrew Yarberry is enrolled in Python 100
Student Vic Vu is enrolled in Python 100
--------------------------------------------------
```

Figure 10: Selection 1 Output PyCharm

```
---- 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 Andrew Yarberry is enrolled in Python 100
Student Vic Vu is enrolled in Python 100
--------------------------------------------------
```

Figure 11: Selection 2 Output PyCharm

```
---- 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
The following data was saved to file.
Student Andrew Yarberry is enrolled in Python 100
Student Vic Vu is enrolled in Python 100
```

Figure 12: Selection 3 Output PyCharm

The program needs to have structured error handling when taking user inputs for first and last names, and an example can be seen in Figure 13.

```
---- 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 students first name: 3
Name cannot contain numbers.
Enter the students first name:
```

Figure 13: Selection 1 Non-Alpha Name Error Handling

The program also need to have structured error handling for reading and writing data to the file. Figure 14 shows how the program will respond if a file is not present in the directory. Where Figure 15 shows and example if there is no data in the file to read.

```
File does not exist, please create a file called Enrollments.json.

-- Technical Error Message --
[Errno 2] No such file or directory: 'Enrollments.json'
File not found.
<class 'FileNotFoundError'>
```

Figure 14: File Not Found Error Handling

```
There was an non specific error.

-- Technical Error Message --
Expecting value: line 1 column 1 (char 0)
Subclass of ValueError with the following additional properties:

msg: The unformatted error message
doc: The JSON document being parsed
pos: The start index of doc where parsing failed
lineno: The line corresponding to pos
colno: The column corresponding to pos


<class 'json.decoder.JSONDecodeError'>
```

Figure 15: Non-Specific Error

```
Microsoft Windows [Version 10.0.26100.2314]
(c) Microsoft Corporation. All rights reserved.

C:\Users\andre>cd C:\Users\andre\Documents\Fundamentals of Python\_Module06\Assignment

C:\Users\andre\Documents\Fundamentals of Python\_Module06\Assignment>python assignment06.py

---- 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 students first name: Andrew
Enter the students last name: Yarberry
Enter the course name: Python 100

Here is the current data:
--------------------------------------------------
Student Andrew Yarberry is enrolled in Python 100
Student Vic Vu is enrolled in Python 100
Student Andrew Yarberry is enrolled in Python 100
--------------------------------------------------

---- 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 Andrew Yarberry is enrolled in Python 100
Student Vic Vu is enrolled in Python 100
Student Andrew Yarberry is enrolled in Python 100
--------------------------------------------------

---- 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
The following data was saved to file.
Student Andrew Yarberry is enrolled in Python 100
Student Vic Vu is enrolled in Python 100
Student Andrew Yarberry is enrolled in Python 100

---- 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: 4
Goodbye!

C:\Users\andre\Documents\Fundamentals of Python\_Module06\Assignment>
```

Figure 16: CMD Output

Summary

Assignment six required us to reorganize our code with an object-oriented perspective by creating class objects to organize our functions. Then write the functions with local variables and utilize parameters to pass information into them and increase their function and reusability. The program needed to continue to have structured error handling when taking inputs from the user as well as when reading and writing the file.

This assignment required a shift in mentality when thinking about how to write our code. In our previous experiences in this class with simple scripts, we tend to write them linearly. Here the code is less governed by the order in which it is written since we can call functions anywhere in our code.

References

1. *W3 Schools, https://www.w3schools.com/python/python_functions.asp, 2024 (External Site)*
2. *W3 Schools, https://www.w3schools.com/python/python_variables_global.asp, 2024 (External Site)*
3. *Geeks for Geeks, https://www.geeksforgeeks.org/variable-shadowing-in-python, 2024 (External Site)*
4. *W3 Schools, https://www.w3schools.com/python/python_classes.asp, 2024 (External Site)*
5. *W3 Schools, https://www.w3schools.com/python/gloss_python_object_methods.asp, 2024 (External Site)*
6. *W3 Schools, https://www.w3schools.com/python/python_json.asp, 2024 (External Site)*
7. *Github, https://github.com/Finviel314/Python110, 2024 (External Site)*

Appendix

```
# ------------------------------------------------------------------------------
--------------- #
# Title: Assignment06
# Desc: This assignment demonstrates building classes and functions to handle
operations within out programs
# Change Log: (Who, When, What)
#  #  Andrew Yarberry, 11/16/2024, Initial Release
# Notes:
#  * Add this starting data to the Enrollments.json file as needed.
# [{'FirstName': 'Vic', 'LastName': 'Vu', 'CourseName' : 'Python 100'
# ------------------------------------------------------------------------------
--------------- #

import json

# Global Variables
FILE_NAME: str = 'Enrollments.json' # File name for storing student data
MENU = '''
---- 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
----------------------------------------
'''

#Main Data Structure
students: list = []  # A list to store student information
menu_choice: str  # Holds the choice made by the user.

# File Processing Class
class FileProcessor:
    """
    Handles file operations for reading and writing student data to a JSON
file.

    Provides methods to:
    - Read student data from a specified JSON file.
    - Write student data to a specified JSON file.
    """
```

```
    @staticmethod
    def read_data_from_file(file_name: str, student_data=None):
        """Reads student data from a JSON file.

        Args:
            file_name: The name of the JSON file.
            student_data: An optional list of student dictionaries. If
provided,
                it will be used as the default data if the file is not found.

        Returns:
            A list of student dictionaries, either read from the file or the
provided
            default data.
        """
        if student_data is None:
            student_data = students
        try:
            with open(file_name, 'r') as file:
                return json.load(file)
        except FileNotFoundError as e:
            IO.output_error_messages('File does not exist, please create a
file called Enrollments.json.', e)
        except Exception as e:
            IO.output_error_messages('There was an non specific error.', e)

    @staticmethod
    def write_data_to_file(file_name: str, student_data: list):
        """Writes student data to a JSON file.

        Args:
            file_name: The name of the JSON file.
            student_data: A list of student dictionaries.
        """
        try:
            with open(file_name, 'w') as file:
                json.dump(student_data, file)
            print('The following data was saved to file.')
            for student in student_data:
                print(f'Student {student['FirstName']} '
                      f'{student['LastName']} is enrolled in
{student['CourseName']}')
        # Move error handling to IO
        except FileNotFoundError as e:
            IO.output_error_messages('File does not exist, please create a
file called Enrollments.json.', e)
        except Exception as e:
            IO.output_error_messages('There was an non specific error.', e)
        finally:
            if not file.closed:
                file.close()
# End of Function Definition

# Input/Output Class
class IO:
    """
    A class for input/output operations in the course registration program.
```

```
    """
    @staticmethod
    def output_error_messages(message: str, error: Exception = None):
        """Prints an error message to the console.

        Args:
            message: The error message to display.
            error: An optional exception object to provide more detailed
information.
        """
        print(message, end='\n\n')
        if error is not None:
                print('-- Technical Error Message -- ')
                print(error, error.__doc__, type(error), sep='\n')

    @staticmethod
    def output_student_courses(student_data: list):
        """Prints a formatted list of students and their courses.

        Args:
            student_data: A list of student dictionaries.
        """
        print("-" * 50)
        for student in student_data:
            print(f'Student {student['FirstName']} '
                  f'{student['LastName']} is enrolled in
{student['CourseName']}')
        print('-' * 50)

    @staticmethod
    def output_menu():
        """Prints the main menu to the console."""
        print(MENU)

    @staticmethod
    def input_menu_choice():
        """Prompts the user to enter a menu choice and validates the input.

        Returns:
            The user's selected menu choice as a string.
        """
        # Define function block
        choice = '0'
        try:
            choice = input('Enter your menu choice number: ')
            if choice not in ('1', '2', '3', '4'):
                raise Exception('You must choose 1, 2, 3, or 4')
        except Exception as e:
            IO.output_error_messages(e.__str__())
        return choice

    @staticmethod
    def input_student_data(student_data: list):
        """Prompts the user to enter student information and adds it to the
list.

        Args:
```

```
            student_data: A list of student dictionaries.
        """
        try:
            while True:
                try:
                    student_first_name = input('Enter the students first
name: ')
                    if not student_first_name.isalpha():
                        raise ValueError('Name cannot contain numbers.')
                    break
                except ValueError as e:
                    print(e)
            while True:
                try:
                    student_last_name = input('Enter the students last name:
')
                    if not student_last_name.isalpha():
                        raise ValueError('Name cannot contain numbers.')
                    break
                except ValueError as e:
                    print(e)
            course_name = input('Enter the course name: ')
            student_data.append({'FirstName': student_first_name,
                                 'LastName': student_last_name,
                                 'CourseName': course_name})
        except Exception as e:
            IO.output_error_messages('There was a non-specific error when
adding data.', e)

#End of function definition

# Main Body of Script
students = FileProcessor.read_data_from_file(FILE_NAME,
student_data=students)

while True:
    IO.output_menu()
    menu_choice = IO.input_menu_choice()

    if menu_choice == '1':
        IO.input_student_data(student_data=students)
        print('\nHere is the current data:')
        IO.output_student_courses(student_data=students)

    elif menu_choice == '2':
        IO.output_student_courses(student_data=students)

    elif menu_choice == '3':
        FileProcessor.write_data_to_file(FILE_NAME, student_data=students)

    elif menu_choice == '4':
        print('Goodbye!')
        break  # out of the while loop
```

Figure 17: Full Python Code