CP1890 – Assignment 2

Handed Out: 04 Mar 2024
Due Date: 19 Mar 2024

Objectives

This problem set will build on your development, Github usage, proper versioning and code contribution skills in a professional environment while building up on what you learnt in this semester with the introduction of inheritance, polymorphism and handling objects.

Collaboration

You will be working with your assigned group. Each group will designate a leader and the members will make commits to the leader's fork of the class by submitting a pull request and going through with the review process. You are free to divide the assignment objectives among your group members. Please note that each pull request has to be reviewed with the two other members of your group.

Once the assignment is complete, **the leader of the group** will raise a pull request to merge the code with the main class repository and will assign the **other group members as reviewers for that pull request.**

Submission

You will create one folder for your group under assignments with your group name. In this folder, you will have one python file for each question. You need to ensure that suitable comments are present in the code and the code has gone through the pull request and review process with your group. Once done, the leader should ensure the assignment link is posted to the drop box for the group that is provided on Brightspace **before the due date (no commits after the due date will be considered for grading)**.

Late submissions

Penalties for late submissions is as follows:

- Up to 24 hours after the due date: Flat 25% penalty
- Beyond 24 hours up to 48 hours: Flat 50% penalty
- Beyond 48 hours: No points for assignment

If for some reason you are unable to submit the assignment on time, please reach out to me to make alternate arrangements – which will be handled based on the merits of the case. **No extensions will be provided.** Please ensure you stay on top of the requirements and manage your time well.

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): y

Random Integers
===============
Integers:  52, 88, 10, 77, 56, 91, 17, 51, 22, 14, 48, 37
Count:     12
Total:     563
Average:   46.917

Continue? (y/n): n

Bye!
```

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

Create a class called Animal with the following attributes and methods:

- Attributes: name, species
- Methods: speak() - prints out the sound the animal makes

Next, create a class called Dog that inherits from the Animal class. Add an additional attribute called breed to the Dog class and override the speak() method to print out "Woof!".

Finally, create a class called Cat that also inherits from the Animal class. Add an additional attribute called color to the Cat class and override the speak() method to print out "Meow!".

Write a program that creates instances of both the Dog and Cat classes, and call the speak() method for each.

Specifications:

1. Animal class should have a constructor that initializes the name and species attributes.
2. Dog class should have a constructor that initializes the breed attribute and calls the superclass constructor.
3. Cat class should have a constructor that initializes the color attribute and calls the superclass constructor.

Sample Code:

```
dog = Dog("Max", "Dog", "Golden Retriever")
cat = Cat("Whiskers", "Cat", "Orange")

print("Dog:")
print("Name:", dog.name)
print("Species:", dog.species)
print("Breed:", dog.breed)
print("Sound:", end=" ")
dog.speak()

print("\n")

print("Cat:")
print("Name:", cat.name)
print("Species:", cat.species)
print("Color:", cat.color)
print("Sound:", end=" ")
cat.speak()
```

Sample Output:

Dog: Name: Max Species: Dog Breed: Golden Retriever Sound: Woof!	Cat: Name: Whiskers Species: Cat Color: Orange Sound: Meow!

Question 4:

Create a class called Event with the following attributes and methods:

- Attributes: name, location, start_date, end_date
- Methods: duration() - returns the duration of the event in days

Next, create a class called Conference that inherits from the Event class. Add an additional attribute called attendees to the Conference class and override the duration() method to calculate the duration in hours instead of days.

Finally, write a program that creates instances of both the Event and Conference classes, and call the duration() method for each.

Specifications:

1. Event class should have a constructor that initializes the name, location, start_date, and end_date attributes.
2. Conference class should have a constructor that initializes the attendees attribute and calls the superclass constructor.
3. The duration() method in the Event class should return the difference between start_date and end_date in days.
4. The duration() method in the Conference class should return the difference between start_date and end_date in hours.

Make use of the datetime module in Python to handle date and time operations.

Sample Code:

```
event = Event("Birthday Party", "New York", datetime(2023, 8, 25),
datetime(2023, 8, 26))
conference = Conference("Tech Conference", "San Francisco", datetime(2023, 9,
15), datetime(2023, 9, 17), 500)

print("Event:")
print("Name:", event.name)
print("Location:", event.location)
print("Start Date:", event.start_date)
print("End Date:", event.end_date)
print("Duration (days):", event.duration())

print("\n")

print("Conference:")
print("Name:", conference.name)
print("Location:", conference.location)
print("Start Date:", conference.start_date)
print("End Date:", conference.end_date)
print("Attendees:", conference.attendees)
print("Duration (hours):", conference.duration())
```

Sample Output:

Event: Name: Birthday Party Location: New York Start Date: 2023-08-25 00:00:00 End Date: 2023-08-26 00:00:00 Duration (days): 1	Conference: Name: Tech Conference Location: San Francisco Start Date: 2023-09-15 00:00:00 End Date: 2023-09-17 00:00:00 Attendees: 500 Duration (hours): 48

Question 5

Create a class called Task with the following attributes and methods:

- Attributes: task_name, task_description, due_date
- Methods: status() - returns the status of the task (completed or pending)

Next, create two classes - Homework and Meeting, both of which inherit from the Task class. Add additional attributes relevant to each class (e.g., subject for Homework, location for Meeting) and override the status() method to provide status specific to each type of task.

Finally, write a program that creates instances of both the Homework and Meeting classes, and call the status() method for each.

Specifications:

- Task class should have a constructor that initializes task_name, task_description, and due_date attributes.
- Homework class should have a constructor that initializes a subject attribute and calls the superclass constructor.
- Meeting class should have a constructor that initializes a location attribute and calls the superclass constructor.
- The status() method in the Task class should return "Pending" if the due_date is in the future, and "Completed" if the due_date is in the past.
- Override the status() method in the Homework class to return "Not started", "In progress", or "Completed" based on the task_status attribute.
- Override the status() method in the Meeting class to return "Scheduled" if the current date is before the due_date, and "Happened" if it is after the due_date.

Make use of the datetime module in Python to handle date and time operations.

Sample Code:

```
homework = Homework("Math Homework", "Complete exercises 1-5", datetime(2023, 10, 15), "Math")
meeting = Meeting("Team Meeting", "Discuss project updates", datetime(2023, 9, 20), "Office A")

print("Homework:")
print("Task Name:", homework.task_name)
print("Task Description:", homework.task_description)
print("Due Date:", homework.due_date)
print("Subject:", homework.subject)
print("Status:", homework.status())

print("\n")

print("Meeting:")
print("Task Name:", meeting.task_name)
print("Task Description:", meeting.task_description)
print("Due Date:", meeting.due_date)
print("Location:", meeting.location)
print("Status:", meeting.status())
```

Sample Output

Homework:
Task Name: Math Homework
Task Description: Complete exercises 1-5
Due Date: 2023-10-15 00:00:00
Subject: Math
Status: Not started

Meeting:
Task Name: Team Meeting
Task Description: Discuss project updates
Due Date: 2023-09-20 00:00:00
Location: Office A
Status: Scheduled