CSCE 1035 Lab 02 Fall 2021

In this course you will learn to write Python 3 programs to solve different problems. From time-to-time this task may become frustrating and difficult. However, with patience and practice, you will gain more experience and things will get easier. The difficulty in using a computer to solve problems comes from the fact that you need to somehow ask a computer to do things for you. Imagine, sitting in front of the computer and asking it to add two numbers, as in "Hey, add 2 and 4 and tell me what the answer is." But we do not normally talk directly to a computer (at least, not yet). Instead, we communicate through an operating system (OS). The communication with the OS is done via the program that we write. Our program is a set of instructions that a computer will follow. These instructions are not in English, but we understand them very well, because they are written in a high-level language called Python3. These instructions will, at some point, get to the machine much differently – in machine language. The machine language is a low-level language.

In general, you will go through several steps before you can get an output from your program:

1) First you will use an editor such as `nano` or `vim` to type your program;
2) then you will invoke the Python 3 interpreter that will convert the source code into an intermediate language, which is again translated into the native/machine language that is executed to obtain the output.

The most important part of a programmer's job is solving the problem *first*. It is much harder to solve a problem than to translate your ideas to a specific programming language. Thus, you should first think about a method and develop an algorithm to solve the problem. An algorithm is a sequence of precise instructions, which results in a solution. The key concept here is precision. If your algorithm has ambiguity in it, then you will not get the correct, or expected, result.

1. zyBooks Labs

Please follow the link on Canvas to complete the following zyBooks labs:

- 1.12 *zyLab training: Basics*
- 1.13 *zyLab training: Interleaved input / output*
- 1.18 *LAB: Warm up: Hello world*
- 1.19 *LAB: Warm up: Basic output with variables*
- 1.20 *Lab: Program: ASCII art*

2. Writing a Simple Python3 Program

Write a small, but complete Python3 program called **Lab2A.py** that calculates the surface area of a cylinder as follows:

a. Assign the value 3.14159 to a variable called **`PI`**.

b. Prompt for and read in a floating-point number variable for the radius of the cylinder in inches and store in the variable **`radius`**.

c. Prompt for and read in another floating-point number variable for the height of the cylinder in inches and store in the variable **`height`**.

d. Compute the surface area of the cylinder using the formula:

$$2 * PI * radius * (radius + height)$$

and save the result in the variable named **`surface_area`**.

e. Finally, print out a meaningful statement that provides the user the calculated surface area of the cylinder in square inches.

For example, the output might look like this (input shown in **bold**):

```
$ python3 Lab02A.py
Enter the radius of the cylinder in inches: 2.1
Enter the height of the cylinder in inches: 3.6
The surface area of the cylinder is 75.2096646 square inches
```

Before writing the code, you may want to compute a hand example to verify that your program solution is correct and matches your example. Feel free to reference the Lab 01 programs you worked with last week or the class lecture notes to see how you might accomplish this. *Note that you will submit this file to Canvas.*

Now that you have completed this lab, it's time to turn in your results. Once you've moved the files to your windows machine (using **WinSCP**), you may use the browser to submit them to Canvas for the **Lab 02** dropbox.

You should submit the following files:

- **`Lab2A.py`**
- **(Note that the zyBooks labs are submitted separately through Canvas.)**

Ask your TA to check your results before submission.

Now that you've finished the lab, use any additional time to practice writing simple programs out of the textbook, lectures, or even ones you come up with on your own to gain some more experience.