LOCAL VARIABLES

When you declare variables inside a function definition, they are not related in any way to other variables with the same names used outside the function - i.e. variable names are **local** to the function. This is called the **scope** of the variable. All variables have the scope of the block they are declared in starting from the point of definition of the name.

Example (save as function_local.py):

```
x = 50


def func(x):
    print 'x is', x
    x = 2
    print 'Changed local x to', x


func(x)
print 'x is still', x
```

Output:

```
$ python function_local.py
x is 50
Changed local x to 2
x is still 50
```

How It Works

The first time that we print the **value** of the name **x** with the first line in the function's body, Python uses the value of the parameter declared in the main block, above the function definition.

Next, we assign the value 2 to x. The name x is local to our function. So, when we change the value of x in the function, the x defined in the main block remains unaffected.

With the last print statement, we display the value of x as defined in the main block, thereby confirming that it is actually unaffected by the local assignment within the previously called function.

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