In [5]:

```
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

We're going to start with generating a random asset price:

In [9]:

```
def generate_brownian_asset_price(S, sigma, r, T):
    """
    S     : current stock
    sigma : volatility
    r     : rate
    T     : time
    """
    ret = S * np.exp((r - 0.5 * sigma**2) * T + sigma * np.sqrt(T) * np.random.r
andn())
    return ret
```

Now set some values for the current stock price `S`, the volatility `sigma`, the interest rate `r`, and the expiration date `T` (in terms of days away from the current date)

In [10]:

```
import datetime
S = 857.29
sigma = 0.2076
r = 0.0014
T = (datetime.date(2013,9,21) - datetime.date(2013,9,3)).days / 365.0
```

Is it anywhere close? yep:

In [13]:

```
generate_brownian_asset_price(S, sigma, r, T)
```

Out[13]:

```
833.91332634495018
```

Next, let's make the call payout:

$$C_t = e^{-r(T-t)} E[max(0, S_T - K)]$$

But we'll adjust for the risk-free interest rate (or discount rate) at the end.

In [1]:

```
def call_payout(S_T, K):
    return max(0, S_T - K)
```

In [2]:

```
call_payout(863.91, 860)
```

Out[2]:

```
3.9099999999999682
```