Functions**¶**

* Created with def
* Possibly empty list of arguments
* May return a value

In [ ]:

def sum(a,b):  
 return a+b

Positional arguments: should be specified in correct order

In [ ]:

sum(2,3)

Variable args**¶**

How do we support variable number of positional args?

In [ ]:

def f(a,b,\*args):  
 print 'a,b:',a,b  
 print 'Rest', args

In [ ]:

f(1,2,3,4,5)

* \*args 'absorbs' rest of the arguments
* 'args' not a keyword; just a (very well-followed) convention; could be \*rest, \*z, ..
* received as a list
* Should be the last argument (why?)

In [ ]:

Default values**¶**

In [ ]:

def f(a,b=0,\*args):  
 print 'a,b:',a,b  
 print 'Rest', args

* Default value args should be last
* Before \*args (why?)

In [ ]:

f(1)

In [ ]:

f(1,2,3)

Keyword Args**¶**

* Large functions may have a large number of optional args: positional args not a good approach
* Better approach : f(config='cfg.py', log='app.log')
* \*\*kwargs defines keyword args
* should be the last arg in arglist, after any \*args if present
* kwargs received as a dict
* allows support for arbitrary keywords as arguments
* again: kwargs not a keyword; could be \*\*d, ..

In [ ]:

def f(a,b,\*args,\*\*kwargs):  
 print 'a,b', a,b  
 print 'Rest', args  
 print 'Keyword args', kwargs

In [ ]:

f(1,2,config='cfg.py', log='app.log')

In [ ]:

Unfolding args when calling**¶**

In [ ]:

vals = [1,2,3,4]  
keyvals = {'key': 1, 'cmp': None}  
f(\*vals, \*\*keyvals)

In [ ]:

keypos = {'b':1, 'a':2}  
f(\*\*keypos)

But, how about..

In [ ]:

f(\*range(5))

In [ ]:

f(\*xrange(5))

Receiving and passing fwd arbitrary args**¶**

In [ ]:

def wrap(func, \*args, \*\*kwargs):  
 # Do something  
 ret = func(\*args, \*\*kwargs)  
 # Do something else  
 return ret

* Supports any kind of function signature
* Reliably passes fwd received args
* A very commonly seen pattern

Rules**¶**

* Mandatory args must be specified exactly once (positionally, or keyword-ly)
* positional args need to be in order
* keyword args may be specified in any order
* \*args can be the last positional parameter
* Default value args should be last, but before \*args
* \*\*kwargs can be the very last parameter
* \*lval to unfold list (iterable) arg
* \*\*dval to unfold dict arg
* positional parameters not valid after \*args, \*\*kwargs; raise a SyntaxError exception

Is **def** the only way to define functions?

Lambda functions**¶**

Adopted from formal mathematics $$\lambda x \rightarrow x^2$$

* Convenient way for creating functions on the fly
* For passing functions around (as args)
* Does not support code-block; just single line code
* no return needed
* a functional programming feature ==> python supports functional paradigm

In [ ]:

sum = lambda a,b: a+b

This is saying exactly the same thing as:

In [ ]:

def sum(a,b):  
 return a+b

In [ ]:

tuplist = [(9,2), (6,3), (1,-1), (2,1)]

In [ ]:

sorted(tuplist)

In [ ]:

sorted(tuplist, key=lambda x:x[1])

In [ ]:

f = lambda:42

In [ ]:

f()

Q: So, only functions can be called thus: f() ?

A: No. Any object with an attribute/method \_\_call\_\_

More under **class**

Mandatory keyword-only args?**¶**

<https://www.python.org/dev/peps/pep-3102/> [PEP3102]

Supported only in 3.5+