```
"""Iterative improvement."""

from ucb import main, interact

# Direct implementations

def square_root(a):
    """Return the square root of a.

    >>> square_root(9)
    3.0
    """
    x = 1
    while x * x != a:
        x = square_root_update(x, a)
    return x

def square_root_update(x, a):
    return average(x, a/x)

def average(x, y):
    return (x + y)/2

def cube_root(a):
    """Return the cube root of a.

    >>> cube_root(27)
    3.0
    """
    x = 1
    while pow(x, 3) != a:
        x = cube_root_update(x, a)
    return x

def cube_root_update(x, a):
    return (2*x  + a/(x * x))/3


# Iterative improvement

def golden_update(x):
    return 1/x + 1

def golden_test(x):
    return x * x == x + 1

def iter_improve(update, done, guess=1, max_updates=1000):
    """Iteratively improve guess with update until done returns a true value.

    guess -- An initial guess
    update -- A function from guesses to guesses; updates the guess
    done -- A function from guesses to boolean values; tests if guess is good

    >>> iter_improve(golden_update, golden_test)
    1.618033988749895
    """
    k = 0
    while not done(guess) and k < max_updates:
        guess = update(guess)
        k = k + 1
    return guess

def square_root_improve(a):
    """Return the square root of a.

    >>> square_root_improve(9)
    3.0
    """
    def update(guess):
        return square_root_update(guess, a)
    def done(guess):
        return guess * guess == a
    return iter_improve(update, done)
```

```
def cube_root_improve(a):
    """Return the cube root of a.

    >>> cube_root_improve(27)
    3.0
    """
    def update(guess):
        return cube_root_update(guess, a)
    def done(guess):
        return guess * guess * guess == a
    return iter_improve(update, done)


# Newton's method

def approx_derivative(f, x, delta=1e-5):
    """Return an approximation to the derivative of f at x."""
    df = f(x + delta) - f(x)
    return df/delta

def newton_update(f):
    """Return an update function for f using Newton's method."""
    def update(x):
        return x - f(x) / approx_derivative(f, x)
    return update

def find_root(f, guess=1):
    """Return a guess of a zero of the function f, near guess.

    >>> from math import sin
    >>> find_root(lambda y: sin(y), 3)
    3.141592653589793
    """
    return iter_improve(newton_update(f), lambda x: f(x) == 0, guess)

def square_root_newton(a):
    """Return the square root of a.

    >>> square_root_newton(9)
    3.0
    """
    return find_root(lambda x: x * x - a)

def cube_root_newton(a):
    """Return the cube root of a.

    >>> cube_root_newton(27)
    3.0
    """
    return find_root(lambda x: pow(x, 3) - a)

@main
def run():
    interact()
```