Creating a Logging Decorator

Sometimes you will want to create a log of what a function is doing. Most of the time, you will probably be doing your logging within the function itself. Occasionally you might want to do it at the function level to get an idea of the flow of the program or perhaps to fulfill some business rules, like auditing. Here's a little decorator that we can use to record any function's name and what it returns:

```
import logging

def log(func):
    """
    Log what function is called
    """
    def wrap_log(*args, **kwargs):
        name = func.__name__
        logger = logging.getLogger(name)
        logger.setLevel(logging.INFO)

        # add file handler
        fh = logging.FileHandler("%s.log" % name)
        fmt = '%(asctime)s - %(name)s - %(levelname)s - %(message)s'
        formatter = logging.Formatter(fmt)
        fh.setFormatter(formatter)
        logger.addHandler(fh)

        logger.info("Running function: %s" % name)
        result = func(*args, **kwargs)
        logger.info("Result: %s" % result)
        return func
    return wrap_log

@log
def double_function(a):
    """
    Double the input parameter
    """
    return a*2

if __name__ == "__main__":
    value = double_function(2)
```

This little script has a **log** function that accepts a function as its sole argument.

It will create a logger object and a log file name based on the name of the

function. Then the log function will log what function was called and what the function returned, if anything.