In [3]:

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

The randomly selected asset price:

In [4]:

```
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
```

The basic call payout:

In [5]:

```
def call_payout(S_T, K):
    """
    S_T : asset price (prediction) at time T
    K   : strike price
    """
    return max(0, S_T - K)
```

Now set the price at 120, the volatility at 25%, the rate at 0.21%, and the strike price at 120 on Oct 16, 2015.

In [6]:

```
S = 120
sigma = 0.25
r = 0.0021
T = (datetime.date(2015,10,16) - datetime.date(2015,9,17)).days / 365.0
K = 120
```

Now we can run this once. But that doesn't give any confidence. So we'll run it *many* times.

In [10]:

```
nsim = 10000
payout = np.zeros((nsim,))
discount = np.exp(-r * T)
```

In [11]:

```
for i in range(nsim):
    S_T = generate_brownian_asset_price(S, sigma, r, T)
    payout[i] = call_payout(S_T, K)
```

In [12]:

```
price = discount * payout.sum() / nsim
print("price: %g" % price)
```

```
price: 3.35782
```

Notice this price seem "reasonable". If we had no interest and no volatility, then the price should be $2. But with interest and with volatility, one would expect this to be more expensive because the payout *may* be better. So 3.3 or so seems reasonable.

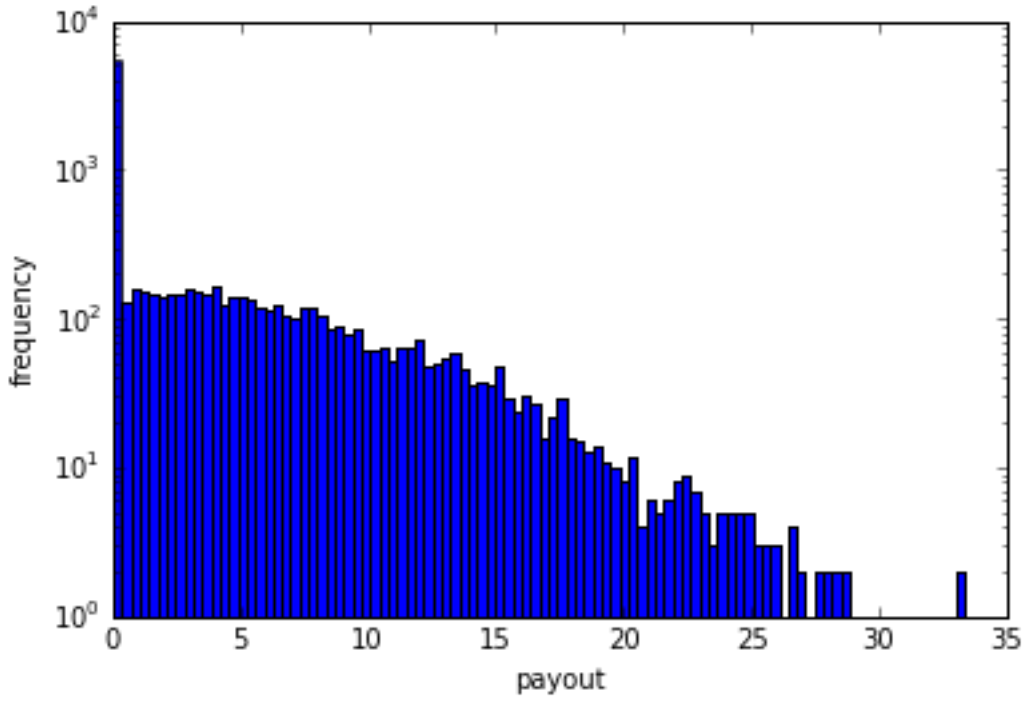
Let's look at how many payouts were at each "value"

In [15]:

```
_ = plt.hist(payout, bins=100, log=True)
plt.xlabel('payout')
plt.ylabel('frequency')
```

Out[15]:

```
<matplotlib.text.Text at 0x10fa56d68>
```



Notice that

1. There are many zeros. This means that the simulation landed with a price below the strike.
2. There's the potential for a much larger payout than 3.3

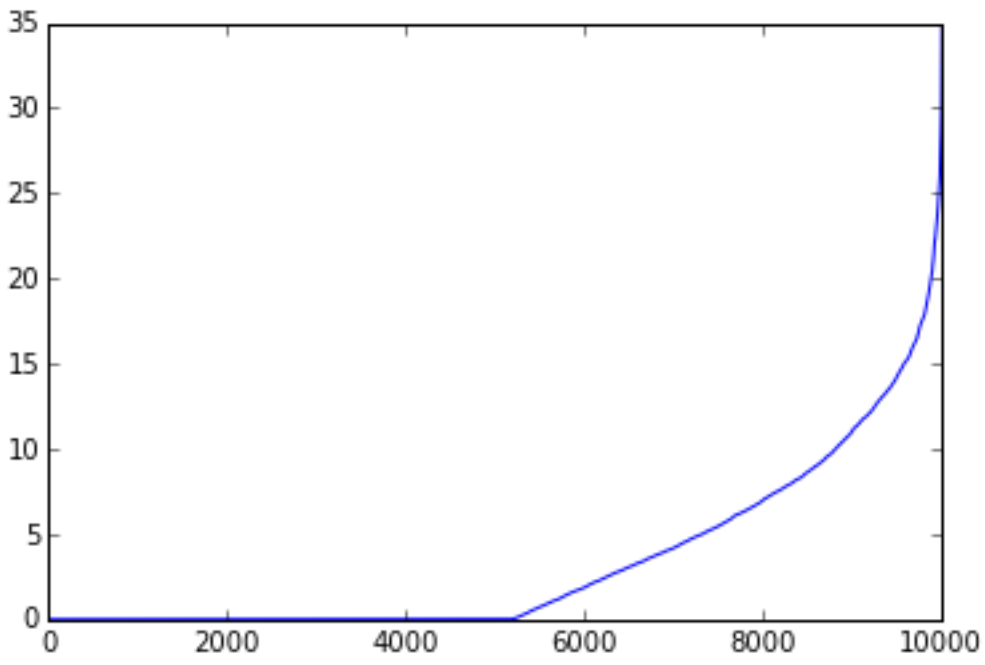
It might be easier to look at it here:

In [16]:

```
plt.plot(np.sort(payout))
```

Out[16]:

```
[<matplotlib.lines.Line2D at 0x10f8fadd8>]
```



In [ ]: