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:
Uy, Junichiro H.
Saturday - 12 nn – 8:30 pm / 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:

By the end of this tutorial, you'll grasp that object-oriented programming (OOP) in Python revolves around creating classes that serve as blueprints for objects. These objects store data and have methods that help manipulate that data. The main principles of OOP in Python include encapsulation, inheritance, abstraction, and polymorphism. To create an object, you instantiate a class by calling its name followed by parentheses. Python's class inheritance enables one class to inherit properties and methods from another class, called the parent class. You can use the super() function in Python to call a method from the parent class, enabling you to extend or adjust inherited behavior.

Before OOP, there were two popular programming paradigms: procedure-oriented programming (POP) and functional-oriented programming. In functional programming, the focus is on functions. These functions can be passed as parameters, returned, or used within other functions. They are designed to consistently produce the same output for a given input. Languages like Erlang, Haskell, and Sakaila follow this paradigm. Meanwhile, POP, which uses languages like C, Pascal, and Fortran, divides a program into smaller tasks through procedures and functions.

II. Methods

General Instruction:

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

1. Classes

- Classes enable the creation of custom data structures in programming. They include methods which define the actions and behaviors that objects created from the class can perform using their data. A class contains data members and member functions, which can be accessed by creating an instance of the class. It serves as a template that defines common properties and methods shared by all objects of a particular type. In Python, classes are defined using the class keyword followed by a name and a colon. The .__init__() method is used to initialize the attributes for each instance of the class.

2. Objects

- An object is the fundamental unit of Object-Oriented Programming, representing real-world entities. It is created as an instance of a class. While defining a class does not allocate memory, memory is allocated when the class is instantiated to create an object. Every object has an identity, a state, and a behavior. Objects store data and include code to manipulate that data. They can interact with one another without needing to know the specifics of each other’s data or code; it is enough to understand the type of messages they accept and the responses they provide.

3. Fields

- Objects can store data using variables that belong to them, commonly referred to as fields. Fields can be categorized into two types: class variables and object variables, depending on whether they are owned by the class or the object.
- Class variables are shared among all instances of a class. There is only one copy of the class variable, and any changes made to it by one instance are reflected across all other instances.
- Object variables, on the other hand, are unique to each individual object or instance. Every object maintains its own copy of the variable, which is independent of variables with the same name in other instances.
- To better understand these concepts, an example (saved as oop_objvar.py) can provide clarity.

4. Methods

- The properties of an object are determined by its attributes, while its behavior is defined through methods. In Python, methods are written within a class and serve as reusable blocks of code that can be called at any point in a program.
- Methods in Python are essentially functions associated with an object that can manipulate its data or perform specific actions. These are invoked using dot notation, where the object name is followed by a period and the method name. Methods play a vital role in implementing object-oriented programming in Python.

There are three main types of methods in Python:

- Instance Methods: These are the most common methods in Python classes, designed to interact with specific instances (objects). They typically have self as their default parameter, which refers to the instance the method is being called on. The name of this parameter can be changed, but the convention is to use self. Any method created within a class is an instance method unless explicitly specified otherwise.

- Class Methods: These methods are tied to the class itself rather than any individual instance. They are used to set or retrieve class-level information and cannot modify instance-specific data. Class methods are defined using the class method decorator and generally take cls (representing the class) as their default parameter.
- Static Methods: Static methods are independent of both the class and its instances. They neither access nor modify class or instance data. These methods operate in isolation, focusing solely on the task they are designed to perform. Static methods are defined using the @staticmethod decorator and do not require self or cls as parameters.

5. Properties

- Properties in OOP are also able to be looked at as functions, a property houses a function that can have its procedures or variables altered without directly going to

the code and editing it. A property can be changed or updated based on user input which allows for a lot of user-interactive programs and applications.'

- In Python, the property keyword allows you to define getter, setter, and deleter methods for class attributes, making them act like properties. This enables you to control the access and modification of class attributes while providing a clean interface for external code. In example, we use the @property decorator to create getter methods for each attribute, allowing us to access these attributes using obj.attribute1

III. Results

In object-oriented programming (OOP), a class is a blueprint that defines the structure and behavior of objects, like a template for creating cars. An object is a specific instance of a class, such as a red Tesla. Fields are variables within a class that store data about the object, like its color or speed. Methods are actions or behaviors that the object can perform, such as driving or braking. Properties are special methods used to control access to fields, ensuring the data is handled properly, like checking a car's speed before updating it.

These concepts, class, object, fields, methods, and properties—are essential to structuring code in a way that models real-world scenarios, supporting modularity, reusability, and easier maintenance in software design. These principles are further explored in works such as Design Patterns: Elements of Reusable Object-Oriented Software by Gamma, Helm, Johnson, and Vlissides (1994).

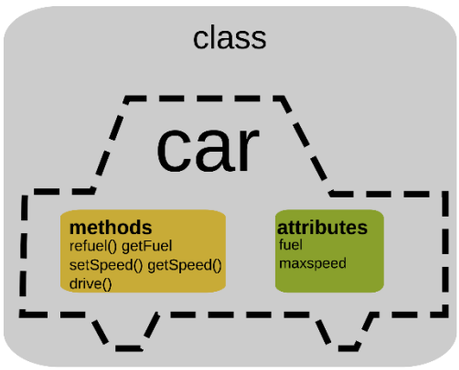

Figure 1. Understanding Object-Oriented Programming through a car example

https//www.unionsquaredesign.com

Here's an image that illustrates the core concepts of object-oriented programming (OOP) using a car as an example. One commonly cited explanation of object-oriented programming (OOP) concepts comes from the book Object-Oriented Analysis and Design with Applications by Grady Booch. Booch explains, "The essence of OOP lies in the identification and organization of classes and objects."

IV. Conclusion

To sum it up, object-oriented programming (OOP) is a way to organize code by using classes, objects, fields, methods, and properties, which help model real-world things in software. A class is like a blueprint, and when you create an object, you're making a specific instance of that blueprint. Fields are like data or details about the object (like its color or speed), while methods define what the object can do, like drive or stop. Properties are special methods that give you control over how data is accessed and updated, helping to keep things secure and well-managed.

The main principles of OOP—encapsulation, inheritance, abstraction, and polymorphism—make the code modular, reusable, and easier to maintain. These ideas let you structure your code in a way that's easier to manage, modify, and expand as your project grows. Using the car example, we can see how these concepts help break down complex systems into simpler, understandable parts, making the code more efficient and organized.

Overall, OOP is a practical and flexible way to write code that mirrors the way we think about the world, and it helps programmers build better, more manageable software.

Reference

Book

[1]

Website
[2]

Real Python, "Python 3 Object-Oriented Programming," Real Python. [Online]. Available: https://realpython.com/python3-object-oriented-programming/#how-do-you-instantiate-a-class-in-python. [Accessed: Jan. 18, 2025].

GeeksforGeeks, "Introduction of Object-Oriented Programming," GeeksforGeeks. [Online]. Available: https://www.geeksforgeeks.org/introduction-of-object-oriented-programming/. [Accessed: Jan. 18, 2025].

Swaroop C H, "Object Oriented Programming," Python Documentation. [Online]. Available: https://python.swaroopch.com/oop.html. [Accessed: Jan. 18, 2025].

Analytics Vidhya, "Basic Concepts of Object-Oriented Programming & Types of Methods in Python," Analytics Vidhya. [Online]. Available: https://www.analyticsvidhya.com/blog/2020/11/basic-concepts-object-oriented-programming-types-methods-python/. [Accessed: Jan. 18, 2025].

Wikibooks, "Object-Oriented Programming/Properties," Wikibooks. [Online]. Available: https://en.wikibooks.org/wiki/Object_Oriented_Programming/Properties. [Accessed: Jan. 18, 2025].

Union Square Design, "Object-Oriented Programming," Union Square Design. [Online]. Available: https://www.unionsquaredesign.com/2016/11/14/object-oriented-programming/?fbclid=IwZXh0bgNhZW0CMTEAAR01YZGKJ7KcbgDWiXYLbdfXm6btX20ncbWCcLfBDR0J8HiFeYgQQX9Hk_g_aem_RJeL9ZBQTkL6GQCEykyhyQ. [Accessed: Jan. 18, 2025].