**Introduction:**

This lab focused on applying Object-Oriented Programming (OOP) principles in Python by working on two exercises. The first exercise involved creating a PersonalInfo class to store personal details such as name, address, age, and phone number. The second, more advanced exercise, involved creating a Pet class to represent pets, allowing users to input and store multiple pet objects. The objective was to reinforce my understanding of class structure, methods, and managing user input and data storage dynamically.

In the first part of the lab, I created the PersonalInfo class with attributes like name, address, age, and phone\_number. I used mutator (setter) methods to update the attributes and accessor (getter) methods to retrieve the data. This exercise was relatively easy as it was a straightforward implementation of OOP principles, focusing on encapsulating data and using methods to manage it. There were no major obstacles in this part. Creating and managing multiple instances of the PersonalInfo class went smoothly, and I was able to efficiently handle the user’s input.

The second exercise required creating a Pet class to store details about pets, such as their name, type, and age. I used a list to store multiple Pet objects and allowed users to input pet data dynamically. While the logic of creating and storing multiple objects was more advanced, ***I initially encountered a significant challenge in handling user input validation for displaying the list of pets.***

The problem arose when users were asked to choose between displaying all pets or filtering by type. There was no input validation for this choice, which led to errors if the user entered anything other than the expected "all" or "type". To address this, I incorporated a try-except block, which caught invalid inputs and prompted the user to enter the correct values. This solution made the program more robust and user-friendly, ensuring that even if the user made a mistake, they could easily correct it.

**Reflection:**

Combining both exercises, the lab covered a wide range of topics, from simple OOP concepts like encapsulation and methods to more complex issues like handling dynamic user input and lists of objects. The first exercise was relatively easy and reinforced my basic understanding of Python classes. The second exercise, however, was more challenging as it required handling multiple objects and implementing input validation, which I had to solve using the try-except method.

Overall, this lab was slightly advanced for me, particularly in the second exercise, but it was a valuable experience. It enhanced my ability to work with Python's OOP features, manage dynamic input, and deal with errors gracefully, which are critical skills in more complex programming scenarios.