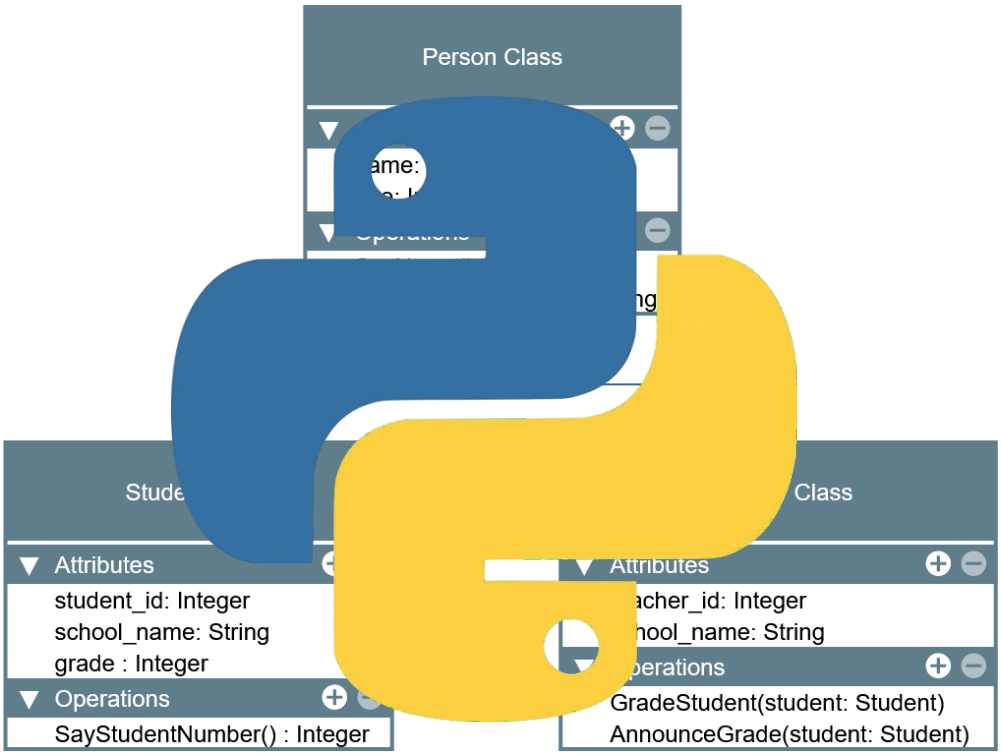
UNIVERSITY OF CALOOCAN CITY
Caloocan, 1400 Metro Manila, Philippines

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. 2.1	
Literals, Operators, and Variables	
Course Code: CPE103	**Program:** BSCPE
Course Title: Object-Oriented Programming	**Date Performed:** 01/25/25
Section: 1-A	**Date Submitted:** 01/25/25
Name: Nerio, Hannah Grace A.	**Instructor: Engr. Maria**

1. Objective(s):

This activity aims to familiarize students in the various data types of Python, assign values to variables, and perform operations in a Python program.

2. Intended Learning Outcomes (ILOs):

The students should be able to:
2.1 Assign different values to variables in Python
2.2 Perform different operations available with variables in Python

3. Discussion:

The **Python** programming language is an interpreted language meaning the lines are evaluated line -by-line at runtime because there is no compile time at Python. This means that Python can dynamically allocate memory to variables as needed depending on the line of code that it interprets that is why Python is also referred to as a Dynamically typed language.

Like other programming languages such as C/C++ and Java, Python can also assign values to specific blocks of memory through variables as well as perform operations such as but not limited to Addition, Subtraction, Multiplication, Division, and Modulo(remainder). This activity will focus on assigning values and performing operations in Python.

Recall that a **variable** is a name that points to a specific location in memory where the data is stored. A variable can be allocated memory based on the data type it is assigned with which in Python can be: **Integer**, **Float**, **Complex Number**, **Boolean**, and **String**. In Python, **lists**, **tuples**, and **dictionaries** are also referred to as data types specifically sequences. More information can be found here (https://docs.python.org/3.8/reference/datamodel.html?highlight=data%20type#objects -values-and-types). These will be discussed further in lab activities.

Variables in Python are assigned in the following manner:

```
variable_name = value
```

Literals refers to the raw data given in a variable or constant. Literals can be some of the following: Numeric, Complex, String, Boolean, Special. Other literals are list, tuple, dict, set, and Unicode literals.

4. Materials and Equipment:

Desktop Computer with Anaconda Python /Python Colab
Windows Operating System

5. Procedure:
Perform the activity using the Jupyter Notebook This activity can be done either locally on Anaconda's Jupyter Notebook or online through Google Collaboratory which offers a free Jupyter Notebook environment for Google Users. IPython Notebook files (.ipynb) that are saved in the Google Drive can be opened on Google Collaboratory. Additional guides are available on the IPython Notebook template file that is provided wit h this activity. If the template is not present, these are the valuable links for reference:

UNIVERSITY OF CALOOCAN CITY
Caloocan, 1400 Metro Manila, Philippines

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

https://jupyter-notebook.readthedocs.io/en/stable/examples/Notebook/Notebook%20Basics.html
https://colab.research.google.com/notebooks/welcome.ipynb
https://colab.research.google.com/notebooks/markdown_guide.ipynb

Assigning variables of different data types in Python

1. In an empty cell, declare a variable **value** and assign it the value of 5 then display its value using print().
2. Create a new cell and type the command: type(value) then run the cell. The output should be like the image below.

```
In [3]: type(value)

Out[3]: int
```

3. In a new cell, use the same variable **value** and assign it the value of 5.0 then print the value.
4. Repeat step 2.
   Note: You may choose to decide how you execute the code in the cells for the next tasks in the procedure.
5. Repeat these steps for the following values:
   a. 2+3j
   b. 'Hello World'
   c. "Hello World"
   d. True
   e. False
   f. [1,2,3,4,5]
   g. (1,2,3,4,5)
   h. { 'name': 'Your_name' }
   i. None
6. Re-assign the **value** variable to be equal to 5.
7. Declare a new variable named **value2** to be equal to -6.

FOR CHECKING PLEASE REFER TO THIS LINK:
https://github.com/HannahGraceNerio/CPE-103-OOP-1-A/blob/main/Laboratory_2.ipynb

UNIVERSITY OF CALOOCAN CITY
Caloocan, 1400 Metro Manila, Philippines

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

Performing Operations with Python

1. Using **value** and **value2**. Type the command: print(**value**+**value2**)
2. Repeat step 1 for the following values of **value** and **value2**:
   Hint: You may try using this assignment ***value, value2 = 5, -6*** in the Notebook for the following steps:
   a. value, value2 = 5.0, 6
   b. value, value2 = -5, 6.1
   c. value, value2 = “Hello”, ‘world’
      Note: Modify the code so that hello and world would be separated.
   d. value, value2 = [1,2,3], [4,5,6]
   e. value, value2 = (1,2,3), (4,5,6)
   f. value, value2 = {"name":"Royce"}, {"age":2}
      Note: Observe the outputs carefully and try repeating them using subtraction.

      FOR CHECKING PLEASE REFER TO THIS LINK: https://github.com/HannahGraceNerio/CPE-103-OOP-1-A/blob/main/Laboratory_2.ipynb

3. Using value, value2 = 30, 4. Type the commands:
   a. print(value*value2)
   b. print(value2**2)
   c. print(value2**3)
   d. print(value*value2+value2**2+1)
   e. print(value/value2)
   f. print(value%value2)

FOR CHECKING PLEASE REFER TO THIS LINK:
https://github.com/HannahGraceNerio/CPE-103-OOP-1-A/blob/main/Laboratory_2.ipynb

Receiving Input Data using Python

Data can be received through keyboard input in Python by using the input() function. The input function has the following syntax:

input("Message Name")

The "Message Name" is an optional String parameter that can be customized to prompt the user for a message instead of having to print a message prompt separately. The default return value of the input() function is a String containing the value received from the keyboard. This value can be assigned to a variable shown in the example below:

name = input("Enter your name: ")

UNIVERSITY OF CALOOCAN CITY
Caloocan, 1400 Metro Manila, Philippines

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

Assigning Input Data to a Variable

Finding a person's BMI (metric)

1. Declare a new variable named **name** and assign it the value **input("Enter your name")**
2. Create another variable named **weight** and assign it the value **input("Enter your weight(kg): ")**
3. Create another variable named **height** and assign it the value **input("Enter your meters(m): ")**
4. Declare another variable called **bmi** and assign it the formula $bmi = \frac{weight}{height^2}$
5. Address the errors displayed step#4. You can accomplish this by converting the String input to another data type. An example would be:

   weight = input("Enter your weight(kg)")
   weight = float(weight)

   Or simply **weight** = float(input("Enter your weight(kg): "))

   There are many functions available that can convert one data type to another. Some of which are the following:
   int(), float(), str()

   Other functions which maybe used in the later lab activities are: complex(real, imaginary), list(), tuple(), set(), dict(), ord(), bin(), hex(), oct().

6. Print the persons's name, weight, height, and bmi
   Name: John Ray
   Weight: 60
   Height: 1.6764
   BMI = 21.3499

   Guide: 5.5 feet ~ 1.6764 m

   FOR CHECKING PLEASE REFER TO THIS LINK: https://github.com/HannahGraceNerio/CPE-103-OOP-1-A/blob/main/Laboratory_2.ipynb

Hint: You can combine two values by converting the output value to String and Concatenating (Addition) the operator on two strings.

print("Value: "+str(12))

You may explore many other methods to format values onto the print() function in Python. Another example is the following:

print("Value: ", 12)

UNIVERSITY OF CALOOCAN CITY
Caloocan, 1400 Metro Manila, Philippines

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

6. Supplementary Activity:

Tasks

1. **Write the Python equivalent code of the following C code:**

```
int main(){
    float base = 0, height = 0, area = 0;
    printf("Enter the base of the triangle: ");
    scanf("%f", &base);
    printf("Enter the height of the triangle: ");
    scanf("%f", &height);
    area = (1/2)*base*height;
    print("The area of the triangle is %f", area);
}
```

2. **Write a program that would convert Celsius to Fahrenheit given the formula: F = (C × 9/5) + 32**
   Example of conversion:

   0°C = 32 °F
   -20°C = -4 °F

3. **Write a program that can determine the distance between two points given the coordinates using the formula:**

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

   Hint/Rule: No library or package is needed to implement this equation.

   Example: x2, y2 = -3, 3 and x1, y1 = 2, 2 d = 5.099019514

   FOR CHECKING PLEASE REFER TO THIS LINK:

https://github.com/HannahGraceNerio/CPE-103-OOP-1-A/blob/main/Laboratory_2.ipynb

Questions:

1. **Give one major difference in syntax that Python has with other languages such as C?**
   In Python there is no need for semicolon and curly braces to terminate the program as compared to C+ it required that every statement should end with semicolon and if you forgot it will show syntax error also the curly braces.

2. **How does variable assignment differ in Python compared with other languages such as C?**
   In Python, variables are used to store data that can be referenced and manipulated during program execution. A variable is essentially a name that is assigned to a value. Unlike many

other programming languages, Python variables do not require explicit declaration of type. The type of the variable is inferred based on the value assigned unlike in C it has declaration of variable.

3. **Try assigning variable names that start with numbers, and special characters. Is the assigning of variables that start with numbers accepted by Python? For Special Characters? Is there an exception for variables special characters?**
   In Python, assigning variable names that start with numbers is not accepted. Variable names must start with a letter (a-z, A-Z) or an underscore (_). For example, 1variable will result in a syntax error. However, special characters can be used in variable names, but only in specific cases. Python allows underscores (_) in variable names, so something like _variable is valid. But other special characters such as @, #, $, %, and so on are not allowed in variable names.

4. **Do the assignment operators (+, -, *, /, %, **) work for all data types? Why or Why not?**
   Assignment operators like +, -, *, /, %, and ** in Python do not work for all data types because their functionality is defined only for certain types. For numeric data types, these operators perform the expected arithmetic operations, such as addition, subtraction, multiplication, division, modulus, and exponentiation. However, for non-numeric data types like strings and lists, some operators have specialized behavior. For example, the + operator can concatenate strings or lists, but other arithmetic operators like - or / are not applicable and will result in a TypeError. This distinction exists because the operations each operator represents are only meaningful within the context of certain data types.

5. **How does the * operator differ from the ** operator?**
   In Python, the * and ** operators serve distinct functions. The * operator is utilized for multiplication and unpacking lists or tuples, making it versatile for both arithmetic operations and simplifying code when dealing with iterable data structures. For instance, multiplying 3 * 4 yields 12. On the other hand, the ** operator is employed for exponentiation and unpacking dictionaries. It handles raising numbers to a power, such as 2 ** 3 which results in 8, and can unpack key-value pairs in dictionaries to be used as function arguments. For example, given my_dict = {'a': 1, 'b': 2}; def func(**kwargs): print(kwargs); func(**my_dict), the output would be {'a': 1, 'b': 2}.

7. Conclusion:

In summary, compared to other programming languages like C, this lab exercise has given me a better knowledge of Python's syntax, variable assignment, and operator usage. Because Python does not require semicolons or curly braces to finish statements like C does, we found that Python's syntax is more simplified and easier to use. Furthermore, C requires explicit type declarations, whereas Python uses dynamic typing and variable assignment, where the type can be determined from the assigned value. The activity explored the restrictions for naming variables in Python, highlighting that variable names cannot begin with digits but can contain underscores, unlike many other special characters. Furthermore, we discovered that Python's assignment operators, such as +, -, *, /, %, and **, only work with specified data types, and using them with incompatible types can result in issues. Finally, we differentiated between the * and ** operators, with the former handling multiplication and unpacking iterables and the latter handling exponentiation and unpacking dictionaries. Overall, these concepts highlight Python's simplicity and flexibility, making it a flexible language for both beginner and professional programmers.

8. Assessment Rubric: