THE RETURN STATEMENT

The return statement is used to **return** from a function i.e. break out of the function.

We can optionally **return a value** from the function as well.

Example (save as function_return.py):

```
def maximum(x, y):
    if x > y:
        return x
    elif x == y:
        return 'The numbers are equal'
    else:
        return y


print maximum(2, 3)
```

Output:

```
$ python function_return.py
3
```

How It Works

The maximum function returns the maximum of the parameters, in this case the numbers supplied to the function. It uses a simple if..else statement to find the greater value and then **returns** that value.

Note that a return statement without a value is equivalent to return None. None is a special type in Python that represents nothingness. For example, it is used to indicate that a variable has no value if it has a value of None.

Every function implicitly contains a return None statement at the end unless you have written your own return statement. You can see this by running print some_function() where the function some_function does not use there turn statement such as:

```
def some_function():
    pass
```

The pass statement is used in Python to indicate an empty block of statements.

TIP	There is a built-in function called max that already implements the 'find maximum' functionality, so use this built-in function whenever possible.

VarArgs parameters

Sometimes you might want to define a function that can take *any* number of parameters, i.e. **var**iable number of**arg**uments, this can be achieved by using the stars (save as `function_varargs.py`):

```
def total(initial=5, *numbers, **keywords):
    count = initial
    for number in numbers:
        count += number
    for key in keywords:
        count += keywords[key]
    return count


print total(10, 1, 2, 3, vegetables=50, fruits=100)
```

Output:

```
$ python function_varargs.py
166
```

How It Works

When we declare a starred parameter such as `*param`, then all the positional arguments from that point till the end are collected as a tuple called 'param'.

Similarly, when we declare a double-starred parameter such as `**param`, then all the keyword arguments from that point till the end are collected as a dictionary called 'param'.

Source: http://www.swaroopch.com/notes/python/