**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:*
**Delinia, Filjohn B.**
**Saturday / BSCpE 1-A**

*Submitted to*
**Engr. Maria Rizette Sayo**
Instructor/Professor

*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

Object-oriented programming (OOP) is a programming paradigm that organizes software by grouping together features and activities into separate objects. This method enables the modeling of real-world entities, such as a person, complete with attributes like name, age, and address, as well as actions like walking and conversing. Similarly, an object can represent an email by including characteristics such as the recipient list and the subject, as well as operations such as sending and attaching files. OOP not only models' actual items like automobiles, but also connections between entities such as businesses and employees or students and educators. By expressing these real-world entities as software objects, OOP allows for data association and structured and modular data operations.

A class is simply a blueprint for building things by encapsulating data and the operations that operate on it. It defines characteristics (variables) and methods (functions) that describe the properties and behaviors of the objects generated by it. For example, a class may include a **__init__** method, which is a special method known as the constructor that, when an instance is created, initializes a new object with certain settings. The class defines the overall structure, although each object may have its own distinct data. A class's attributes represent an object's state or properties, whereas methods specify an object's actions or behaviors. Methods can change the attributes of an object or perform certain class-related actions. The self-keyword is used in methods to refer to the individual instance of the object on which the method is called, allowing each object to preserve its own state.

An object is an instance of a class. When a class is defined, no memory is allocated until an instance of that class is created. Each object has its own collection of data, defined by the class, and can perform actions using the class's methods. For example, a Car class object can hold specific characteristics such as the car's brand and model, as well as call methods to display or edit them. Python's use of classes and objects enables a modular, organized, and reusable code structure.

Instance methods in Python are functions defined within a class that can only be called on an instance of that class. They always take self as the first parameter, allowing access to the instance's attributes. For example, in a Dog class, methods like description() and speak() can return information or perform actions specific to each dog object. Additionally, the special **__str__()** method can be used to define a custom string representation of an object, improving how it is displayed when printed. Understanding instance methods and special methods like __str__() is key to effectively using object-oriented programming in Python.

Encapsulation, inheritance, abstraction, and polymorphism are the fundamental principles of object-oriented programming. Encapsulation is the process of bundling data and methods within a class and regulating access to them in order to protect data integrity and security. Inheritance allows subclasses to inherit characteristics and methods from parent classes, encouraging code reuse and minimizing duplication. Abstraction makes interaction with things easier by hiding complex aspects and exposing only necessary functionality. Polymorphism allows objects of different kinds to be considered as instances of the same base type, which promotes flexibility and code reuse. These principles work together to construct modular, manageable, and scalable software systems by emphasizing clarity, reusability, and simplicity of modification.

## III. Results

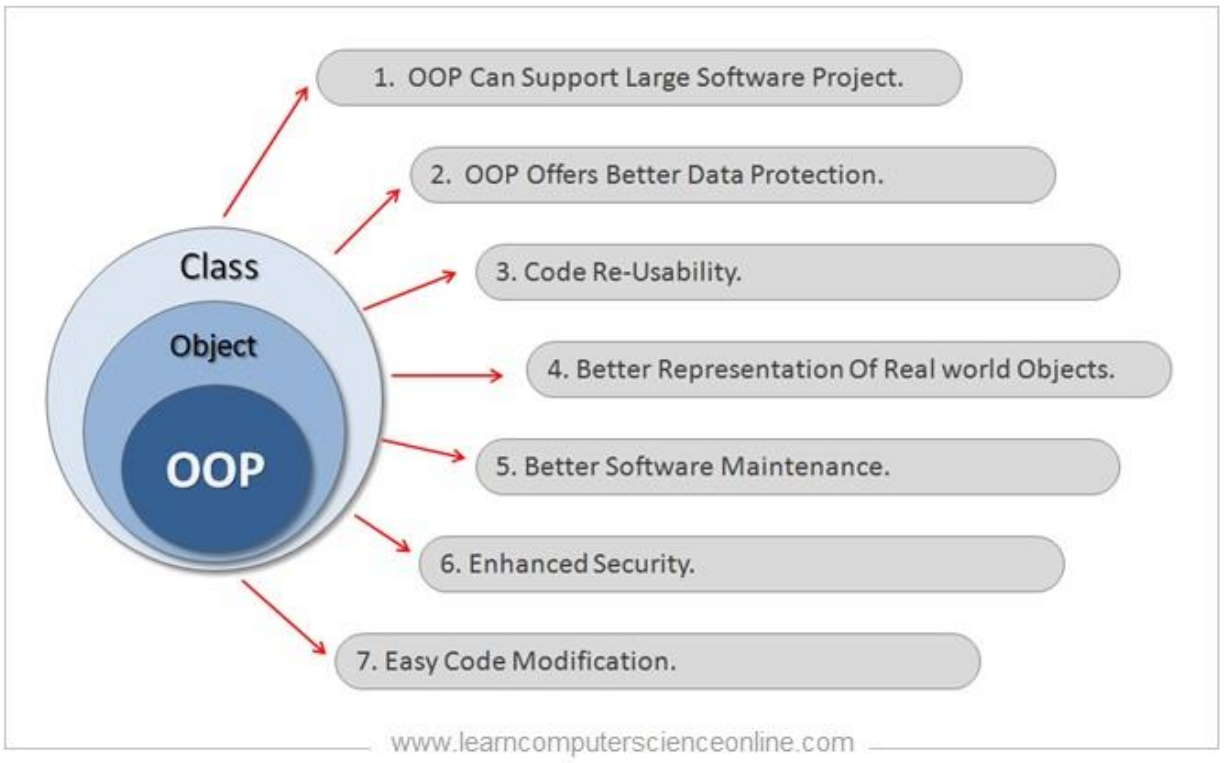
Class
Object
OOP
1. OOP Can Support Large Software Project.
2. OOP Offers Better Data Protection.
3. Code Re-Usability.
4. Better Representation Of Real world Objects.
5. Better Software Maintenance.
6. Enhanced Security.
7. Easy Code Modification.
www.learncomputerscienceonline.com

**Figure 1.** *https://www.learncomputerscienceonline.com/wpcontent/uploads/2021/01/Advantages-Of-Object-Oriented-Programming.jpg*

Object-Oriented Programming (OOP) comes with a range of benefits that make it a go-to approach for software development. It's great for managing large projects because it organizes code into smaller, reusable parts, making complex systems easier to handle. With features like encapsulation, OOP helps protect sensitive data by keeping the inner workings of objects hidden and only exposing what’s necessary.

It also saves time and effort by allowing developers to reuse classes and objects in different parts of a program or even across projects. OOP is particularly useful for presenting real-world concepts, as it lets you create objects that mimic real-life entitles with specific traits and actions. It also makes maintaining and updating software simpler since you can modify parts of the code and reduce vulnerabilities. Finally, with tools like inheritance and polymorphism and efficient way to build reliable, scalable software.

## III. Conclusion

In today's laboratory session, our professor introduced us to Python Fundamentals and its various applications in programming. The session began with an in-depth discussion of core topics, such as Python Indentation, Comments, Variable Naming Conventions, and Variables. We also delved into concepts like Casting, the proper use of Double or Single Quotes, and Python’s Case Sensitivity rules. Additionally, we explored how to work with Multiple Variables, assigning One Value to Multiple Variables, and Output Variables. The discussion progressed to a thorough explanation of Python's operators, including Arithmetic, Comparison, Logical, Identity, and Bitwise Operators. Each topic was explained with examples, helping us understand their real-world applications in coding and problem-solving.

After the lecture, we had the opportunity to apply what we learned by completing an activity called Application 1. This task required us to integrate the topics discussed earlier, allowing us to practice the fundamentals in a hands-on manner. The activity reinforced our understanding and helped us see how these concepts work together in actual programming scenarios. The session was not only educational but also engaging and interactive, making the learning process more enjoyable. Overall, it was a productive and insightful day that gave us a solid foundation in Python programming and inspired us to further explore its potential.

## References

[1]

R. Python, “Object-Oriented Programming (OOP) in Python 3 – Real Python,” *realpython.com*. https://realpython.com/python3-object-oriented-programming/#what-is-object-oriented-programming-in-python

[2]

geeksforgeeks, “Python Classes and Objects,” *GeeksforGeeks*, Oct. 15, 2019. https://www.geeksforgeeks.org/python-classes-and-objects/

[3]

D. Amos, “Object-Oriented Programming (OOP) in Python,” *Realpython.com*, Dec. 15, 2024. https://realpython.com/python3-object-oriented-programming/#instance-methods (accessed Jan. 18, 2025).

[4]

D. Amos, “Object-Oriented Programming (OOP) in Python,” *Realpython.com*, Dec. 15, 2024. https://realpython.com/python3-object-oriented-programming/#instance-methods (accessed Jan. 18, 2025).

[5] Figure 1

https://www.learncomputerscienceonline.com/wpcontent/uploads/2021/01/Advantages-Of-Object-Oriented-Programming.jpg