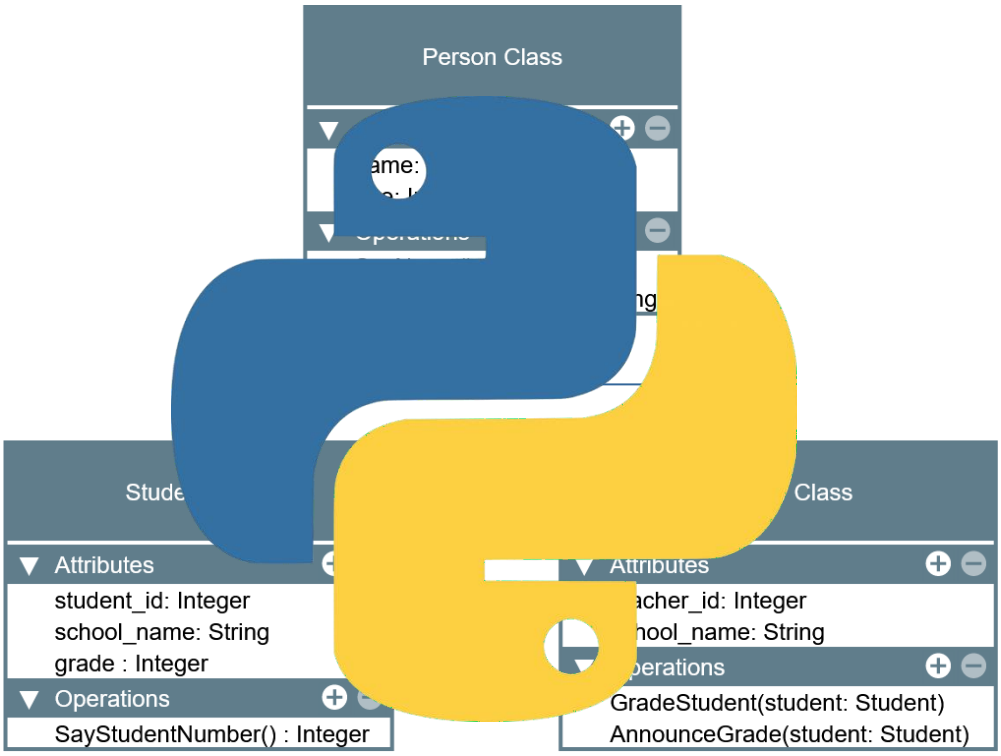
UNIVERSITY OF CALOOCAN CITY
Caloocan, 1400 Metro Manila, Philippines

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



LABORATORY MANUAL

Object-Oriented Programming
(CPE 103)

UNIVERSITY OF CALOOCAN CITY
Caloocan, 1400 Metro Manila, Philippines

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

Laboratory Activity No. 2.2	
Literals, Operators, and Variables	
Course Code: CPE103	**Program:** BSCPE
Course Title: Object-Oriented Programming	**Date Performed: FEB. 1, 2025**
Section: 1A	**Date Submitted: FEB. 7, 2025**
Name: CATAHAN, JOSHUA A.	**Instructor:** ENGR. MARIA RIZETTE SAYO

1. Objective(s):

This activity aims to familiarize students with using literals and variables in Python, applying them to calculate student grades, and properly document the program with comments and Markdown

2. Intended Learning Outcomes (ILOs):

The students should be able to:
2.1 Write a simple program implementing literals and variables.
2.2 Use comments and identify keywords from identifiers created by users.

3. Discussion:

In Python, variables are used to store values that can change during the program. For example, if you're working on a program that collects the user's age, you'd store that value in a variable, let's say age. This makes it easy to update the value every time the user provides input. On the contrary, constants are values that remain the same throughout the program. For instance, if you're working on a program that calculates the area of a circle, you could use a constant like PI = 3.14, which represents the value of pi and is used whenever needed. Literals are fixed values that appear directly in the code. For example, when you write radius = 10, the number 10 is a literal representing the radius. The main difference is that variables can change their values throughout the program, constants stay the same once defined, and literals are direct values used in expressions or assignments. By using these elements, the program becomes more organized and easier to maintain

4. Materials and Equipment:

Desktop Computer with Anaconda Python /Python Colab
Windows Operating System

5. Procedure:

Perform the activity using the GOOGLE COLAB

1. I opened the Google Colab from the repository of our professor.

2. I analyzed the given task and formulated the appropriate computations and codes to get the following outputs.

3. I then implemented and executed the code to obtain the required results

Task:

A teacher wants to calculate the final grade in a CpE course and want to write it in a python program. The following are the requirements:

1. PRELIM GRADE = 50% Prelim Exam + 50% Prelim Class Standing (CS)
2. PRELIM CS = 50% Hands-on activity + 30% Quiz + 20% Assignment
3. MIDTERM GRADE = 1/3 of PRELIM GRADE + 2/3 of (50% Midterm Exam + 50% Midterm Class Standing (CS))

4. MIDTERM CS = 50% Hands-on activity + 30% Quiz + 20% Assignment
5. FINAL GRADE = 1/3 of MIDTERM GRADE + 2/3 of (50% Final Exam + 50% Final Class Standing (CS))
6. FINAL CS = 50% Hands-on activity + 30% Quiz + 20% Assignment
7. HOAs, Quizzes and Assignments are inputted as average of all submissions and are out of 100%.
8. Major exams are inputted out of 100%.
9. Show the codes that successfully run the program.
10. Provide comments or documentation strings for your program.

PLEASE VISIT THIS LINK FOR THE CODE: https://colab.research.google.com/github/Joshua-Catahan/CPE-103-OOP-1-A/blob/main/CPE_103_Laboratory_No_2_2_Literals%2C_Operators%2C_and_Variables_Supplementary_Activity(FINAL_COPY).ipynb#scrollTo=WORPHnvZKBVE&line=3&uniqifier=1

6. Supplementary Activity:

1. Test 3 students from the program you created.
2. The program should show the name of the student, the PRELIM, MIDTERM and FINAL grades.
3. Convert the final grade into the UCCs numerical grade. Please refer to the grading system.

PLEASDE VISIT THIS LINK FOR THE CODE:

1ST STUDENT: https://colab.research.google.com/github/Joshua-Catahan/CPE-103-OOP-1-A/blob/main/CPE_103_Laboratory_No_2_2_Literals%2C_Operators%2C_and_Variables_Supplementary_Activity(FINAL_COPY).ipynb#scrollTo=X2QfbKrmKOJK&line=18&uniqifier=1

2ND STUDENT: https://colab.research.google.com/github/Joshua-Catahan/CPE-103-OOP-1-A/blob/main/CPE_103_Laboratory_No_2_2_Literals%2C_Operators%2C_and_Variables_Supplementary_Activity(FINAL_COPY).ipynb#scrollTo=H0oI4lxIFDtt&line=7&uniqifier=1

3RD STUDENT: https://colab.research.google.com/github/Joshua-Catahan/CPE-103-OOP-1-A/blob/main/CPE_103_Laboratory_No_2_2_Literals%2C_Operators%2C_and_Variables_Supplementary_Activity(FINAL_COPY).ipynb#scrollTo=S3ATkG4EGsDb&line=7&uniqifier=1

PS: ALL CODES ARE IN THE SAME LINK BUT DIFFERENT CELLS

Questions:

1. How does using functions in Python help make the code more readable and maintainable, especially when handling repetitive tasks?

Using functions in Python makes the code easier to read and manage because we can reuse the same logic without repeating

ourselves. It helps keep the code short and organized. This also reduces the chance of errors by isolating specific tasks into their own functions.

2. What advantages do you see in breaking down a complex task (like grade calculation) into smaller, more manageable pieces, and how does this approach benefit the clarity of the code?

Breaking down tasks like grade calculation into smaller parts helps keep the code clean and simple, making it easier to understand and update later. This approach makes the whole process less overwhelming. It also allows for easier debugging since each part is simpler to check.

7. Conclusion:

Through this process, I've learned how much cleaner and simpler the Python language is compared to C++. By using Python functions, I was able to create shorter lines of code. In this activity, I initially tried to write the code in C++, and when compared to Python, it was much longer and messier. Of course, this was partly due to my lack of knowledge at the time. As I continued, I realized that by handling tasks like class standing and exams separately, the code became cleaner and easier to understand. I also saw that when all inputs are the same, like 100 in this case, the calculations remain consistent, which helps ensure the accuracy of the code. In the end, the grade point equivalence worked as expected, clearly showing how grades translate into the point system. Overall, this laboratory activity has further enhanced my understanding of how classes and objects can be used effectively.

8. Assessment Rubric: