# THE GLOBAL STATEMENT

If you want to assign a value to a name defined at the top level of the program (i.e. not inside any kind of scope such as functions or classes), then you have to tell Python that the name is not local, but it is **global**. We do this using the global statement. It is impossible to assign a value to a variable defined outside a function without the global statement.

You can use the values of such variables defined outside the function (assuming there is no variable with the same name within the function). However, this is not encouraged and should be avoided since it becomes unclear to the reader of the program as to where that variable's definition is. Using the global statement makes it amply clear that the variable is defined in an outermost block.

**Example (save as function_global.py):**

```
x = 50
def func():
    global x
    print 'x is', x
    x = 2
```

```
    print 'Changed global x to', x

func()

print 'Value of x is', x
```

**Output:**

```
$ python function_global.py

x is 50

Changed global x to 2

Value of x is 2
```

**How It Works**

The `global` statement is used to declare that `x` is a global variable - hence, when we assign a value to `x` inside the function, that change is reflected when we use the value of `x` in the main block.

You can specify more than one global variable using the same `global` statement e.g. `global x, y, z`.

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