

LABORATORY MANUAL

Object-Oriented Programming (CPE 103)

|  |  |
| --- | --- |
| **Laboratory Activity No. 2.1** | |
| **Literals, Operators, and Variables** | |
| **Course Code:** CPE103 | **Program:** BSCPE |
| **Course Title:** Object-Oriented Programming | **Date Performed:** Jan. 25, 2025 |
| **Section:** CPE 1A | **Date Submitted:** Jan. 30, 2025 |
| **Name:** Asis, Cy Jerome B. | **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:   * 1. Assign different values to variables in Python   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](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: | |

|  |
| --- |
| <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.      1. In a new cell, use the same variable **value** and assign it the value of 5.0 then print the value. 2. Repeat step 2.   **Note:** You may choose to decide how you execute the code in the cells for the next tasks in the procedure.   1. Repeat these steps for the following values:    1. 2+3j    2. ‘Hello World’    3. “Hello World”    4. True    5. False    6. [1,2,3,4,5]    7. (1,2,3,4,5)    8. { ‘name’: ‘Your\_name’ }    9. None 2. Re-assign the **value** variable to be equal to 5. 3. 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:   * 1. value, value2 = 5.0, 6   2. value, value2 = -5, 6.1   3. value, value2 = “Hello”, ‘world’   Note: Modify the code so that hello and world would be separated.   * 1. value, value2 = [1,2,3], [4,5,6]   2. value, value2 = (1,2,3), (4,5,6)   3. value, value2 = {"name":"Royce"}, {"age":2}   Note: Observe the outputs carefully and try repeating them using subtraction. |
| 1. Using value, value2 = 30, 4. Type the commands:    1. print(value\*value2)    2. print(value2\*\*2)    3. print(value2\*\*3)    4. print(value\*value2+value2\*\*2+1)    5. print(value/value2)    6. 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 = 𝑤𝑒𝑖𝑔ℎ𝑡   ℎ𝑒𝑖𝑔ℎ𝑡2   1. 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().   1. 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);  }   1. **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**   1. **Write a program that can determine the distance between two points given the coordinates using the formula:**   𝒅 = √(𝒙𝟐 − 𝒙𝟏)𝟐 + (𝒚𝟐 − 𝒚𝟏)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**  **Questions:**   * 1. **Give one major difference in syntax that Python has with other languages such as C?**   C divides integers unless specifically instructed to do so, but Python handles floating-point division by default. Python eliminates boilerplate code by not requiring a main () function like C does.   * 1. **How does variable assignment differ in Python compared with other languages such as C?**   While variables in C must be declared with a specific type before being used, variable assignment in Python is dynamic and does not require explicit type declarations.   * 1. **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?**   Variable names in Python cannot begin with a number or contain most special characters, such as @, #, or $. Nevertheless, underscores (\_) are the only permitted special characters, and digits can be used after the first character.   * 1. **Do the assignment operators (+, -, \*, /, %, \*\*) work for all data types? Why or Why not?**   No, not all Python data types can use assignment operators like +, -, \*, /, %, and \*\*. They may not function as intended or even be applicable for other data types, although they are compatible with the majority of numeric types (such as int, float, and complex).   * 1. **How does the \* operator differ from the \*\* operator?**   \* is employed for unpacking, multiplication, and repetition (in the case of sequences). \*\* is used to unpack dictionaries into function arguments and to exponentiate (for numbers). |

|  |
| --- |
|  |
| **7. Conclusion:** In conclusion, Python has a more readable and versatile syntax than C. Python employs dynamic typing instead of explicit type declarations like C does, which makes coding easier. With the exception of underscores, which have specialized purposes, Python does not permit variable names to begin with numbers or contain the majority of special characters. Python places a high value on readability and simplicity, making it both user-friendly for novices and robust for developers. |
|  |
| **8. Assessment Rubric:** |