```
 1: import numpy
 2: import theano.tensor as T
 3: from theano import shared, function
 4: rng = numpy.random
 5:
 6: # 1. Declare Theano variables
 7: x = T.matrix()
 8: y = T.ivector()
 9: w = shared(numpy.random.randn(100))
10: b = shared(numpy.zeros(()))
11: print "Initial model:"
12: print w.get_value(), b.get_value()
13:
14: # 2. Construct Theano expression graph
15: p_1 = 1 / (1 + T.exp(-T.dot(x, w)-b))
16: xent = -y*T.log(p_1) - (1-y)*T.log(1-p_1)
17: prediction = p_1 > 0.5
18: cost = xent.mean() + 0.01*(w**2).sum()
19: gw,gb = T.grad(cost, [w,b])
20:
21: # 3. Compile expressions to functions
22: train = function(
23:             inputs=[x,y],
24:             outputs=[prediction, xent],
25:             updates={w:w-0.1*gw, b:b-0.1*gb})
26: predict = function(inputs=[x], outputs=prediction)
27:
28: # 4. Call Theano functions on numpy ndarrays
29: N = 4
30: feats = 100
31: D = (numpy.random.randn(N, feats),
32:      numpy.random.randint(size=4,low=0, high=2))
33: training_steps = 10
34: for i in range(training_steps):
35:     pred, err = train(D[0], D[1])
36: print "Final model:"
37: print w.get_value(), b.get_value()
38:
39: print "target values for D"
40: print D[1]
41:
42: print "prediction on D"
43: print predict(D[0])
```