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:
Regondola, Jezreel P.
12:00 – 8:30 Sun / BSCpE – 1A

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

   A class represents a template or blueprint for creating objects in OOP. It specifies the structure and behavior objects of a class are expected to have. Classes are created using the keyword class. The attributes are variables defined within the class. Attributes represent properties of the class.

   Example:

```
class Car:
    def __init__(self, model, car_type):
        self.model = model
```

2. Objects

   An object is a class instance. It is the representation of a particular entity which has been built with the blueprint of the class. Objects are the actual entities holding data and interacting in an OOP system. They possess states, which are defined by fields, and behaviors, which are defined by methods.

   Example:

```
car1 = Car("Lamborghini")
car2 = Car("Ferrari")
car3 = Car("Electric")
```

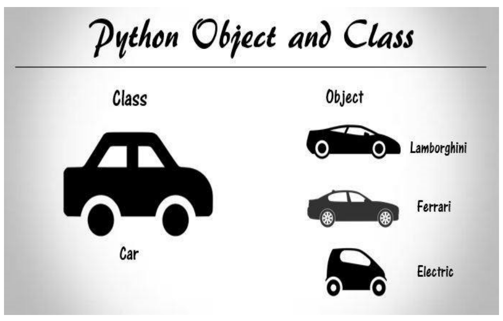


Figure 1: Python Object and Class

Source: Python Object And Class (Tutorial With Examples)

3. Fields

   A field or an attribute of a class is actually a variable which holds data or state in the class or object. It may be shared with all the objects created or different for each of them. An example of such an attribute could be the color in the class Car.

   Example:

```
class Car:
    def __init__(self, color):
        self.color = color
```

4. Methods

   Methods are functions written inside a class that describe behaviors of objects created by the class. Methods enable actions to be implemented or interaction from one object toward another. There are two instances: methods based on instance and class-level operation.

   Example:

```
class Car:
    def drive(self):
        print("The car is driving")
```

5. Properties

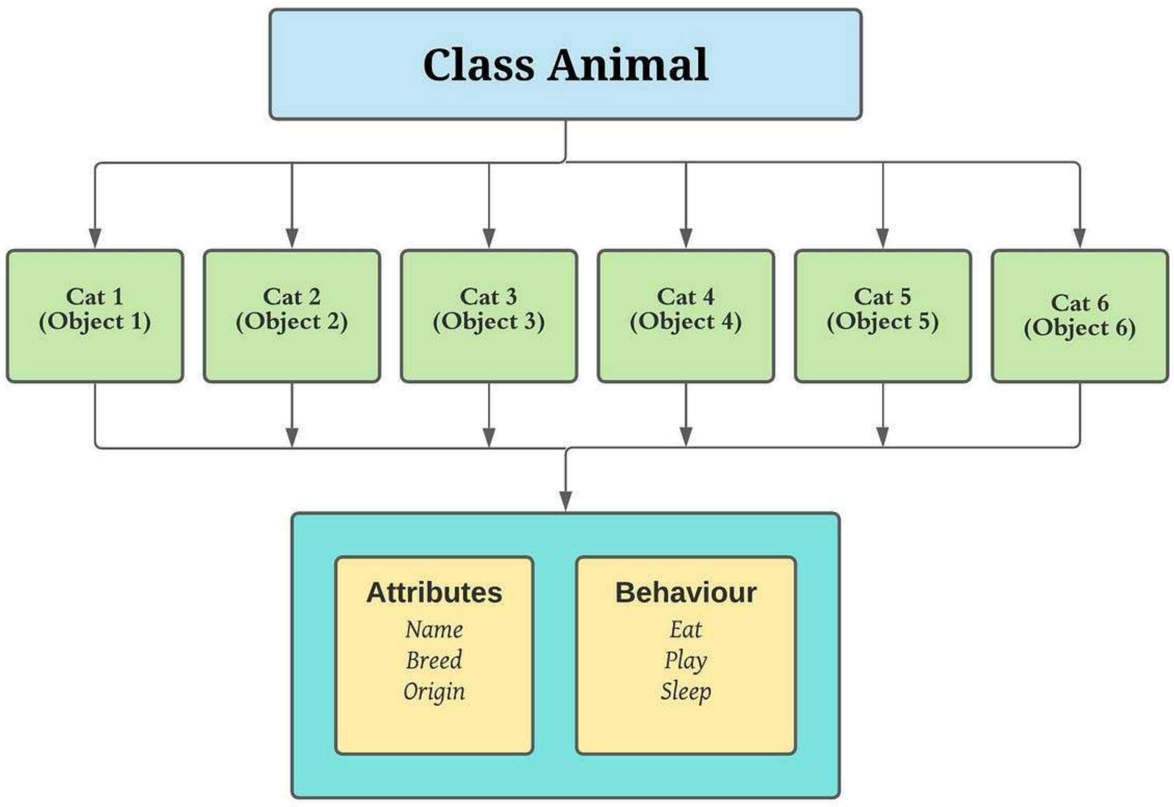
   Properties can be used to control access to fields in a class. They allow encapsulation with the provision of getter, setter, and deleter methods. Properties can validate or modify data when fields are accessed or updated. They are often used to enforce rules about field access.

   Example:

```
@property
def color(self):
    return self._color
@color.setter
def color(self, value):
    if value in ["Red", "Blue", "Green"]:
        self._color = value
    else:
        raise ValueError("Invalid color")
```

III. Results

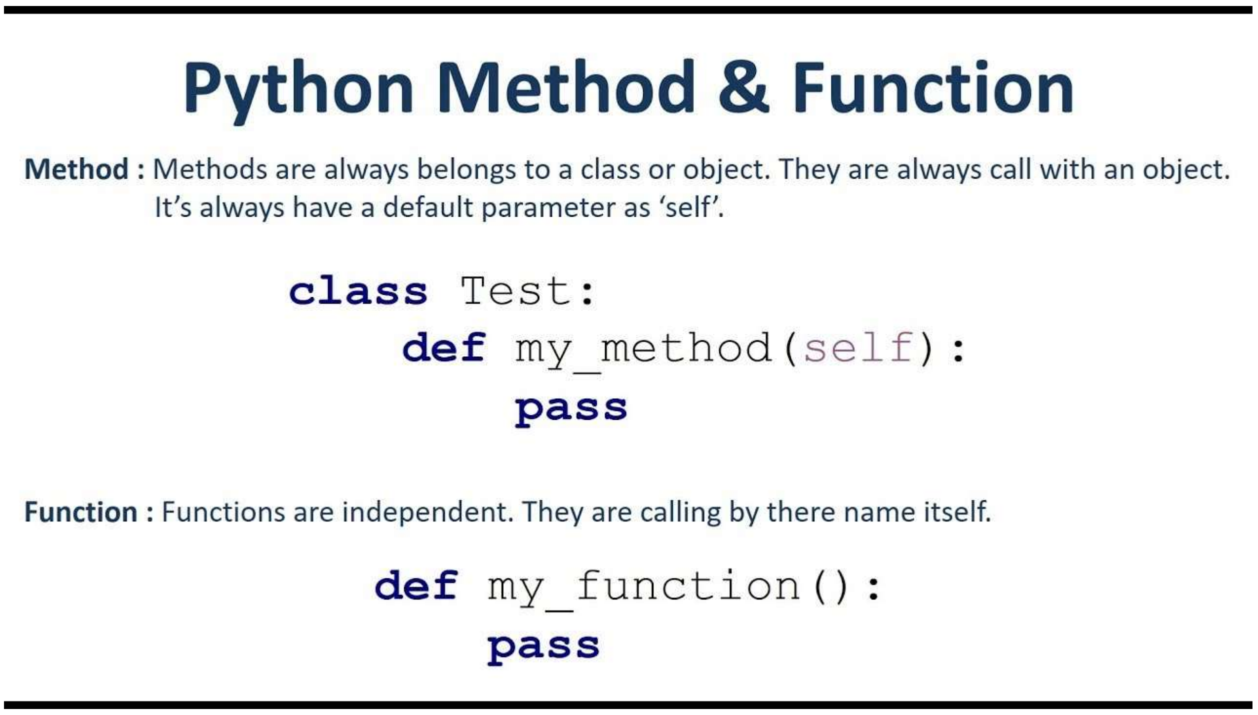
Figure 2: Sample Diagram of Classes and Object



Source: [PYTHON CLASSES AND OBJECTS. Classes | by Sathvik S Bangera | Medium]

A class is the blue print which will create the objects. Class defines a group of attributes/data and behaviors/operations which would be inherent with the objects instantiating from class. In this diagram, class Animal represents blueprint for creation of objects such as Cat 1, Cat 2, etc. The class Animal defines the attributes, such as Name, Breed, Origin, and behaviors, like Eat, Play, Sleep, that all objects will share.

An object is an instance of a class. It is the concrete realization of the blueprint defined by the class, with its specific values for the attributes and its ability to perform the behaviors. In this diagram, each cat (Cat 1, Cat 2, ., Cat 6) is an object created from the Animal class. Each object may have unique values for the attributes (Name, Breed, Origin) while sharing the same behaviors (Eat, Play, Sleep).



Python Method & Function

Method : Methods are always belongs to a class or object. They are always call with an object. It's always have a default parameter as 'self'.

```
class Test:
    def my_method(self):
        pass
```

Function : Functions are independent. They are calling by there name itself.

```
def my_function():
    pass
```

Figure 3: Python Method & Function

Source: What is Methods and Functions in Python

A class is the blue print which will create the objects. Class defines a group of attributes/data and behaviors/operations which would be inherent with the objects instantiating from class. In this diagram, class Animal represents blueprint for creation of objects such as Cat 1, Cat 2, etc.
The class Animal defines the attributes, such as Name, Breed, Origin, and behaviors, like Eat, Play, Sleep, that all objects will share.

An object is an instance of a class. It is the concrete realization of the blueprint defined by the class, with its specific values for the attributes and its ability to perform the behaviors. In this diagram, each cat (Cat 1, Cat 2, ., Cat 6) is an object created from the Animal class. Each object may have unique values for the attributes (Name, Breed, Origin) while sharing the same behaviors (Eat, Play, Sleep).

IV. Conclusion

This lab activity gave an overview of OOP concepts by implementing them using Python. Major components and concepts such as classes, objects, fields, methods, and properties were illustrated and discussed in detail so that how all these parts and pieces come to emphasize defining structure and behavior of OOP systems. This language gives simplicity and readability in the approach of Python OOP. The diagrams and examples clarified the relationship between attributes and behaviors further, thereby reemphasizing the significance of OOP concepts in software development. With the application of knowledge about OOP in C++ to Python, the transition is seamless due to shared principles such as encapsulation, inheritance, and polymorphism. While Python reduces its syntax and automates memory management compared to that of C++, the fundamental comprehension of these in C++ really gives a nice base for writing efficiently in

Python, showing again that OOP principles are rather universal and widely applicable across varying programming languages.

Reference

Website

[1] *"How to define _fields in a class with Validator descriptors in Python | LabEx," LabEx. https://labex.io/tutorials/python-how-to-define-fields-in-a-class-with-validator-descriptors-in-python-417438*

[2] GeeksforGeeks, "Python Classes and Objects," *GeeksforGeeks*, Oct. 15, 2019. Accessed: Jan. 18, 2025. [Online]. Available: https://www.geeksforgeeks.org/python-classes-and-objects/

[3] GeeksforGeeks, "Python property() function," *GeeksforGeeks*, Sep. 21, 2018. Accessed: Jan. 18, 2025. [Online]. Available: https://www.geeksforgeeks.org/python-property-function/

[4] P. Team, "Python Methods vs Functions," *Python Geeks*, Jun. 02, 2021. Accessed: Jan. 18, 2025. [Online]. Available: https://pythongeeks.org/python-methods-vs-functions/