

Scientific Programming in Python
Notes
25 September 2009

Functions

A function is a construction that accepts input parameters, manipulates data and returns results. As a first example, we define a polynomial function

$$p(x, y) = 3x^2 + y^2 - xy + 1.$$

In Python, we write this as

```
def p(x,y):  
    return 3*x**2 + y**2 - x*y + 1
```

Note:

- The name of the function is `p`—we will use this later to refer to the function.
- The function definition `def p(x,y)` tells the computer that the function takes two input arguments and that we are going to refer to them as `x` and `y` inside the function.
- We send back the result of the function using the `return` keyword. This causes the function to stop immediately and return the value indicated after the keyword.

Somewhere else in our code we can call this function to compute the function value for particular inputs.

```
>>> print p(2.56,3)  
21.98
```

For a slightly more complex example, we can code the function

$$f(x) = \begin{cases} \frac{x}{2} & \text{if } x \text{ is even} \\ 3x + 1 & \text{if } x \text{ is odd} \end{cases}$$

using

```
def f(x):  
    if x % 2 == 0:  
        return x/2  
    else:  
        return 3*x+1
```

Keyword arguments

The inputs that we pass to a function are called arguments. We distinguish between regular arguments and keyword arguments. Regular arguments are always listed in the order in which they were defined and all of these arguments have to be provided when calling the function. Keyword arguments are specified with a default value when defining the function and are optional when calling the function.

```
from __future__ import division
from scipy import *

def logarithm(x, base=e):
    return log(x) / log(base)

print logarithm(8)
print logarithm(8, 2)
print logarithm(8, base = 2)
```

This produces the output

```
2.0794415416798357
3.0
3.0
```

Note that in the first function call we did not supply the value of **base**, which means that it was set equal to the default value, **e**. The second and third function calls are equivalent—both assign the value 2 to **base**. A function can be defined with any number of arguments and keyword arguments. The non-keyword arguments always have to be listed before the keyword arguments.

Returning values

You can return any value from a function or choose not to return a value. If you do not return anything Python returns the special value **None** for you. If you want to return more than one value, you can use a comma separated list.

```
def sum_and_product(x, y):
    return x+y, x*y

s, p = sum_and_product(3, 5)
print 'Sum:', s
print 'Product:', p
```

This will display

```
Sum: 8
Product: 15
```