Assignment-8&9
PROG8421-24-Sec2
Spring 2024
Due Date: Tuesday, July 23, 2024

Important Notes:

- This is a group assignment with maximum 4 students in a group
- Make sure you follow the specifications for each question.

Question-1. Create an object-oriented program that allows you to enter data for customers and employees.

Console

Customer/Employee Data Entry
Customer or employee? (c/e): c

DATA ENTRY
First name: Frank
Last name: Wilson
Email: frank44@gmail.com
Number: M10293

CUSTOMER
Name: Frank Wilson
Email: frank44@gmail.com
Number: M10293

Continue? (y/n): y
Customer or employee? (c/e): e

DATA ENTRY
First name: joel
Last name: murach
Email: joel@murach.com
SSN: 123-45-6789

EMPLOYEE
Name: Joel Murach
Email: joel@murach.com
SSN: 123-45-6789

Continue? (y/n): n

Bye!

Specifications

- Create a Person class that provides attributes for first name, last name, and email address. This class should provide a property or method that returns the person's full name.

- Create a Customer class that inherits the Person class. This class should add an attribute for a customer number.

- Create an Employee class that inherits the Person class. This class should add an attribute for a social security number (SSN).

- The program should create a Customer or Employee object from the data entered by the user, and it should use this object to display the data to the user. To do that, the program can use the **isinstance()** function to check whether an object is a Customer or Employee object.

Question-2. Create an object-oriented program that uses a custom list object to automatically generate and work with a series of random integers.

Console

```
Random Integer List
How many random integers should the list contain?: 12
Random Integers
===============
Integers: 17, 34, 34, 15, 71, 44, 97, 48, 19, 12, 83, 42
Count: 12
Total: 516
Average: 43.0

Continue? (y/n):
```

Specifications

- Create a RandomIntList class that inherits the list class. This class should allow a programmer to create a list of random integers from 1 to 100 by writing a single line of code. For example, a programmer should be able to create a custom list that stores 12 random integers with this line of code:
  int_list = RandomIntList(12)
- To do that, you can use the self-keyword to access the list superclass like this:
  self.append(rand_int)
- The RandomIntList class should contain methods or properties for getting the count, average, and total of the numbers in the list. In addition, it should contain a __str__method for displaying a comma-separated list of integers as shown above.
- The program should use the RandomIntList class to generate the list of random integers, display the list, and get the summary data (count, total, and average).
- The program should make sure the integer entered by the user is valid.

Question-3. Enhance the Product Viewer program so it provides one more type of product: a music album. When you enter the product number for a music album, it should print the data to the console like this:

Enter product number: 4

PRODUCT DATA

Name:	**Rubber Soul**
Artist:	**The Beatles**
Format:	**CD**
Discount price:	**10.00**

Open and test the program

1. In PyCharm, open the **objects.py** and **product_viewer.py** files provided to you in eConestoga.
2. Review the code and run the program to make sure it works correctly.

Note that the Movie class displays the format, which is DVD, as part of the name of the product. Improve the Movie and Book classes

3. In the Movie class, add an attribute named format that stores the format of the product. For example, the format could be DVD, streaming, and so on.
4. In the **product_viewer** module, modify the code that creates the Movie object so it stores “DVD” as the format attribute instead of appending this data to the end of the name attribute. Then, modify the code that displays the Movie object so it displays the format on a separate line, after the year of the movie.
5. Repeat steps 3 and 4 for the Book class. You can use “Hardcover” as the format for the book. Or, if you prefer, you can specify a different type of book format such as “Paperback” or “ebook”.

Add an Album class

6. In the objects module, add a class named Album that inherits the Product class. The Album class should add two attributes: one for storing the artist and another for storing the format.
7. In the **product_viewer.py** file, modify the code that creates the objects so it includes a fourth object, an Album object. This object should contain the data for a music album that you like. Then, add code that displays the Album object as shown at the beginning of this exercise.

Add a Media class

8. In the objects module, add a class named Media that inherits the Product class. This class should add a format attribute to the Product class.
9. Modify the Movie, Book, and Album classes so they inherit the Media class, not the Product class. This should create a class hierarchy that looks like this:

Product
  Media
    Movie
    Book
    Album

10. In the **product_viewer.py** file, modify the code that displays a product so it only displays the format attribute for Media objects. Note how this reduces code duplication.

Question-4. Work with a Book object that uses an Authors object to store one or more Author objects.

Open and test the program

1. In **PyCharm**, open the **objects.py** and **authors_tester.py** files provided in eConestoga
2. Review the code and note how the Book object uses an Authors object to store one or more Authors.
3. Run the code to see how it works. At this point, it doesn't display the book or author information correctly, but you'll fix that later in this exercise.

Improve the Author, Authors, and Book classes

4. In the Author class, add a __str__() method that returns the first and last name of the author, separated by a space.
5. In the Authors class, add a __str__() method that returns the name of each author, separating multiple authors with a comma and a space.
6. In the Book class, add a __str__() method that returns the title of the book, followed by the word "by" and one or more authors, with each author separated by a comma.
7. Run the **authors_tester** module again. This time, it should display the correct data for the book and author information. Make sure the program works correctly for a single author
8. In the **authors_tester** module, comment out the statement that adds the second author. Then, run the module again. This should work, but the last line says "Authors" where it should say "Author".
9. Modify the code so it uses the count property of the Authors object to display the correct label for the author or authors depending on the number of authors for the book.

Define and use an iterator for the Authors object

10. In the Authors class, add the __iter__() method that makes it possible to use a for statement to loop through all Author objects stored in the Authors object.
11. In the **authors_tester** module, add a for statement that loops through each author in the Authors object and prints each author to the console. When you're done, running this module should display this data to the console:

```
The Authors Tester program
BOOK DATA - SINGLE LINE
The Gilded Age by Mark Twain, Charles Warner

BOOK DATA - MUTLIPLE LINES
Title: The Gilded Age
Authors: Mark Twain, Charles Warner

AUTHORS
Mark Twain
Charles Warner
```