CMSC 245 – Principles of Programming Languages
Lab#1: Introducing Python

In this lab you will install the Python programming environment (IDLE) as well as the programming language assigned to you on your computer and begin your journey into learning these languages. First, you are directed to the Official Python website to download and install the Python programming environment on your computer. You do not have to do this if you are comfortable using the computer science lab Linux machines. To fire up IDLE on a Linux computer in the CS Labs, just use the command "idle". You will also install an implementation of the programming language assigned to you and use it in PART 4.

PART 1: Installing Python & IDLE

Go to the website: https://www.python.org/ download and run the installer for your computer. You may need to restart your computer.

Test your install by starting IDLE. An interactive Python window will pop-up showing you the version of Python (make sure it is 3.1x).

PART 2: A First Python Program – Hello, World!

Create a new directory for storing all your Python programs.

In IDLE Code, start a new file (under the File Menu in IDLE) and enter the following program, as shown below:

```
print("Hello, world!")
```

Save the program in a file called `hello.py` in the directory you created above.

Next, Run the program by selecting the Run Module option in the Run menu of the code window. The program will be loaded into the Python interpreter (IDLE Window) and the command will be executed. You should see the following:

```
Hello, World!
```

That's it. Your first Python program!

PART 3: A Second Program.

Enter and run the following program:

```
1  import math
2
3  for n in range(1, 11):
4    print(f"Square root of {n} is {math.sqrt(n)}")
```

Observe the program carefully. Line numbers are not part of the program. They have been added to locate lines in the discussion below. Here are some salient features:

- **Line 1:** `math` is a standard Python package and we get the `sqrt()` function from it.

- **Lines 3 -4:** Define a for-loop. n is the loop control variable. `range(1, 11)` produces values in the range [1, 11) (not including 11) and assigns them to n.
- **Line 4:** Uses the formatted `printf()` function. Formatted `print()` is very similar to the C/Java `printf()`. It takes a format string

  ```
  (f"Square root of {n} is {math.sqrt(n)}")
  ```

  The string specifies exactly how the print will occur in the output. Except, all expressions inside the curly braces ({…}) are evaluated and their results are inserted.

  We use the `math.sqrt()` function to compute the square root of n.

Save the program in a file sqRoot.py. As above, run the program to see the output. Try changing the range command to the following:

- range(5)
- range(1,11, 2)
- range(10,1,-1)

Variables and while-loops

As you can see from above, there are no variable declarations in Python. In Python, variables type bindings can change dynamically (this is dangerous but is the way Python is designed). For example, the following are legal statements in Python:

```
# a as an integer
a = 42
print(a)
# a as a floating point
a = 1.414
print(a)
```

Python also has while-loops:

```
while <condition>:
      <statements to be repeated>
```

For example, the loop in the `sqRoot.go` program above can be re-written as:

```
n = 1
while n <= 10:
      print(f"Square root of {n} is {math.sqrt(n)}")
      n = n + 1
```

Rewrite and run the `sqRoot.py` program above to use this loop.

PART 4: Your Programming Language.

Install an implementation of the programming language assigned to you just as you did for Python. Repeat PART 2 and PART 3 in your programming language. Observe the differences.