```python
from __future__ import division

# problem 13
def log(b, a, niter=10):
    """Returns an approximation of log_b(a)."""
    g = lambda x : b**x - a
    x1 = a**(.5)
    x2 = .5*x1
    for i in xrange(niter):
        try:
            x1,x2 = x2, x2 - g(x2)*(x2-x1)/(g(x2)-g(x1))
        except ZeroDivisionError:
            return x2
    return x2

# problem 14
def newtons(x0, e, df, d2f, maxiters=100):
    """Returns the approximate minimizer of f where 'df' and 'd2f' are
    the first and second derivatives, respectively.

    Inputs:
        x0 (float) - initial guess
        e (float) level of accuracy
        df (callable) - derivative of f
        d2f (callable) - second derivative of f
        maxiters (int, optional) - maximum iterations to handle
            divergence. Defaults to 100.
    """
    x1 = x0 - df(x0)/d2f(x0)

    count = 0
    while abs(x1-x0) < x0*e:
        count += 1
        if count > maxiters:
            return x1
        x0,x1 = x1, x0 - df(x0)/d2f(x0)
    return x1

# problem 15
def secant(x0, x1, e, df, maxiters=100):
    """Returns the approximate minimizer of f where 'df' is the first
    derivative.

    Inputs:
        x0 (float) - initial guess #1
        x1 (float) - initial guess #2
        e (float) - accuracy
        df (callable) - derivative of f
        maxiters (int, optional) - maximum iterations to handle
            divergence. Defaults to 100.
    """
    count = 0
    while abs(x1-x0) < x0*e:
```

```
54            count += 1
55            if count > maxiters:
56                return x1
57            x0,x1 = x1, x0 - df(x1)*(x1-x0)/(df(x1)-df(x0))
58        return x1
59
60
```