Monte Carlo Simulation

Lecturer: John Guttag

Carvaggio, The Cardsharps

```
def rollDie():
    """returns a random int between 1 and 6"""
    return random.choice([1,2,3,4,5,6])

def checkPascal(numTrials, roll):
    yes = 0.0
    for i in range(numTrials):
        for j in range(24):
            d1 = roll()
            d2 = roll()
            if d1 == 6 and d2 == 6:
                yes += 1
                break
    print 'Probability of losing =',\
          1.0 - yes/numTrials
```


3

```
def rollLoadedDie():
    if random.random() < 1.0/5.5:
        return 6
    else:
        return random.choice([1,2,3,4,5])
```

Property of National Screen Service Corp. Licensed for display only in connection with the exhibition of this picture at your theatre. Must be returned immediately thereafter.

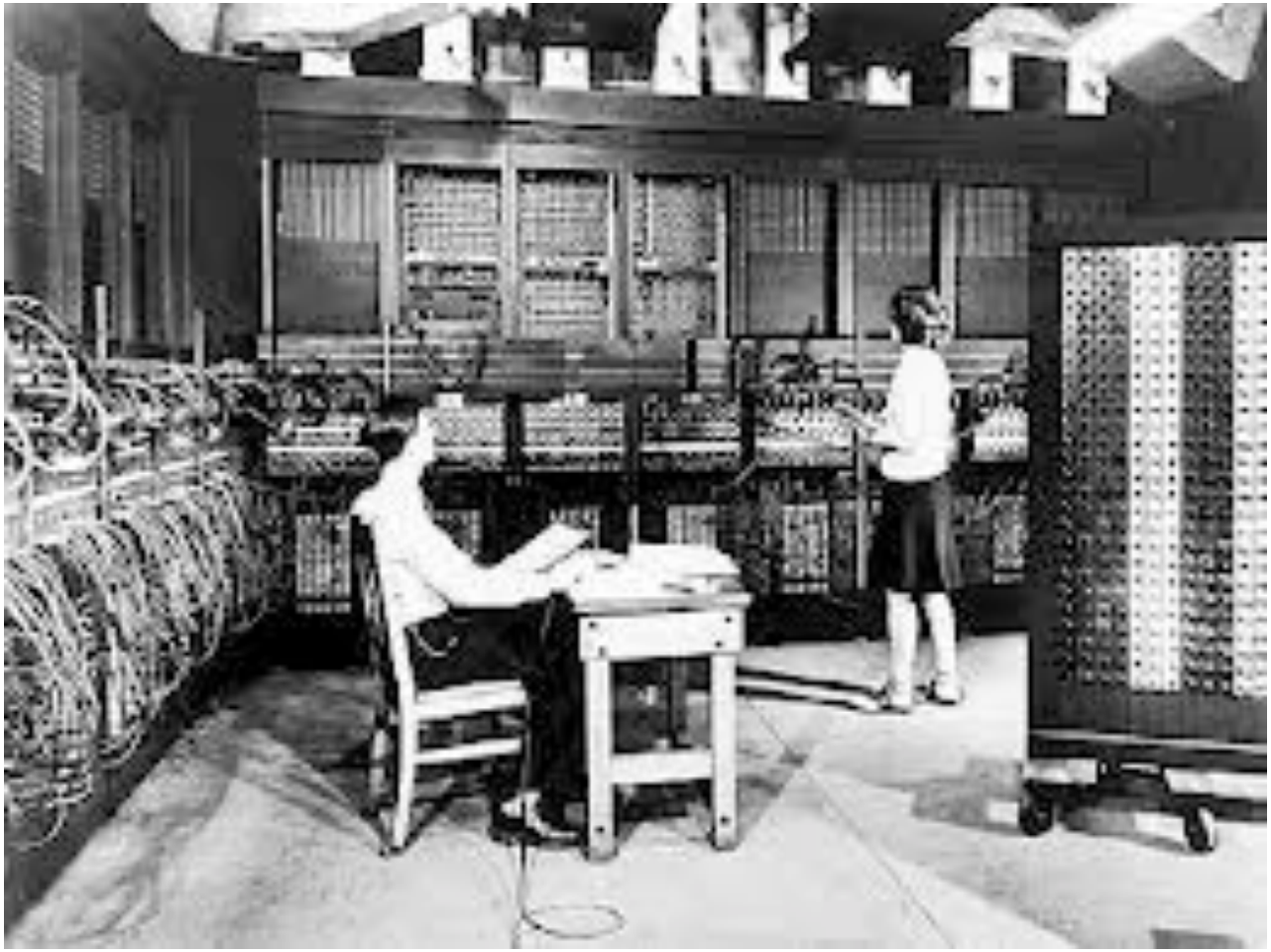
"GUYS AND DOLLS"

55/381

6.00x

Monte Carlo Simulation


undo
01:23
New Game



6.00x

Monte Carlo Simulation

6.00x

Monte Carlo Simulation

6.00x

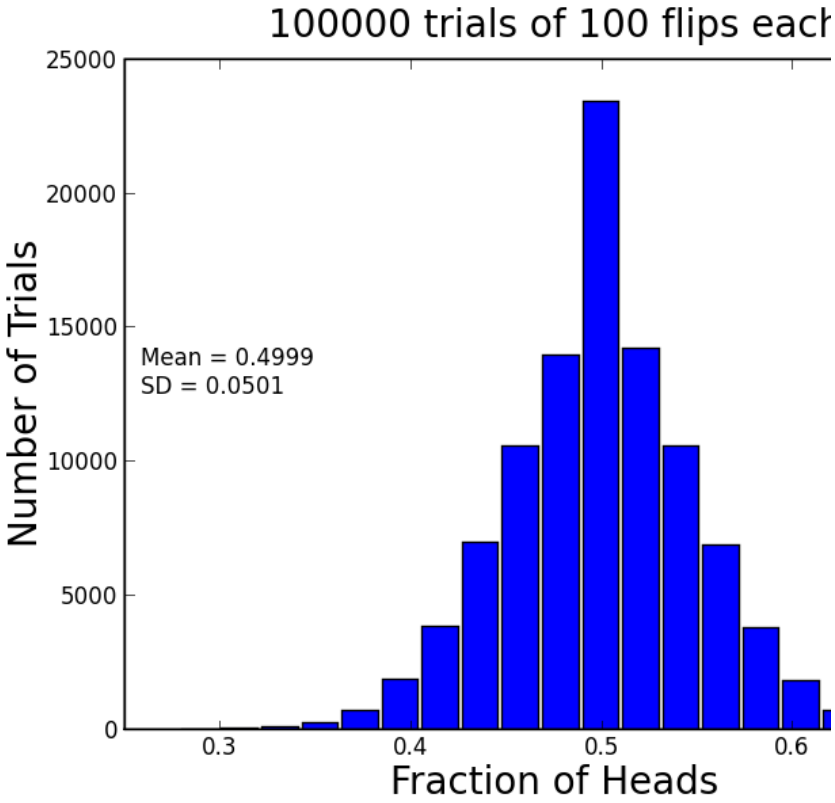
Monte Carlo Simulation

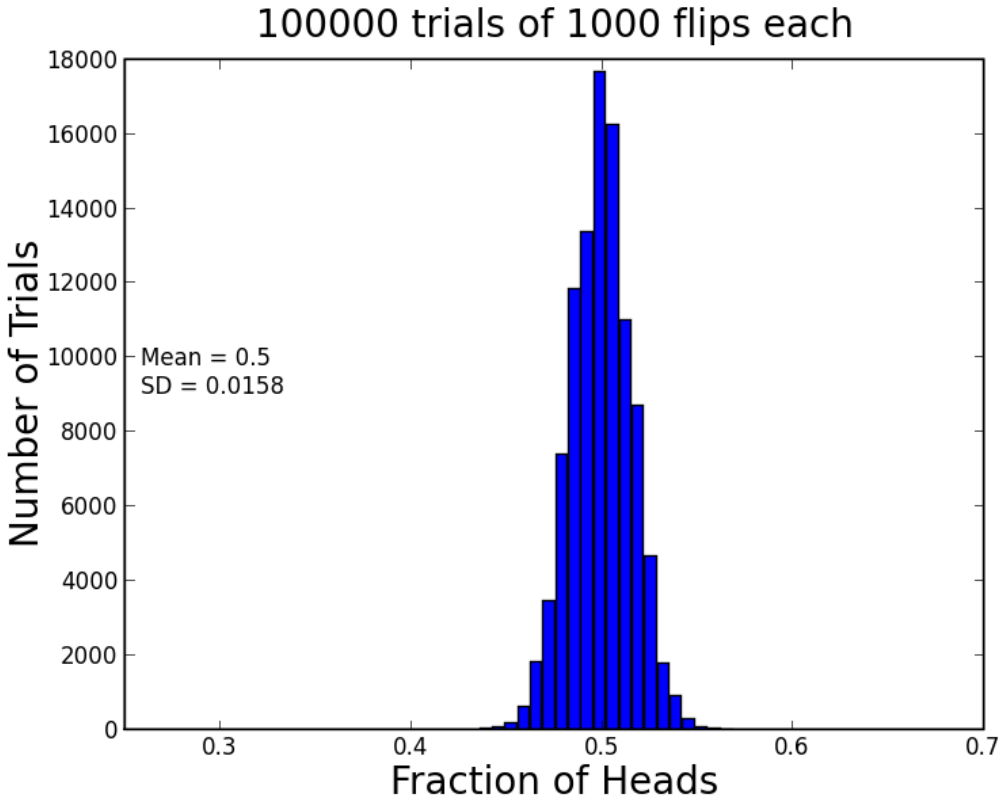

ELIZABETH II
AUSTRALIA
2008

```
def flip(numFlips):
    heads = 0
    for i in range(numFlips):
        if random.random() < 0.5:
            heads += 1
    return heads/float(numFlips)
```

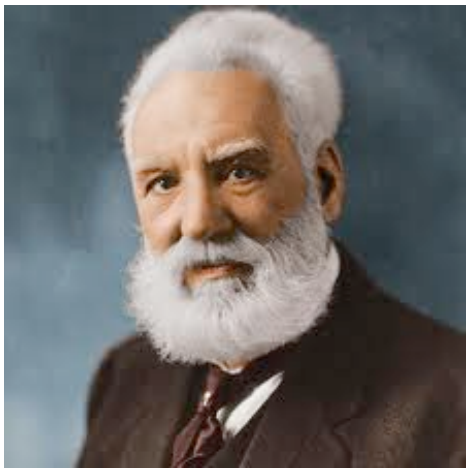
Normal Distributions

Lecturer: John Guttag


100000 trials of 100 flips each
Number of Trials
Fraction of Heads
Mean = 0.4999
SD = 0.0501
25000
20000
15000
10000
5000
0
0.3
0.4
0.5
0.6


100000 trials of 1000 flips each
Number of Trials
Fraction of Heads
Mean = 0.5
SD = 0.0158
18000
16000
14000
12000
10000
8000
6000
4000
2000
0
0.3
0.4
0.5
0.6
0.7

Normal Distributions



Normal Distributions

Normal Distributions

Normal Distributions

```
def makeNormal(mean, sd, numSamples):
    samples = []
    for i in range(numSamples):
        samples.append(random.gauss(mean, sd))
    pylab.hist(samples, bins = 101)
```

Confidence Levels and Intervals

Instead of estimating an unknown parameter by a single value (e.g., the mean of a set of trials), a confidence interval provides a range that is likely to contain the unknown value and a confidence level that the unknown value lays within that range.

Empirical Rule

of the data falls within 1 standard deviation of the mean

of the data falls within 2 standard deviations of the mean

of the data falls within 3 standard deviations of the mean

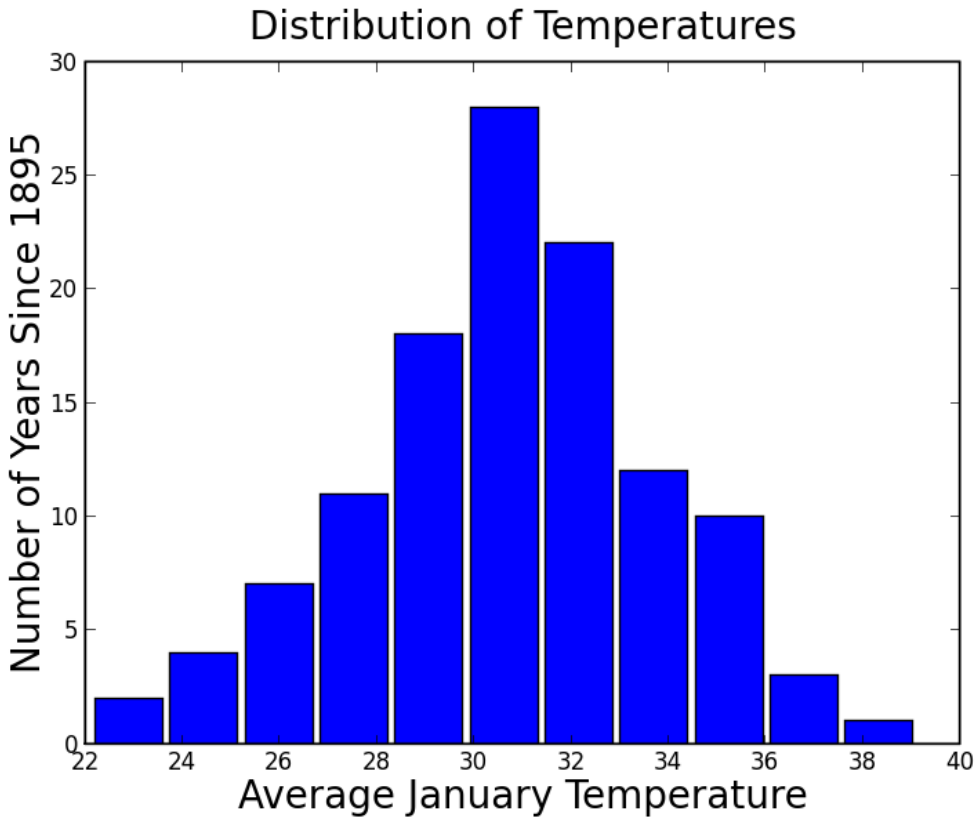
Distributions

Lecturer: John Guttag

Normal Distributions

Normal Distributions

Normal Distributions



Uniform Distributions

Exponential Distributions

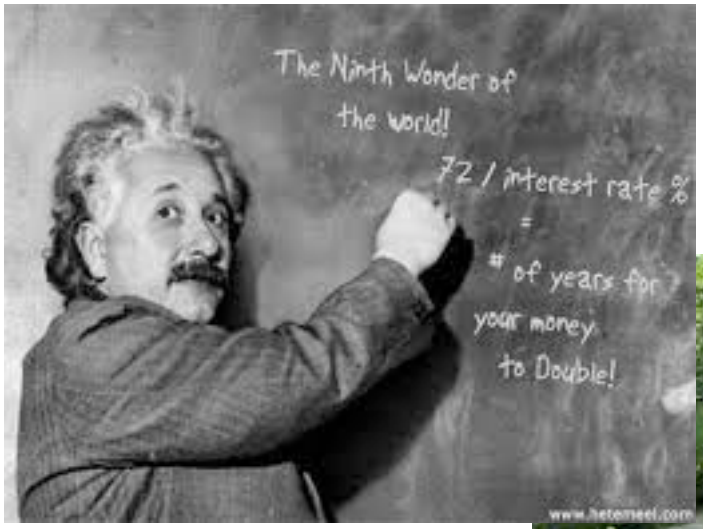
Exponential Distributions

Exponential Distributions

Exponential Distributions

```
def clear(n, clearProb, steps):
    numRemaining = [n]
    for t in range(steps):
        numRemaining.append(n*((1-clearProb)**t))
    pylab.plot(numRemaining,
               label = 'Exponential Decay')
```

Exponential Distributions


The Ninth Wonder of
the world!
72 / interest rate %
=
# of years for
your money
to Double!

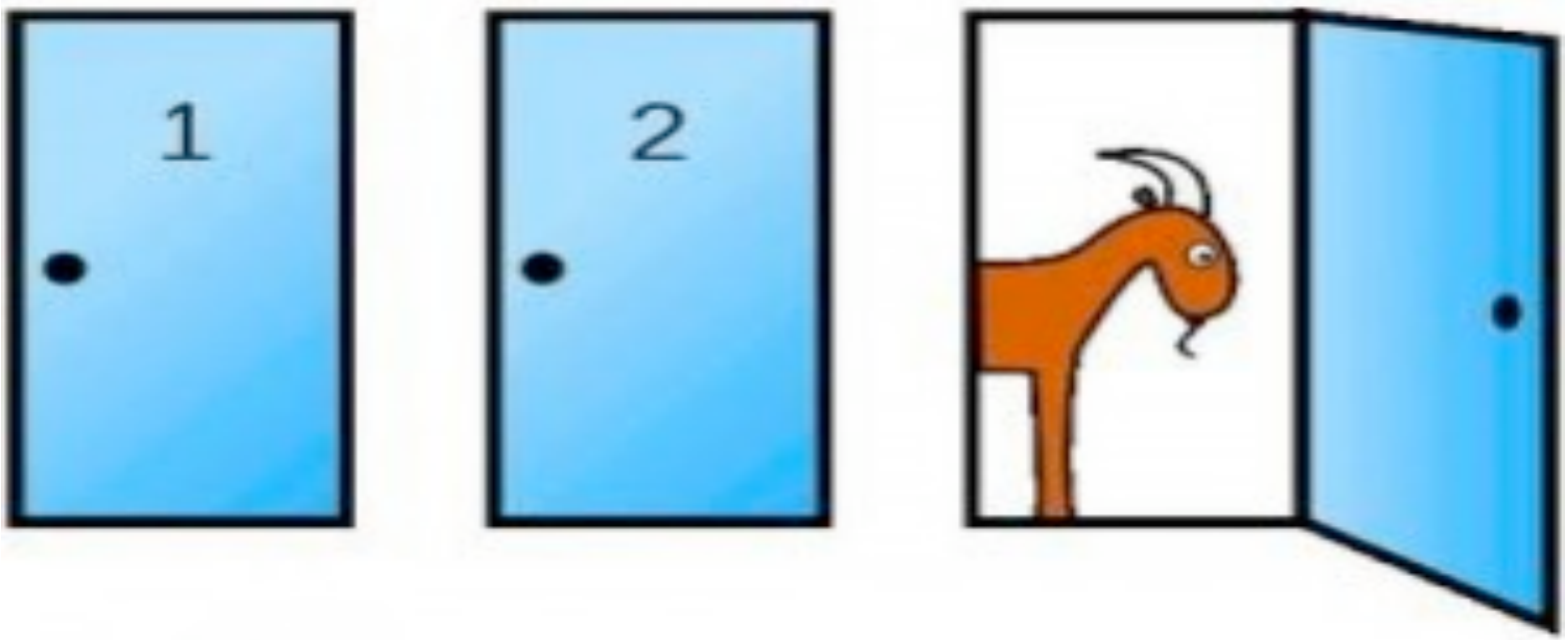
The Monty Hall Problem

Lecturer: John Guttag


1
2
3

She Picks Door 1

Monty Opens a Door that Has a Goat



Do You Want to Switch?

```
def simMontyHall(numTrials = 100,chooseFcn):
    stickWins, switchWins, noWin = (0, 0, 0)
    prizeDoorChoices = [1,2,3]
    guessChoices = [1,2,3]
    for t in range(numTrials):
        prizeDoor = random.choice([1, 2, 3])
        guess = random.choice([1, 2, 3])
        toOpen = chooseFcn(guess, prizeDoor)
        if toOpen == prizeDoor: noWin += 1
        elif guess == prizeDoor: stickWins += 1
        else: switchWins += 1
    return (stickWins, switchWins)
```

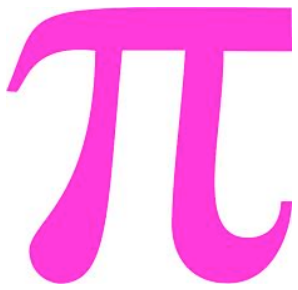
```
def montyChoose(guessDoor, prizeDoor):
    if 1 != guessDoor and 1 != prizeDoor:
        return 1
    if 2 != guessDoor and 2 != prizeDoor:
        return 2
    return 3

def randomChoose(guessDoor, prizeDoor):
    if guessDoor == 1:
        return random.choice([2,3])
    if guessDoor == 2:
        return random.choice([1,3])
    return random.choice([1,2])
```

```
def displayMHSim(simResults, title):
    stickWins, switchWins = simResults
    pylab.pie([stickWins, switchWins],
              colors = ['r', 'c'],
              labels = ['stick', 'change'],
              autopct = '%.2f%%')
    pylab.title(title)
```

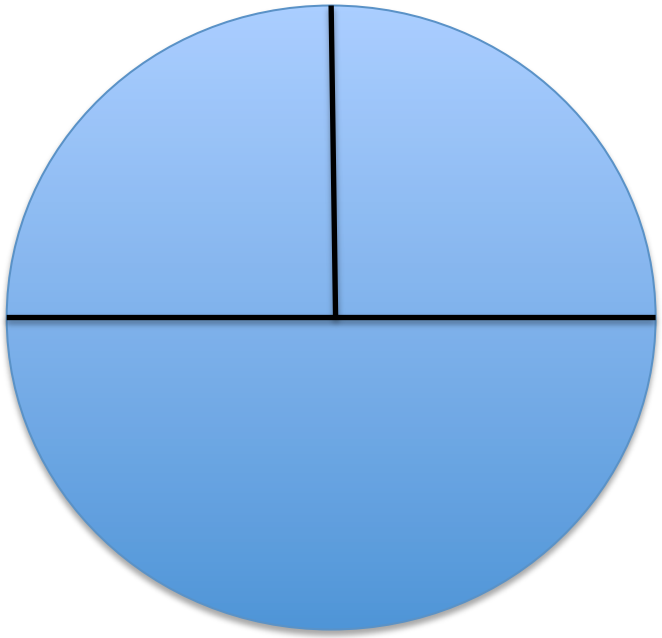
Finding

Lecturer: John Guttag

Finding



Lecturer: John Guttag



$$\frac{circumference}{diameter} = \Pi \qquad area = \Pi * radius^2$$

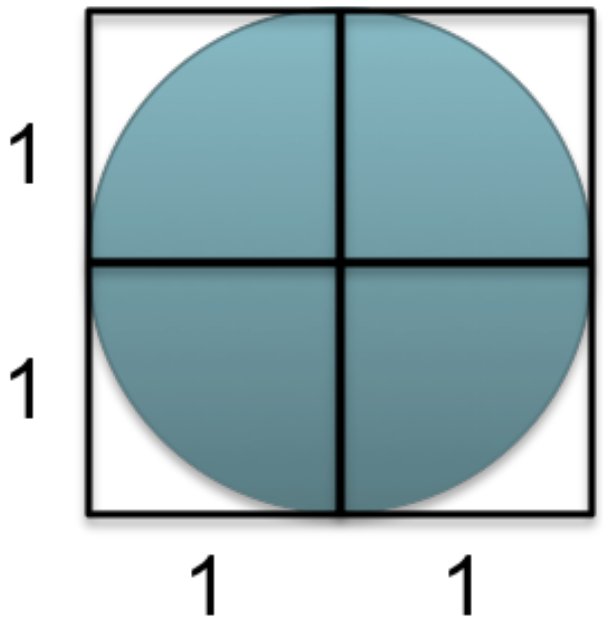
Rhind Papyrus

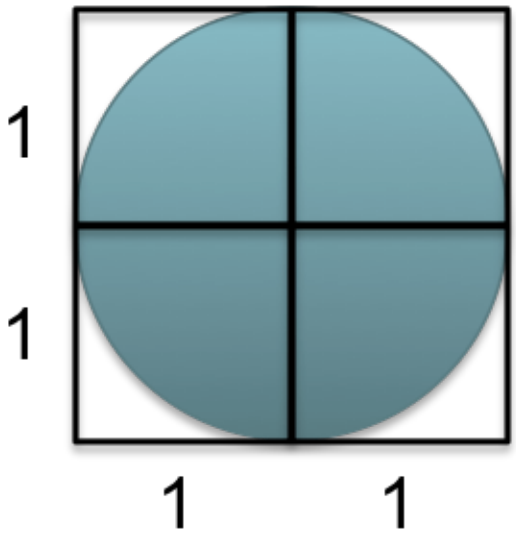
The Bible

"And he made a molten sea, ten cubits from the one brim to the other: it was round all about, and his height was five cubits: and a line of thirty cubits did compass it round about."

—1 Kings 7.23

Archimedes


1
1
1
1


1
1
1
1

```
def throwNeedles(numNeedles):
    inCircle = 0
    for Needles in xrange(1, numNeedles + 1, 1):
        x = random.random()
        y = random.random()
        if (x*x + y*y)**0.5 <= 1.0:
            inCircle += 1
    return 4*(inCircle/float(numNeedles))
```

```
def getEst(numNeedles, numTrials):
    estimates = []
    for t in range(numTrials):
        piGuess = throwNeedles(numNeedles)
        estimates.append(piGuess)
    sDev = stdDev(estimates)
    curEst = sum(estimates)/len(estimates)
    print 'Est. = ' + str(curEst) +\
          ', Std. dev. = ' + str(round(sDev, 6))\
         + ', Needles = ' + str(numNeedles)
    return (curEst, sDev)
```

```
def estPi(precision, numTrials):
    numNeedles = 1000
    sDev = precision
    while sDev >= precision/2.0:
        curEst, sDev = getEst(numNeedles, numTrials)
        numNeedles *= 2
    return curEst
```

```
Est. = 3.14844, Std. dev. = 0.047886, Needles = 1000
Est. = 3.13918, Std. dev. = 0.035495, Needles = 2000
Est. = 3.14108, Std. dev. = 0.02713, Needles = 4000
Est. = 3.141435, Std. dev. = 0.016805, Needles = 8000
Est. = 3.141355, Std. dev. = 0.0137, Needles = 16000
Est. = 3.14131375, Std. dev. = 0.008476, Needles = 32000
Est. = 3.141171875, Std. dev. = 0.007028, Needles = 64000
Est. = 3.1415896875, Std. dev. = 0.004035, Needles = 128000
Est. = 3.14174140625, Std. dev. = 0.003536, Needles = 256000
Est. = 3.14155671875, Std. dev. = 0.002101, Needles = 512000
```

The Right Ballpark