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:
Ruperto, April Anne A.
Sat 12:00PM-4:00PM and 4:30PM-8:30PM/ BSCpE – 1A

Submitted to
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 OOP 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 acts as a template for making an object. It defines the data attributes, also known as fields or properties, and methods that define the objects created from these classes. (Moore, et al., 2025) A class is a way of organizing information about a type of data so a programmer can reuse elements when making multiple instances of that data type.
2. **Objects** – An object is a collection of data with associated behaviors. (Gillis and Lewis, 2025) These can correspond to the real-world objects or entity, and these are represented by a class that is created with data that has been specified. (Lott and Phillips, 2021)
3. **Fields** – Fields are variables declared within a class that contain object-specific data; they are described as properties or attributes of an object.
4. **Methods** - Functions that objects can perform are called methods. They are defined within a class that explains how an object behaves. Every method in class definitions begins with an instance object reference. Instance methods also refer to the subroutines that are part of an object. (Gillis and Lewis, 2025)
5. **Properties** – Attributes are commonly referred to as members or properties. Some authors argue that the phrases have different meanings, with attributes being settable and properties being read-only. A Python property can be defined as read-only, but the value will be predicated on attribute values that are ultimately editable, rendering the concept of read-only somewhat meaningless. (Lott and Phillips, 2021)

III. Results

Figure 1. Python Logo

Source: Python Logo PNG Transparent – Brands Logos

Figure 2. Guido van Rossum (creator of Python)

Source: Guido van Rossum, el creador de Python, se une a Microsoft. – Blog EHCGroup

Python was created in the early 1990s by Guido van Rossum at Stichting Mathematisch Centrum (CWI, see https://www.cwi.nl/) in the Netherlands. Python has evolved significantly since its introduction, with various versions introduced over the years. Understanding these versions and the reasons some are no longer functional is essential for developers and users alike.
Key Versions of Python

1. **Python 0.9.0** (February 1991)
    - The first public release, introducing core features like exception handling and functions.
2. **Python 1.0** (January 1994)
    - Official public release that included functional programming constructs.
3. **Python 2.0** (October 2000)
    - Introduced list comprehensions and garbage collection, laying the groundwork for modern Python.
4. **Python 2.7** (July 2010)
    - The last major release of the Python 2.x series, designed to ease the transition to Python 3.x by incorporating some features from it.
5. **Python 3.0** (December 2008)
    - A major overhaul that was not backward compatible with Python 2.x, introducing new syntax and removing outdated features.
6. **Subsequent Python 3.x Releases**
    - Ongoing improvements with versions like Python 3.6 (December 2016), 3.7 (June 2018), up to the latest versions like Python 3.12 (October 2023) which continue to add features and enhancements.

As our world continues to be developed, programming languages continue to evolve in making updates, security patches, or bug fixes to fix the security vulnerabilities that can be exploited. As the community moves forward with newer versions, resources such as documentation, tutorials, and community forums increasingly focus on current versions, leaving older versions without support or guidance.

For Python fundamentals, it can be accessed through this hyperlink: CPE-103-OOP-1A/Python_Fundamentals.ipynb at main · Ruperto-April-Anne/CPE-103-OOP-1A

IV. Conclusion

In summary, the development of Python and its basic concepts illustrate the flexibility of the language and its importance in modern software engineering. Python's designs support the reusability of code and the modularity of its development, making it a preferred choice for developers across various fields in software development.

The evolution of Python from its initial iterations to the more advanced 3.x series has introduced vital features that not only augment functionality but also mitigate security weaknesses. The transition from Python 2 to Python 3 was a crucial point in the development of the language, and the importance of keeping up with modern standards for peak performance and support is clear.

As the world of programming continues to evolve, a developer must understand such core concepts as well as version changes and the impact on code. Keeping pace with the new developments helps the programmer make full use of all that Python offers while maintaining the best practices of software design. Using the newest versions makes the programmer more efficient and creates a safer and better environment for coding.

Reference

Book

[1] *Python Object-Oriented Programming*. (n.d.). Google Books. https://books.google.com.ph/books?hl=en&lr=&id=J5Y2EAAAQBAJ&oi=fnd&pg=PP1&dq=object+oriented+programming+concepts+about+classes+in+python&ots=j77fJNIceT&sig=mqfOKm0T0NiXdb85D94lo9wMfHM&redir_esc=y#v=onepage&q=object%20oriented%20programming%20concepts%20about%20classes%20in%20python&f=false

Website

[2] Gillis, A. S., & Lewis, S. (2024, June 14). *object-oriented programming (OOP)*. https://www.techtarget.com/searchapparchitecture/definition/object-oriented-programming-OOP

[3] *Python Version History*. (n.d.). https://www.tutorialsteacher.com/python/python-version-history

[4] GeeksforGeeks. (2024, September 25). *Python Version History*. GeeksforGeeks. https://www.geeksforgeeks.org/python-version-history/

[5] *Welcome to Python.org*. (2025, January 15). Python.org. https://www.python.org/