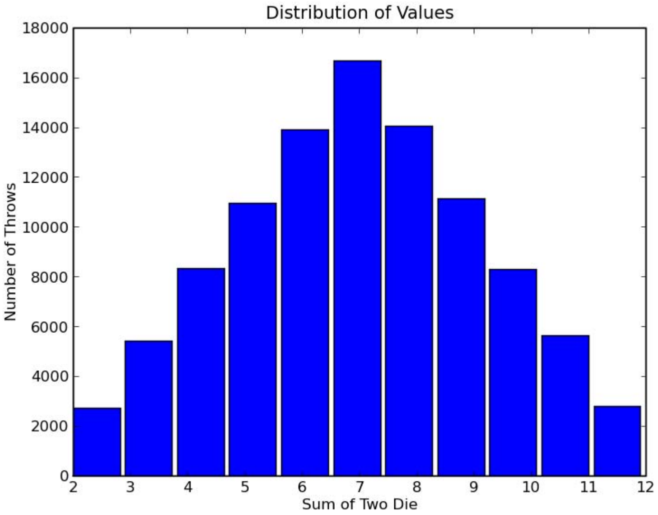
6.00 Introduction to Computer Science and Programming
Fall 2008

6.00 Handout, Lecture 22
(Not intended to make sense outside of lecture)

```
mport random, pylab

fair = [1,2,3,4,5,6]

def throwPair(vals1, vals2):
    d1 = random.choice(vals1)
    d2 = random.choice(vals2)
    return d1, d2


def conductTrials(numThrows, die1, die2):
    throws = []
    for i in range(numThrows):
        d1, d2 = throwPair(die1, die2)
        throws.append(d1+d2)
    return throws

numThrows = 100000
throws = conductTrials(numThrows, fair, fair)
pylab.hist(throws, 11)
pylab.xticks(range(2,13), ['2','3','4','5','6','7','8','9','10','11','12'])
pylab.title('Distribution of Values')
pylab.xlabel('Sum of Two Die')
pylab.ylabel('Number of Throws')

#Get probabilities for fair dice
pylab.figure()
sums = pylab.array([0]*14)
for val in range(2, 13):
    sums[val] = throws.count(val)
probs = sums[2:13]/float(numThrows)
xVals = pylab.arange(2, 13)
pylab.plot(xVals, probs, label='Fair Dice')
pylab.xticks(range(2,13), ['2','3','4','5','6','7','8','9','10','11','12'])
pylab.title('Probability of a Value')
pylab.xlabel('Sum of Two Die')
pylab.ylabel('Probability')

def craps(die1, die2):
    """Return True if shooter wins at craps by betting pass line"""
    d1, d2 = throwPair(die1, die2)
    tot = d1 + d2
    if tot in [7, 11]: return True
    if tot in [2, 3, 12]: return False
    point = tot
    while True:
        d1, d2 = throwPair(fair, fair)
        tot = d1 + d2
        if tot == point: return True
        if tot == 7: return False
```

```
def simCraps(numBets, die1, die2):
    wins, losses = (0, 0)
    for i in range(numBets):
        if craps(die1, die2): wins += 1
        else: losses += 1
    print wins, losses
    houseWin = float(losses)/float(numBets)
    print houseWin
    print 'House winning percentage: '  + str(100*houseWin) + '%'
    print 'House profits per $%d bet: $%d' % (numBets, losses - wins)

simCraps(100000, fair, fair)

#Try some unfair dice
weighted = [1,2,3,4,5,5,6]
throws = conductTrials(numThrows, fair, weighted)
sums = pylab.array([0]*14)
for val in range(2, 13):
    sums[val] = throws.count(val)
probs = sums[2:13]/float(numThrows)
xVals = pylab.arange(2, 13)
pylab.plot(xVals, probs, label = 'Weighted Dice')
pylab.legend()
simCraps(100000, fair, weighted)

pylab.show()
```


Distribution of Values
Number of Throws
Sum of Two Die