Returning values from functions

Functions may return *one* value to the place from which they were called. This value may be a literal, a variable, or a mathematical expression (anything that normally may go on the right side of an assignment statement).

```
def name_of_function(param1, param2, ...):
   statement
   statement
   [ more statements if you want ]
   return value
   # in real code, replace "value" above with a variable, literal, or math
```

The only new syntax here is the **`return`** keyword. Whenever Python encounters a line of code that says "**`return`** something," the function immediately ends, and the "something" is sent back to the place where the function was called.

Capturing the return value

When you call a function that returns a value, if you want to use that value later (as you probably do), you need to "capture" it. The easiest way to do this is to use a variable assignment statement:

```
some_variable = name_of_function(arg1, arg2, ...)
```

Whenever Python sees a line like the one above, Python calls the function as it normally would, but when the function returns its value (whatever that value is), it is saved into the variable `some_variable`. The end result is now the code that called the function can use the value that the function calculated, because you have your own copy of it now.

Examples:

```
def square(x):
   return x * x

def larger(a, b):
  if a > b:
    return a
  else:
    return b

def main():
  num = int(input("Give me a number: "))
  num_sq = square(num)                  # Call square, capture return value in num_sq
  larger_one = larger(num, num_squared) # Call larger, capture return value in larger_one
  print("Number is", num)
  print("Number squared is", num_sq)
  print("The larger one is", larger_one)
```