Creating a Context Manager using contextlib

Python 2.5 not only added the with statement, but it also added the contextlib module. This allows us to create a context manager using contextlib's contextmanager function as a decorator. Let's try creating a context manager that opens and closes a file after all:

```
from contextlib import contextmanager

@contextmanager
def file_open(path):
    try:
        f_obj = open(path, 'w')
        yield f_obj
    except OSError:
        print("We had an error!")
    finally:
        print('Closing file')
        f_obj.close()

if __name__ == '__main__':
    with file_open('test.txt') as fobj:
        fobj.write('Testing context managers')
```

Here we just import **contextmanager** from **contextlib** and decorate our file_open function with it. This allows us to call file_open using Python's with statement. In our function, we open the file and then yield it out so the calling function can use it.

Once the **with** statement ends, control returns back to the file_open function and it continues with the code following the yield statement. That causes the finally statement to execute, which closes the file. If we happen to have an **OSError** while working with the file, it gets caught and finally statement still closes the file handler.