Scope of variables, Tabulating a function, lambda functions

Scope of variables

Let us study scopes of variables to a slight more detail.

The LGB rule : Python searches for variable names in the following order :

1. First it looks at the local variable names.
2. Then it searches for the variale in the list of global variables.
3. Finally it tries to see if there is some builtin function having the same name.

In [1]:

```
# An example from the book :
print "builtin sum = ", sum # Here no local or global variable is defined, so it takes the inbuilt sum.
sum = 12             # Assigning 12 to sum. This is a global variable for this code.
print "global sum = ", sum

# To see local variables we have to define a function :
def somefn(n) :
    sum = n + 1   # Here it is a local variable. The global variable is untouched.
    print "local sum = ", sum
    return sum

print "whuch sum am I?", sum
sum = somefn(4)
print "New global sum = ", sum
somefn(12)
print "Global sum = ", sum
```

```
builtin sum =  <built-in function sum>
global sum =  12
whuch sum am I? 12
local sum =  5
New global sum =  5
local sum =  13
Global sum =  5
```

In [2]:

```
del sum
```

Look at what the keyword global does

In [3]:

```
# An example from the book :
print "builtin sum = ", sum # Here no local or global variable is defined, so it takes the inbuilt sum.
sum = 12            # Assigning 12 to sum. This is a global variable for this code.
print "global sum = ", sum

# To see local variables we have to define a function :
def somefn(n) :
    global sum
    sum = n + 1   # Here it is a local variable. The global variable is untouched.
    print "local sum = ", sum
    return sum

print "whuch sum am I?", sum
sum = somefn(4)
print "New global sum = ", sum
somefn(12)
print "global sum = ", sum
```

```
builtin sum =  <built-in function sum>
global sum =  12
whuch sum am I? 12
local sum =  5
New global sum =  5
local sum =  13
global sum =  13
```

Can functions access a global variable without using the keyword global? (Or of what use is a global variable if you cannot access it everywhere?)

The answer is that you can access it everywhere. Without using global you cannot modify it. Here is an example

In [4]:

```
globvar = 4

def fun_just_accessing_globvar (n) :
    x = globvar
    print "fun_just_accessing_globvar: 1 : ", x
    x = n
    print "fun_just_accessing_globvar: 2 : ", x
    return globvar

def fun_using_loc_var_globvar (n) :
    globvar = n
    print "fun_using_loc_var_globvar :", globvar
    return globvar

def fun_modifying_globvar (n) :
    global globvar
```

```
    print "fun_modifying_globvar: 1 : ", globvar
    globvar = n
    print "fun_modifying_globvar: 2 : ", globvar
    return globvar

fun_just_accessing_globvar(100)
print "After fun_just_accessing_globvar, globvar =", globvar

fun_modifying_globvar(200)
print "After fun_modifying_globvar, globvar =", globvar

fun_using_loc_var_globvar(30)
print "After fun_using_loc_var_globvar, globvar =", globvar

fun_just_accessing_globvar(300)
print "After fun_just_accessing_globvar, globvar =", globvar

fun_modifying_globvar(300)
print "After fun_modifying_globvar, globvar =", globvar
```

```
fun_just_accessing_globvar: 1 :  4
fun_just_accessing_globvar: 2 :  100
After fun_just_accessing_globvar, globvar = 4
fun_modifying_globvar: 1 :  4
fun_modifying_globvar: 2 :  200
After fun_modifying_globvar, globvar = 200
fun_using_loc_var_globvar : 30
After fun_using_loc_var_globvar, globvar = 200
fun_just_accessing_globvar: 1 :  200
fun_just_accessing_globvar: 2 :  300
After fun_just_accessing_globvar, globvar = 200
fun_modifying_globvar: 1 :  200
fun_modifying_globvar: 2 :  300
After fun_modifying_globvar, globvar = 300
```

Exercise : Understand the output above by printing/copying tha above program and marking the globvar's occuring there as local or global.

Another example :

In [5]:

```
x = 100

def print_glob_x () :
    print "print_glob_x : Global x = ", x



def fn_loc_x (n) :
    x = n
    print "fn_loc_x : Local x : ", x
    print_glob_x()
```

```
def fn_glob_x (n) :
    global x
    x = n
    print "fn_glob_x : Local x : ", x
    print_glob_x()

print "Global x = ", x
fn_loc_x(200)
print "Global x = ", x
fn_glob_x(200)
print "Global x = ", x
```

```
Global x =  100
fn_loc_x : Local x :  200
print_glob_x : Global x =  100
Global x =  100
fn_glob_x : Local x :  200
print_glob_x : Global x =  200
Global x =  200
```

Informal exercise : Experiment with variables as much as you can

Tabulating functions

In python one can pass functions to other functions, just as one passes variables

In [6]:

```
def tabulate(fn, x_lo, x_hi, step) :
    a = x_lo
    while a <= x_hi :
        print "%f\t\t%f" %(a, fn(a))
        a += step
```

In [7]:

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

In [8]:

```
tabulate(square, 0, 10, 1)
```

```
0.000000                0.000000
1.000000                1.000000
2.000000                4.000000
3.000000                9.000000
4.000000                16.000000
5.000000                25.000000
6.000000                36.000000
7.000000                49.000000
8.000000                64.000000
9.000000                81.000000
10.000000               100.000000
```

Lambda functions

In [9]:
```
tabulate(lambda y : y*y, 0, 10, 1)
```

```
0.000000                0.000000
1.000000                1.000000
2.000000                4.000000
3.000000                9.000000
4.000000                16.000000
5.000000                25.000000
6.000000                36.000000
7.000000                49.000000
8.000000                64.000000
9.000000                81.000000
10.000000               100.000000
```

Most basic plotting

In [10]:
```
def very_simple_plot(f, xlow, xhigh, xscale, yscale, yoffset) :
    number_of_lines = int(float(xhigh - xlow)/xscale)
    if number_of_lines < 0 :
        temp = xlow
        xlow = xhigh
        xhigh = temp

    for i in range(number_of_lines) :
        xi = xlow + i * xscale
        yi = (yoffset + f(xi)) * yscale
        ith_line = ""
        for j in range(int(yi) - 1) :
            ith_line += " "
        ith_line += "*"
        print ith_line
```

In [11]:
```
very_simple_plot(lambda x : x*x, 0, 1, .1, 50, 0)
```

```
*
*
 *
   *
       *
           *
                 *
                       *
                               *
```

```
                                       *
```

In [12]:

```
from math import sin
```

In [13]:

```
very_simple_plot(sin, 0, 14, .2, 10, 2)
```

```
                   *
                    *
                      *
                        *
                          *
                           *
                            *
                            *
                            *
                            *
                            *
                           *
                         *
                        *
                      *
                    *
                  *
                *
              *
            *
           *
          *
         *
         *
         *
         *
          *
           *
            *
              *
                *
                  *
                    *
                      *
                       *
                         *
                          *
                           *
                            *
                            *
                            *
                            *
                           *
                          *
```

```
                        *
                       *
                     *
                   *
                 *
               *
             *
            *
          *
         *
         *
         *
         *
         *
          *
            *
             *
               *
                 *
                   *
                     *
                       *
                        *
                          *
                           *
                            *
```

In [14]:

```python
def very_simple_plot_with_x_axis(f, xlow, xhigh, xscale, yscale, yoffset) :
    number_of_lines = int(float(xhigh - xlow)/xscale)
    if number_of_lines < 0 :
        temp = xlow
        xlow = xhigh
        xhigh = temp

    for i in range(number_of_lines) :
        xi = xlow + i * xscale
        yi = (yoffset + f(xi)) * yscale
        ith_line = ""
        # x-axis is y = 0 :
        if yoffset * yscale < int(yi) :
            for j in range(1, int(yi)) :
                if j == yoffset * yscale :
                    ith_line += "*"
                else :
                    ith_line += " "
            ith_line += "*"
        else :
            for j in range(1, yoffset * yscale) :
```

```
                if j == int(yi) :
                    ith_line += "*"
                else :
                    ith_line += " "
            ith_line += "*"
        print ith_line
```

In [15]:

```
very_simple_plot_with_x_axis(sin, 0, 14, .2, 20, 2)
```

```
                    *
                    *  *
                    *      *
                    *          *
                    *             *
                    *               *
                    *                 *
                    *                  *
                    *                  *
                    *                  *
                    *                 *
                    *               *
                    *            *
                    *         *
                    *     *
                    * *
                  * *
              *     *
           *        *
       *            *
    *               *
  *                 *
*                   *
*                   *
*                   *
*                   *
  *                 *
    *               *
       *            *
          *         *
              *     *
                  * *
                    * *
                    *     *
                    *        *
                    *            *
                    *              *
                    *                *
                    *                  *
                    *                  *
                    *                  *
                    *                 *
```

```
                    *                *
                    *             *
                    *          *
                    *       *
                    *   *
                    *
                *   *
            *       *
         *          *
      *             *
   *                *
 *                  *
*                   *
*                   *
*                   *
 *                  *
   *                *
      *             *
         *          *
            *       *
                *   *
                    *
                    *   *
                    *       *
                    *          *
                    *             *
                    *                *
                    *                 *
```