REPUBLIC OF THE PHILIPPINES
POLYTECHNIC UNIVERSITY OF THE PHILIPPINES
STA. MESA, MANILA

COLLEGE OF ENGINEERING

ELECTRONICS ENGINEERING DEPARTMENT

CMPE 20022:

COMPUTER PROGRAMMING

INSTRUCTIONAL MATERIAL

MODULE 2 – LAB ACTIVITY

Module 2

Laboratory Activity # 1

Objectives

- becoming familiar with the concept of storing and working with different data types in Python;
- experimenting with Python code.

Scenario

Here is a short story:

Once upon a time in Appleland, John had three apples, Mary had five apples, and Adam had six apples. They were all very happy and lived for a long time. End of story.

Your task is to:

- create the variables: `john`, `mary`, and `adam`;
- assign values to the variables. The values must be equal to the numbers of fruit possessed by John, Mary, and Adam respectively;
- having stored the numbers in the variables, print the variables on one line, and separate each of them with a comma;
- now create a new variable named `totalApples` equal to addition of the three former variables.
- print the value stored in `totalApples` to the console;
- **experiment with your code**: create new variables, assign different values to them, and perform various arithmetic operations on them (e.g., +, -, *, /, //, etc.). Try to print a string and an integer together on one line, e.g., `"Total number of apples:"` and `totalApples`.

Laboratory Activity # 2

Objectives

- becoming familiar with the concept of, and working with, variables;
- performing basic computations and conversions;
- experimenting with Python code.

Scenario

Miles and kilometers are units of length or distance.

Bearing in mind that `1` mile is equal to approximately `1.61` kilometers, complete the program in the editor so that it converts:

- miles to kilometers;
- kilometers to miles.

Do not change anything in the existing code. Write your code in the places indicated by `###`. Test your program with the data we've provided in the source code.

```
1  kilometers = 12.25
2  miles = 7.38
3
4  miles_to_kilometers = ###
5  kilometers_to_miles = ###
6
7  print(miles, "miles is", round(miles_to_kilometers, 2), "kilometers")
8  print(kilometers, "kilometers is", round(kilometers_to_miles, 2), "miles")
```

Pay particular attention to what is going on inside the `print()` function. Analyze how we provide multiple arguments to the function, and how we output the expected data.

Note that some of the arguments inside the `print()` function are strings (e.g., `"miles is"`, whereas some other are variables (e.g., `miles`).

TIP

There's one more interesting thing happening there. Can you see another function inside the `print()` function? It's the `round()` function. Its job is to round the outputted result to the number of decimal places specified in the parentheses, and return a float (inside the `round()` function you can find the variable name, a comma, and the number of decimal places we're aiming for). We're going to talk about functions very soon, so don't worry that everything may not be fully clear yet. We just want to spark your curiosity.

After completing the lab, open Sandbox, and experiment more. Try to write different converters, e.g., a USD to EUR converter, a temperature converter, etc. - let your imagination fly! Try to output the results by combining strings and variables. Try to use and experiment with the `round()` function to round your results to one, two, or three decimal places. Check out what happens if you don't provide any number of digits. Remember to test your programs.

Experiment, draw conclusions, and learn. Be curious.

Expected output

```
7.38 miles is 11.88 kilometers
12.25 kilometers is 7.61 miles
```

output

Laboratory Activity # 3

Objectives

- becoming familiar with the concept of numbers, operators, and arithmetic operations in Python;
- performing basic calculations.

Scenario

Take a look at the code in the editor: it reads a `float` value, puts it into a variable named `x`, and prints the value of a variable named `y`. Your task is to complete the code in order to evaluate the following expression:

$3x^3 - 2x^2 + 3x - 1$

The result should be assigned to `y`.

```
x =  # hardcode your test data here
x = float(x)
# write your code here
print("y =", y)
```

Remember that classical algebraic notation likes to omit the multiplication operator - you need to use it explicitly. Note how we change data type to make sure that `x` is of type `float`.

Keep your code clean and readable, and test it using the data we've provided, each time assigning it to the `x` variable (by hardcoding it). Don't be discouraged by any initial failures. Be persistent and inquisitive.

Test Data

Sample input

```
x = 0
x = 1
x = -1
```

Expected Output

```
y = -1.0
y = 3.0
y = -9.0
```

output

Laboratory Activity # 4

Objectives

- becoming familiar with the concept of comments in Python;
- using and not using comments;
- replacing comments with code;
- experimenting with Python code.

Scenario

The code in the editor contains comments. Try to improve it: add or remove comments where you find it appropriate (yes, sometimes removing a comment can make the code more readable), and change variable names where you think this will improve code comprehension.

NOTE

Comments are very important. They are used not only to make your programs **easier to understand**, but also to **disable those pieces of code that are currently not needed** (e.g., when you need to test some parts of your code only, and ignore other). Good programmers **describe** each important piece of code, and give **self-commenting names** to variables, as sometimes it is simply much better to leave information in the code.

It's good to use **readable** variable names, and sometimes it's better to **divide your code** into named pieces (e.g., functions). In some situations, it's a good idea to write the steps of computations in a clearer way.

One more thing: it may happen that a comment contains a wrong or incorrect piece of information - you should never do that on purpose!

```
1  #this program computes the number of seconds in a given number of hours
2  # this program has been written two days ago
3
4  a = 2 # number of hours
5  seconds = 3600 # number of seconds in 1 hour
6
7  print("Hours: ", a) #printing the number of hours
8  # print("Seconds in Hours: ", a * seconds) # printing the number of seconds in a given number of hours
9
10 #here we should also print "Goodbye", but a programmer didn't have time to write any code
11 #this is the end of the program that computes the number of seconds in 3 hour
```

Laboratory Activity # 5

Objectives

- becoming familiar with the concept of numbers, operators and arithmetic operations in Python;
- understanding the precedence and associativity of Python operators, as well as the proper use of parentheses.

Scenario

Your task is to complete the code in order to evaluate the following expression:

```
x = float(input("Enter value for x: "))

# put your code here

print("y =", y)
```

$$\cfrac{1}{x+\cfrac{1}{x+\cfrac{1}{x+\cfrac{1}{x}}}}$$

The result should be assigned to `y`. Be careful - watch the operators and keep their priorities in mind. Don't hesitate to use as many parentheses as you need.

You can use additional variables to shorten the expression (but it's not necessary). Test your code carefully.

Test Data

Sample input: `1`

Expected output:

```
y = 0.6000000000000001
```

Sample input: `10`

Expected output:

```
y = 0.09901951266867294
```

Sample input: `100`

Expected output:

```
y = 0.009999000199950014
```

Sample input: `-5`

Expected output:

```
y = -0.19258202567760344
```

Laboratory Activity # 6

Objectives

- improving the ability to use numbers, operators, and arithmetic operations in Python;
- using the `print()` function's formatting capabilities;
- learning to express everyday-life phenomena in terms of programming language.

Scenario

Your task is to prepare a simple code able to evaluate the **end time** of a period of time, given as a number of minutes (it could be arbitrarily large). The start time is given as a pair of hours (0..23) and minutes (0..59). The result has to be printed to the console.

```
1  hour = int(input("Starting time (hours): "))
2  mins = int(input("Starting time (minutes): "))
3  dura = int(input("Event duration (minutes): "))
4
5  # put your code here
```

For example, if an event starts at **12:17** and lasts **59 minutes**, it will end at **13:16**.

Don't worry about any imperfections in your code - it's okay if it accepts an invalid time - the most important thing is that the code produce valid results for valid input data.

Test your code carefully. Hint: using the `%` operator may be the key to success.

Test Data

Sample input:

```
12
17
59
```

Expected output: `13:16`

Sample input:

```
23
58
642
```

Expected output: `10:40`

Sample input:

```
0
1
2939
```

Expected output: `1:0`