```
import scipy.optimize as optimize
import numpy as np


# **************** problem #1 ******************
def func(x):
    return np.cos(x)**2 + 6 - x


def f(x):
    return np.power(x, 2) + 6.0 * x + 4


def f_prime(x):
    return 2.0 * x + 6.0

# 0<=cos(x)**2<=1, so the root has to be between x=6 and x=7
print(optimize.bisect(func, 6, 7))
6.77609231632

a = optimize.bisect(f, -1, 0)

print("This is the bisection method:{:.16e}".format(a))
# -7.6393202249983005e-01
print("This is the bisection method:{0}".format(a))
# -0.7639320225


# ********************* problem #2 ******************


def g(x):
    return (-4.0) / x + 6.0


def h(x):
    return -1.0 * np.sqrt(-6.0 * x - 4.0)

# This finds the value of x such that func(x) = x, that is, where
# -x**3 + 1 = x
print(optimize.fixed_point(g, -5.2))
# 5.2360679775

print(optimize.fixed_point(h, -0.77))

# ***************** problem #4 ********************
# use the secant method.
# well scipy.optimize.newton() uses the secant method
# if we don’t give it the functions derivative.
```

```
a = 0

a = optimize.newton(f, 0, tol=1e-16, maxiter=8)

# using equation (1) from the assignment. where -7.6 is the fixed point "s"
def convergence(n_plus_one, n):
    numerator = np.log(np.abs(n_plus_one - -7.6393202250021031e-01))
    denominator = np.log(np.abs(n - -7.6393202250021031e-01))
    return numerator / denominator

print("This is the convergence:", convergence(-0.74996874883, -0.66665555574))
print("This is the convergence:", convergence(-0.763636264454, -0.74996874883))
print("This is the convergence:", convergence(-0.763931103843, -0.763636264454))

print("This is secant method answer {:.16e}".format(a))

# **************** problem #5 **********************

# newton() will use the newton raphson method because we gave it the derivative.
a = optimize.newton(f, 0, fprime=f_prime, tol=1e-16, maxiter=14)

print"This is Newton method answer{:.16e}".format(a)


print("This is the convergence:", convergence(-0.761904761905, -0.666666666667))
print("This is the convergence:", convergence(-0.763931104357, -0.761904761905))
print("This is the convergence:", convergence(-0.7639320225, -0.763931104357))

# ************* problem #6 *********************


def k(x):
    return np.power(x, 20) - 10.0


def k_prime(x):
    return 20.0 * np.power(x, 19)


a = optimize.newton(k, 8, fprime=k_prime, tol=1e-16, maxiter=44)

print"This is Newton method answer: {:.16e}".format(a)

# **************** problem #7 ************


def f(x):
    return np.power(x,2) - 6.0*x + 9.0


def f_prime(x):
    return 2.0 * x - 6.0

a = optimize.newton(f, 4, fprime=f_prime, tol=1e-16, maxiter=29)

print"This is Newton method answer: {:.16e}".format(a)

# using equation (1) from the assignment. where -7.6 is the fixed point "s"
def convergence(n_plus_one, n):
    numerator = np.log(np.abs(n_plus_one - 3.0000000298023224e+00))
    denominator = np.log(np.abs(n - 3.0000000298023224e+00))
    return numerator / denominator


print("This is the convergence:", convergence(3.25, 3.5))
print("This is the convergence:", convergence(3.125, 3.25))
print("This is the convergence:", convergence(3.0625, 3.125))

print("This is the convergence:", convergence(3.00000011921, 3.00000023842 ))
```