

# print vs return in a Function

## 1. Defining your function

When you write a function, there are two general things your function can do:

- First, it can do some computations, maybe print stuff out, and return a value to whoever called it. The difference between this and the next example is the extra return line. Example:

```
def b(x):  
    x = x + 3  
    print 'b of x is', x  
    return x
```

- Second, it can do some computations, maybe print some stuff out. Example:

```
def a(x):  
    x = x + 2  
    print 'a of x is', x
```

However, in Python, every function must return something. So if your function does not have an explicit line that returns a value (as the one above), the behavior of Python is to insert at run-time a line that says `return None`. This value `None` has a type `NoneType` (just like the value `10` has a type `int`). So the above function (and any other that does not have a return statement) can be rewritten as:

```
def a(x):  
    x = x + 2  
    print 'a of x is', x  
    return None
```

## 2. How to call the functions you defined?

You just type their names and then give them some parameters. The confusion lies when you combine calling the function along with using print. So below I will illustrate the 4 different combinations of functions (with or without a return) and calling (with or without print).

- **No return and no print**

```
x = 1
a(x)
|---> prints out "a of 1 is 3" (what the function prints)
```

- **No return and yes print**

```
x = 1
print a(x)
|---> prints out "a of 1 is 3" (what the function prints)
|---> prints out "None"
```

When we called `a(x)`, the function returned `None`. In the first example, we didn't do anything with the return value. In the second, example, we printed the return value, which is why the `None` also got printed.

- **Yes return and no print**

```
x = 1
b(x)
|---> prints out "b of 1 is 4" (what the function prints)
```

While this function returns some value, we don't do anything with the value.

- **Yes return and yes print**

```
x = 1
print b(x)
|---> prints out "b of 1 is 4" (what the function prints)
|---> prints out "4"
```

When we called `b(x)`, the function returned the new `x` value. Now, we are printing the return value, so we also print 4.