Johnathan Luu
Intro to Python
Assignment 8
6/6/2022
https://github.com/LofiLogan/

Classes and Objects

Intro

In module 8, a more in depth dive into the understanding of classes and objects was done. Classes, defined as a way of grouping data and functions, allow for more organized writing. For example, as shown in Assignment 8, multiple classes were created to group functions so that certain functions and data can be referenced easily if changes were necessary. In addition, properties (getters and setters) were introduced in this module. This allows for the bundling of data with the methods that operate on them. 'Getter' is used to retrieve the data whereas 'setter' is used for changing the data.

Creating the Program

In the first section labeled 'data', the class 'Product' was created. This essentially defines and stores data about the product. The first method, the '__init__' method is used to initialize the object's state. In it, it defines that the 'Product' object is made of a 'product_name' and 'product_price'.

```
def __init__(self, product_name, product_price):

    try:
        self.__product_name = str(product_name)
        self.__product_price = float(product_price)
    except Exception as e:
        raise Exception("An error has occured setting initial values: "+ str(e))
```

This is followed by getter and setter properties of the class. The getter function returns the value of the attribute while the setter method sets/ updates the value.

```
@property
def product_name(self):
    return str(self.__product_name)

@product_name.setter
def product_name(self, value):
    if str(value).isnumeric():
        self.__product_name = value
    else:
        raise Exception("The product name cannot be numeric")

@property
def product_price(self):
    return float(self.__product_price)

@product_price.setter
def product_price(self, value):
    if str(value).isnumeric():
        self.__product_price = value
    else:
        raise Exception("The price must be a number")


def __str__(self):
    return self.product_name + "," + str(self.product_price)
```

The FileProcessor class will process data to and from a file and a list of product objects. This includes the function to save data to the file, and read data from the file.

```
@staticmethod
def save_data_to_file(file_name, list_of_product_objects):
    success_status = False
    try:
        file = open(file_name, "w")
        for product in list_of_product_objects:
            file.write(product.__str__() + "\n")
        file.close()
        success_status = True
    except Exception as e:
        print("There was an error!")
        print(e, e.__doc__, type(e), sep='\n')
    return success_status

@staticmethod
def read_data_from_file(file_name: str):

    lstProducts = []

    try:
        file = open(file_name, "r")
        for line in file:
            data = line.split(",")
            row = Product(data[0], data[1])
            lstProducts.append(row)
        file.close()
    except Exception as e:
        print("There was an error")
        print(e,e.__doc__, type(e), sep='\n')
    return lstProducts
```

The menu class follows, as reference to previous assignments to display options for the user to choose and input/retrieve data.

```
@staticmethod
def DisplayMenu():
    """  Display a menu of choices to the user
    """
    print('''
    Menu of Options
    1) Show current products
    2) Add a new item
    3) Save Data to File
    4) Exit Program
    ''')
    print()


# TODO: Add code to get user's choice
@staticmethod
def MenuChoice():
    option = str(input("Choose an option (1-4) ")).strip()
    print()
    return option


# TODO: Add code to show the current data from the file to user
def CurrentData(RowLst):
    print("The current products are: ")
    for row in RowLst:
        print(row.product_name + ", " + str(row.product_price))
        print()

# TODO: Add code to get product data from user
def AddData():
    try:
        item = ''
        product = str(input("Enter a product name: ").strip())
        price = float(input("Enter its price: ").strip())
        item = Product(product_name = product, product_price = price)
    except Exception as e:
        print(e)
    return item
```

And finally, the main body of the script that will load data from the file and previous classes to run the functions and process the inputs given.

```
try:
    lstofProducts = FileProcessor.read_data_from_file(strFileName)

    while True:
        # Show user a menu of options
        IO.DisplayMenu()
        # Get user's menu option choice
        Choice = IO.MenuChoice()
        if Choice.strip() == '1':
            # Show user current data in the list of product objects
            IO.CurrentData(lstofProducts)
            continue
        elif Choice.strip()=='2':
            # Let user add data to the list of product objects
            lstofProducts.append(IO.AddData())
            continue
        elif Choice.strip() == '3':
            # let user save current data to file and exit program
            FileProcessor.save_data_to_file(strFileName, lstofProducts)
            continue
        elif Choice.strip() =='4':
            break
except Exception as e:
    print("There was an error: ")
    print(e,e.__doc__, type(e), sep = '\n')
```

Results

```
Assignment08 — -bash — 80×52
Last login: Wed Jun  8 21:59:52 on ttys001
StudioRig1:Assignment08 coding$ python3 assignment08.py

        Menu of Options
        1) Show current products
        2) Add a new item
        3) Save Data to File
        4) Exit Program


Choose an option (1-4) 2

Enter a product name: chair
Enter its price: 5

        Menu of Options
        1) Show current products
        2) Add a new item
        3) Save Data to File
        4) Exit Program


Choose an option (1-4) 1

The current products are:
chair, 5.0


        Menu of Options
        1) Show current products
        2) Add a new item
        3) Save Data to File
        4) Exit Program


Choose an option (1-4) 3


        Menu of Options
        1) Show current products
        2) Add a new item
        3) Save Data to File
        4) Exit Program


Choose an option (1-4) 4

StudioRig1:Assignment08 coding$
```

```
products.txt   Open with TextEdit
chair,5.0
```

Conclusion

As a result, the ability to build classes allows for a more organized way of compartmentalizing data to be stored. By creating an object in the class, it allows for its properties to be established, called upon, and manipulated once the class is called upon. Properties of the class will include getters and setters, with getters being the method that calls upon the object and the setter which changes the data of the object. That is why a getter and setter is established for each attribute in the object. In this example, they would be the product name and product value.