```
 1 import numpy
 2 class Percept():
 3     def __init__(self,
   weights, threshold):
 4         self.weights =
   weights
 5         self.threshold =
   threshold
 6         self.inputs = None
 7     def set_inputs(self,
   inputs):
 8         self.inputs =
   inputs
 9     def activiator(self,
   sum):
10         if sum > self.
   threshold:
11             return 1
12         else:
13             return 0
14
15     def multiply(self):
```

```
16          sum = 0
17          for i in range(len
   (self.inputs)):
18              sum+= self.
   inputs[i].input*self.
   weights[i]
19          return sum
20
21      def eval(self):
22          for i in range(len
   (self.inputs)):
23              cool = type(
   self.inputs[i]).__name__
24              if cool is "
   Percept":
25
   actual_inputs = []
26                  for j in
   self.inputs:
27
   actual_inputs.append(j.
   eval())
```

```
28                      num =
   numpy.dot(actual_inputs,
   self.weights)
29                      return
   self.activiator(num)
30                  else:
31                      sum = self
   .multiply()
32                      return
   self.activiator(sum)
33
34              sum = numpy.dot(
   self.inputs,self.weights)
35
36              if sum >= self.
   threshold:
37                  return 1
38              else:
39                  return 0
40 class Input:
41     def __init__(self):
42         self.input = None
```

```
43      def set_value(self,
    input):
44          self.input = input
45
46 x1 = Input()
47 x2 = Input()
48 node_3 = Percept([1,1],1.5
    )
49 node_4 = Percept([1,1],.5)
50 node_5 = Percept([-2,1],.5
    )
51 node_3.set_inputs([x1,x2])
52 node_4.set_inputs([x1,x2])
53 node_5.set_inputs([node_3,
     node_4])
54 xor = node_5
55 for a in range(2):
56     for b in range(2):
57         x1.set_value(a)
58         x2.set_value(b)
59         print(a,b,xor.eval
    ())
```