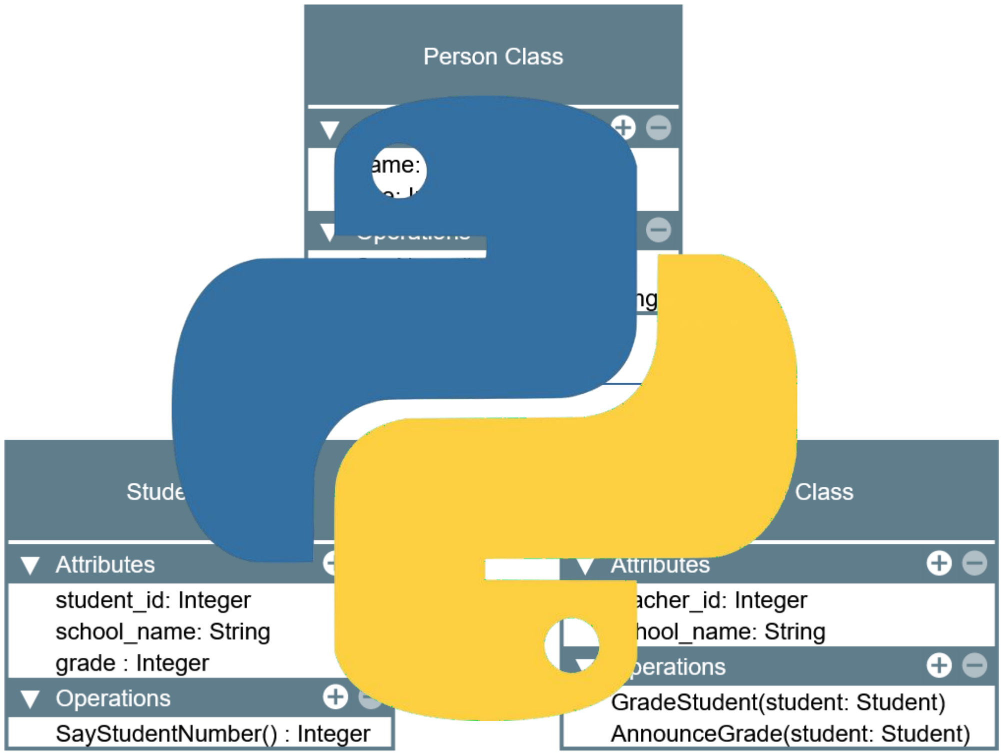
UNIVERSITY OF CALOOCAN CITY
Caloocan, 1400 Metro Manila, Philippines

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: 01/25/2025**
Section: 1A	**Date Submitted:01/25/2025**
Name: CATAHAN, JOSHUA A.	**Instructor: ENGR. MARIA RIZETTE SAYO**

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

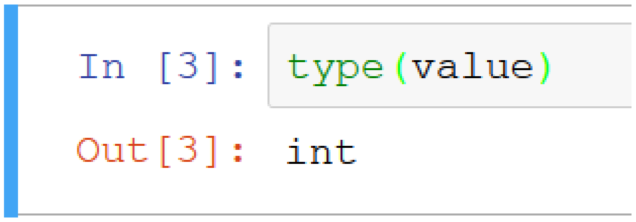
This activity can be done either locally on Anaconda's Jupyter Notebook or online through Google Collaboratory which offers a free Jupyter Notebook environment for Google Users. IPython Notebook files (.ipynb) that are saved in the Google Drive can be opened on Google Collaboratory. Additional guides 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:

Here is the link for the codes done in this activity: https://colab.research.google.com/drive/10hHr9_-opVCs7nOdQDJOFL12Ok_RCoMk?usp=sharing

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.

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.

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)

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: “)

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

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)
```

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);
}
```

LINK FOR THE CODES:
https://colab.research.google.com/drive/10hHr9_-opVCs7nOdQDJOFL12Ok_RCoMk#scrollTo=atXHj2mVvU-o&line=1&uniqifier=1

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

LINK FOR THE CODES:
https://colab.research.google.com/drive/10hHr9_opVCs7nOdQDJOFL12Ok_RCoMk#scrollTo=WjZMDraryIjK&line=1&uniqifier=1

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

LINK FOR THE CODES:
https://colab.research.google.com/drive/10hHr9_opVCs7nOdQDJOFL12Ok_RCoMk#scrollTo=ObYDbDxqzpCw&line=2&uniqifier=1

Questions:

1. **Give one major difference in syntax that Python has with other languages such as C?**

Through the activity, I learned that one major difference between Python and C is how they handle the starting point of a program. In C, the program begins from the int main() function, which is mandatory and specifies an integer return type. In contrast, Python doesn't require a main() function; the code runs sequentially from top to bottom, making it simpler for smaller scripts. Although Python allows you to define a main() function, it is not necessary for the program to execute.

2. **How does variable assignment differ in Python compared with other languages such as C?**

From the activity, I learned that one major difference between Python and C in terms of variable assignment is the need for explicit type declaration in C. In C, you must declare the type of a variable before using it, such as int x = 10;, where the type is fixed. In contrast, Python is dynamically typed, so you don't need to specify a variable's type. You can simply write x = 10, and Python will automatically assign the correct type to the variable. This flexibility in Python allows variables to change types during the program's execution, unlike in C, where the type remains constant once declared.

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?**

From the activity, I learned that Python does not accept variable names that start with numbers. For example, trying to assign 1variable = 10 will result in a syntax error. Additionally, Python restricts the use of most special characters in variable names, allowing only underscores (_) to be used. Special characters like @, #, or ! are not permitted. However, an exception is that underscores are commonly used in variable names, such as my_variable, and they are perfectly valid in Python.

4. **Do the assignment operators (+, -, *, /, %, **) work for all data types? Why or Why not?**

In the activity, I learned that assignment operators like +, -, *, /, %, and ** don't work for all data types in Python. For example, when trying to use the + operator to combine two dictionaries like value = {"name": "Royce"} and value2 = {"age": 2}, it resulted in an error. This is because the + operator isn't defined for dictionaries in Python—unlike strings or lists, which can be concatenated using +. Therefore, while these operators work for certain data types like numbers and strings, they aren't universally applicable to all data types, such as dictionaries, unless specific methods or functions are defined.

5. **How does the * operator differ from the ** operator?**

The * operator in Python is primarily used for multiplication when applied to numbers, and it can also be used to unpack iterables (like lists or tuples) in function calls. The ** operator, on the other hand, is used for exponentiation, meaning it raises a number to a power (e.g., 2 ** 3 results in 8). Additionally, the ** operator is used for unpacking dictionaries into keyword arguments in functions. So, while * is for multiplication and iterable unpacking, ** is for exponentiaton and dictionary unpacking.

7. Conclusion

In conclusion, through this activity, I have gained a clearer understanding of several key differences between Python and languages like C. I learned that Python doesn't require a main() function and executes code sequentially, unlike C, which mandates the int main() function. Additionally, Python's dynamic typing simplifies variable assignment, as there is no need to declare variable types, which contrasts with C's static type system. I also discovered that Python has specific rules for variable names, prohibiting numbers at the beginning and most special characters, with underscores being the main exception. Furthermore, while assignment operators like +, -, *, /, %, and ** work for certain data types, they are not universally applicable, especially for dictionaries. Lastly, I learned the distinction between the * and ** operators: * is used for multiplication and iterable unpacking, while ** is used for exponentiation and dictionary unpacking, highlighting the versatility of these operators in Python.

8. Assessment Rubric: