Python Basics

1 Variables, Types, and `print`

A variable can be viewed as 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. you can change this value 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 for loops.)

Let's make some variables! We use the `print` *statement* to print values from a running program. This is useful for seeing what is going on inside that pretty little robot head.

```
a = 1
b = 2
c = 3

print a, b, c

print 'all done'
```

2 Functions: `def`

Functions let you write code once, then use is many times. Let's make two functions:

```
def add(x, y):
    val = x + y
    return val

def mult(x, y):
    val = x * y
    return val

x = add(3, 4)
y = mult(3, 4)

print x, y
```

Just like in math, a function takes *arguments* as inputs and *returns* a value as an output. We name the arguments in the definition of the function. A function does not run when you define it, it needs to be *called* from another function, or by typing its name at the prompt. When you call a function, you need to supply the arguments.

Note that we can use the same argument names in two different functions. The *scope* of the arguments is limited to the function they are used in. How does Python know the scope? It looks at the indentation of your code. Things that are indented belong to the scope of the first non-indented line above it. We will talk more about scope later this week.