Optional and Keyword Arguments

Python

- An argument may be assigned a default value in the parameter list; if so that argument becomes optional. If not present in the calling list it takes the assigned default value.

```
 def func(x,y=0,w=3):
```

- Keyword arguments can be passed by keyword, not position. They must follow any positional arguments in the argument list.

```
def func(x,y,w):
 z=func(x,w=6,y=2)
```

Warning

- Default values are set only *once*, when the function is compiled to bytecode.
- Avoid using a mutable type as a default value for optional arguments (remember that primitive types are immutable)

Keyword and Variable Lists

- Variable arguments: a function can use an argument of the form *name, which will cause all arguments in that position to be bundled into a tuple and passed to the function.
- Variable keyword lists are specified with **name. They are passed as a dictionary. These arguments are often called **kwargs in documentation.
- If you use both then *name must come before **name

Examples

```
def test_var_args(farg, *args):
   print "formal arg:", farg
   for arg in args:
   print "another arg:", arg test_var_args(1, "two", 3)
test_var_args(1, "two", 3)
-------
def test_var_kwargs(farg, **kwargs):
   print "formal arg:", farg
   for key in kwargs:
   print "another keyword arg: %s: %s" % (key, kwargs[key])
test_var_kwargs(farg=1, myarg2="two", myarg3=3)
```