

**6.00 Handout, Lecture 21**  
**(Not intended to make sense outside of lecture)**

class Market:

```
def __init__(self):
    self.tickers = set()
    self.stocks = []
def addStock(self, stk):
    if stk.getTicker() in self.tickers:
        raise 'Duplicate ticker'
    self.tickers.add(stk.getTicker())
    self.stocks.append(stk)
def getStocks(self): return self.stocks
def moveAllStocks(self):
    for stock in self.stocks:
        stock.makeMove()

def generateStocks(n):
    mkt = Market()
    for i in range(n):
        ticker = str(i)
        stk = Stock(ticker, random.random()*10)
        mkt.addStock(stk)
    return mkt
```

```
def simMkt(mkt, numYears, plot):
    endPrices = []
    for stk in mkt.getStocks():
        startPrice = 100
        history = []
        if plot: history.append(startPrice)
        testStk(stk, startPrice, numYears, plot)
        endPrices.append(stk.getPrice())
    if plot:
        xVals = pylab.arange(0, numYears)
        pylab.plot(xVals, history)
    if plot: pylab.figure()
    pylab.hist(endPrices)
    pylab.title('Distribution of Closing Prices')
    pylab.xlabel('Last Sale')
    pylab.ylabel('Number of Securities')
```

---

def makeMove(self):

```
    baseMove = random.uniform(-self.volatility, self.volatility)
    self.price = self.price * (1.0 + baseMove)
    if self.price < 0.01: self.price = 0.0
```

---

def simMkt(mkt, numYears, plot):

```
    endPrices = []
    for stk in mkt.getStocks():
        startPrice = 100
        history = []
        if plot: history.append(startPrice)
        testStk(stk, startPrice, numYears, plot)
        endPrices.append(stk.getPrice())
    mean = sum(endPrices)/len(endPrices)
    print 'Mean Last Sale:', mean
    endPrices.sort()
    for i in range(len(endPrices)):
        if endPrices[i] >= mean: break
    print i, endPrices[i]
    print 'Fraction below mean:', (i-1)/float(len(endPrices))
    pylab.hist(endPrices, bins = 20)
    pylab.title('Distribution of Last Sales')
    pylab.xlabel('Last Sale')
    pylab.ylabel('Number of Securities')
```

```
def makeMove(self):
```

```
    baseMove = random.gauss(0, self.volatility)
    self.price = self.price * (1.0 + baseMove)
    if self.price < 0.01: self.price = 0.0
```

```
def generateStocks(n):
```

```
    mkt = Market()
    for i in range(n):
        ticker = str(i)
        volatility = abs(random.gauss(0.005, 0.02))
        stk = Stock(ticker, volatility)
        mkt.addStock(stk)
    return mkt
```