UNIVERSITY OF CALOOCAN CITY
Caloocan, 1400 Metro Manila, Philippines

COLLEGE OF ENGINEERING
Computer Engineering
2nd Semester, School Year 2024-2025

Object-Oriented Programming

Laboratory Activity No. 1

Review of Technologies

Submitted by:
Villacin, Justine R.
Sat 12:00PM – 4:00PM/ 4:30PM – 8:30PM / BSCpE 1 - A

Submitted to
Engr. Maria Rizette H. Sayo
Instructor

Date Performed:
18-01-2025

Date Submitted
18-01-2025

I. **Objectives**

In this section, the goals in this laboratory are:

- To define the key terms in Object-oriented programming
- To be able to know the construction of OO concepts in relation to other types of programming such as procedural or functional programming

II. **Methods**

General Instruction:

A. Define and discuss the following Object-oriented programming concepts:

1. CLASSES

- An abstract logical entity that acts as a first prototype or template for creating objects is what a Python class is. Python classes provide methods for modifying an object's state and defining any attributes it may have.

Types of Classes in Python

- Python Abstract Class
- Python Concrete Class
- Python Partial Class

Python Abstract Class

A class that has one or more abstract methods is called an abstract class. A method that has a declaration, but no implementation is referred to as a "abstract method." It could be challenging to recall every class when working with a big codebase. A Python Abstract Class can be utilized in that situation. Python does not by default have an abstract class, in contrast to the majority of high-level languages.

Python Concrete class

While abstract classes might have both concrete and abstract methods, concreate classes only have concrete methods. Abstract methods are implemented by the concrete class, but they can also be started by the abstract base class using "super ()."

Python Partial Class

We can design functions that only apply portions of statements and keywords that we supply to it to generate new objects by using the partial Python class. The "Functools" module can be used to implement it.

2. OBJECTS

- The creation and manipulation of objects is the main focus of Python, an object-oriented programming language. Everything in the Python universe may be thought of as an object, and type(), which only contains variables and Python functions from Python itself, allows us to check its status. Lists, dictionaries, files, sets, strings, and other objects come in a variety of types—just think about integer variables! State Identity Behavior, which are physical things that specify how an object acts within itself and vice versa, is defined by an object's class.

Example of Python Classes and Objects

To further grasp the idea of Python classes and objects, let's look at an example. An object can be thought of as a typical, everyday item, such as a car. As was previously mentioned, we now know that a class contains its own data and functions, which may all be regarded as the object's features and actions, respectively. In other words, the car's features (data) include its color, price, number

of doors, and so on. The car's (object) activities (functions) include braking, speed, etc. As shown in the following illustration, a class can be used to construct several objects with various data and functionalities attached to them.

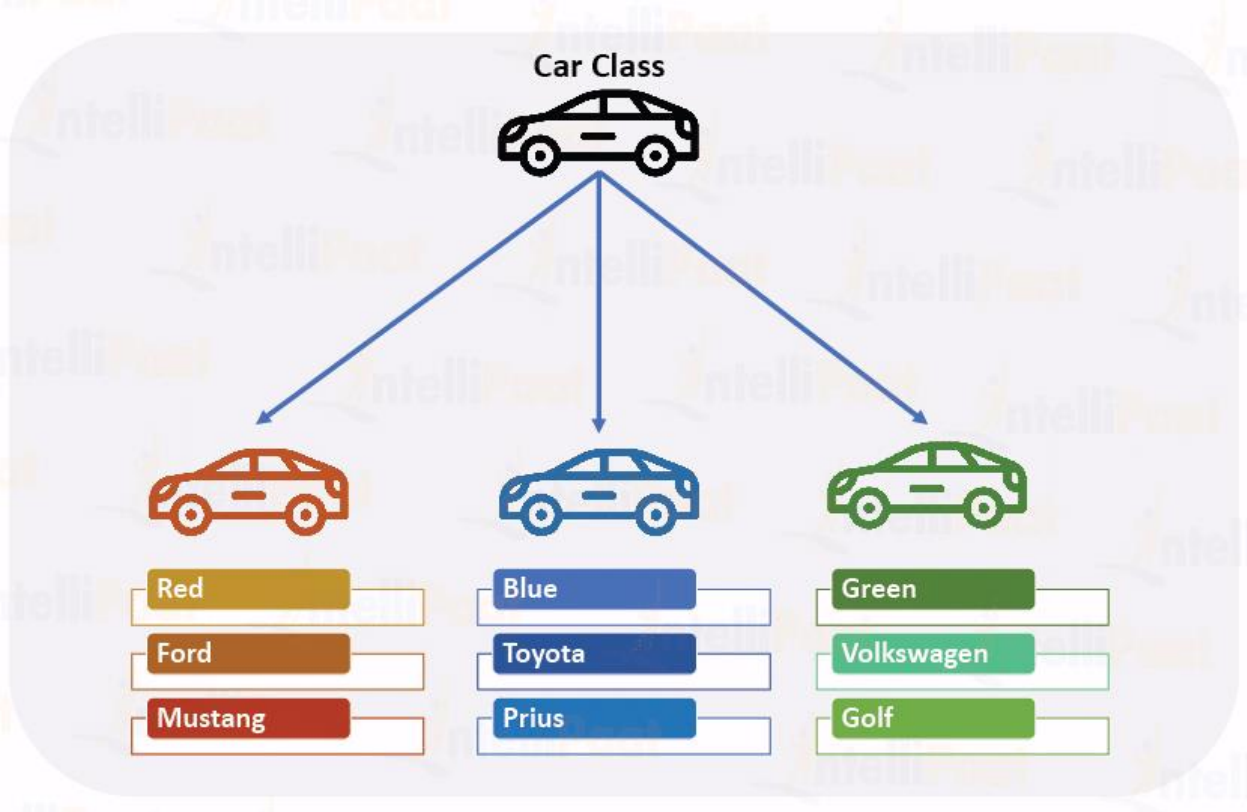

Figure 1. Python Classes and Object about Cars

Source: https://intellipaat.com/mediaFiles/2019/03/python10.png

3. FIELDS

- Fields are variables that are declared inside a class and serve as representations of the class's data attributes. They aid in capturing an object's attributes. For instance, fields in a Rectangle class may contain length and width, which hold the rectangle's measurements.

4. METHODS

- A method is a function that specifies an object's behavior and is connected to it. Methods can change the data of an object and are defined inside a class.

5. PROPERTIES

- An object's properties, sometimes referred to as its attributes, are its variables. They contain an object's data or status.

III. Results

A key idea of Python is object-oriented programming, which enables programmers to create scalable, modular, and maintainable applications. Programmers can fully utilize Python's OOP features to create sophisticated and effective solutions to challenging issues by comprehending the fundamental OOP concepts of classes, objects, inheritance, encapsulation, polymorphism, and abstraction.

OOPs are a type of code organization where real-world entities, and their behavior are represented by objects and classes. An object with attributes in an OOP is anything that has certain data and has the ability to use methods to carry out specific tasks.

OOPs Concepts in Python

- Class in Python
- Objects in Python
- Polymorphism in Python
- Encapsulation in Python
- Inheritance in Python
- Data Abstraction in Python

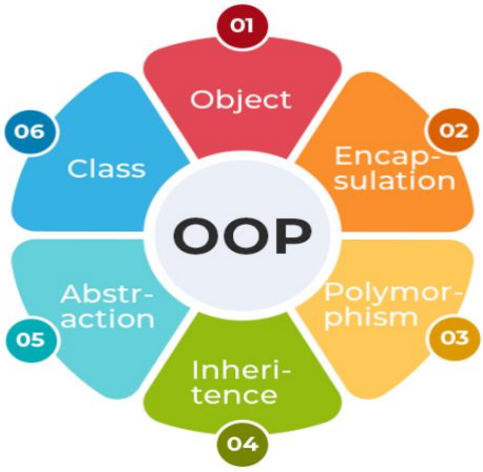

Figure 2. Python OOP's Concepts

Source: https://media.geeksforgeeks.org/wp-content/uploads/20230818181616/Types-of-OOPS-2.gif

Figure 3. Python Logo

Source: https://logodownload.org/wp-content/uploads/2019/10/python-logo.png

Python is integral to data science, a rapidly expanding field that leverages vast amounts of data for innovative solutions. The language's capabilities in deep learning and graphic generation further showcase its diverse applications. With an extensive library of resources available for learners at all levels, Python offers numerous opportunities for skill development and career advancement in technology. Overall, the article emphasizes that learning Python can be a strategic investment for anyone looking to enter or advance in the tech industry.

IV. **Conclusion**

In conclusion, the Object-Oriented Programming (OOP) lab exercise offers a thorough rundown of the core OOP ideas in Python, such as classes, objects, fields, methods, and properties. Students can appreciate how Python makes it easier to create scalable, modular, and maintainable systems by comprehending these essential components. According to my research, OOP enables programmers to efficiently model real-world elements, which helps them create intricate solutions for challenging issues.

Additionally, the use of OOP concepts—such as abstraction, polymorphism, inheritance, and encapsulation—improves code organization and facilitates improved data management. Anyone hoping to progress in their career in technology must grasp OOP concepts, especially as Python continues to gain prominence, especially in domains like data science and machine learning. In addition to giving students useful skills, this core knowledge gets them ready for upcoming software development problems.

Reference

Website

[1] Kislay. (2024c, November 25). Python classes and objects. Intellipaat. https://intellipaat.com/blog/tutorial/python-tutorial/python-classes-and-objects/

[2] Seoadmin. (2024b, December 30). OOPs concept in Python: Object-Oriented programming in Python for Data Analysis. Console Flare Blog. https://consoleflare.com/blog/object-oriented-programming-in-python/

[3] 3.1. Variables, Fields, and Parameters — Masters of Engineering Bridge Course. (n.d.). https://opendsa-server.cs.vt.edu/ODSA/Books/MengBridgeCourse/html/1114VariablesFieldsAndParameters.html

[4] P. Zandbergen and C. Cena, "Object-Oriented Programming: Objects, Classes & Methods Video with Lesson Transcript | Study.com," Study.com, 2019. https://study.com/academy/lesson/oop-object-oriented-programming-objects-classes-interfaces.html

[5] GeeksforGeeks. (2024, December 13). Python OOPs Concepts. GeeksforGeeks. https://www.geeksforgeeks.org/python-oops-concepts/

[6] Virginia. (n.d.). The Importance & Benefits of Learning Python | 10 Reasons & Advantages. https://www.idtech.com/blog/top-reasons-why-you-should-learn-python