Lab 3a: Python I - Variables, Modules, and the rone API

1 Preliminaries

Remember, computers can only do three things:

1. **Processing:** The central processing unit (CPU) can add, subtract, multiply, and logical operations, like "this **and** that". That's it.
2. **Read and write to memory:** Memory is just a long list of numbers, each with an address. The CPU can read and write to memory. Your robot can remember 65536 numbers.
3. **Input/Output:** The CPU can load data from memory and send it to the outside world, and vice-versa. This is how you turn lights and motors on, and read the sensors on your robot.

2 Interactive Prompt

Let's get going with Python! We'll start with the interactive prompt, it's a good way to get started. Everything you type, the computer will try to do.

3 Variables and Assignment

A variable can be viewed as simply a name given to a memory location. For example:

```
i = 5
```

This names some location in memory "`i`", and writes the value 5 to that location. The "=" isn't equality, it's assignment. You can change the value of `i` at any time. For example:

```
i = i + 1
```

This takes the previous value of `i`, adds one to it, and stores it back in the same memory location. This is useful when you want the computer to do something for a certain number of times, like in a loop.

4 Math

We can use your robot as a calculator:

```
i = 5
j = 6
k = i + j
```

wow. Addition.

```
k = i * j
```

wow. Multiplication.

```
k = i / j
```

wow. Divis... Wait a minute, what happened? `k` should be 0.833, but we got 0. Turns out that Python has two different types of numbers: *Ints* (Integers) and *Floats* (real-valued numbers). Let's try the same calculation with Floats:

```
i = 5.0
j = 6.0
k = i + j
```

ok, Addition.

```
k = i / j
```

much better. We'll talk more about the *Types* of variables next time.

5 Functions

A function let's you do complex things by hiding the implementation. Functions are just like block diagrams: Their operation is *abstracted*, so you don't know what goes on inside. They have clearly defined inputs and outputs. You don't care *how* they work, just *what* they do. Today, we will use functions. Next time, we will make some.

5.1 Modules: `import` and `dir()`

A module is a collection of functions. All wrapped up together. You have to import modules into memory before you can use them. If you want to see what you have in a module, use the `dir()` function:

```
import rone

dir(rone)
```

This list of functions is called an *Application programming interface*, or API for short.

5.2 the rone API

Once you have imported a module, you can use functions from it with the modulename.function syntax:

```
import rone

rone.led_set_group('g',40)
rone.motor_set_pwm('r',80)
```

5.3 sys

Another useful module is `sys`:

```
import sys

sys.time()
sys.sleep(1000)
sys.time()
```

6 Programs

Typing at the command line is only useful for the simpliest tasks. More complex tasks will require hundreds of liens of code, far too many to type at the command line. We will put these lines of code into a file called a *program*. Download our sample program for blinking lights: `00-LEDMotorTest.py` from the website. It contains all the techniques from this activity, plus a *while* loop. This loop causes a set of lines of code to be run forever. We will introduce the while loop formally next time, but the rest of the program should make sense. Connect to the robot, and run the program with the command:

```
pyhthon> run 00-LEDMotorTest.py
```

This transfers the program from your computer to the robot, and then runs it. Neat.

We are done with this tutorial. Have fun programming your robots!