Python decorators

August 16, 2007 / Gavin Jackson

code programming python

I came across an interesting recent addition to the Python language (as of Python 2.4). They are called decorators. Decorators are Python objects that can register, annotate, and/or wrap a Python function or method. They are primarily used to achieve code reuse. This approach to programming struck me as being very similar to Aspect oriented programming. A decorator is a callable object (like a function) that accepts one argument—the function being decorated. The return value of the decorator replaces the original function definition. One use I found useful, was timing how long function calls actually take:

```
import time

def print_timing(func):

 def wrapper(*arg):
 t1 = time.time()
 res = func(*arg)
 t2 = time.time()
 print '%s took %0.3f ms' % (func.func_name, (t2-t1)*1000.0)
 return res

 return wrapper

# declare the @ decorator just before the function, invokes print_timing()

@print_timing
def myFunction(n):
```

The following page discusses a number of other uses of this pattern (well worth a read).
http://www.phyast.pitt.edu/~micheles/python/documentation.html