In [1]:
```
import pymc as mc
```

Parameterizing the Negative Binomial Distribution

The PyMC docstring for pymc.negative_binomial_like says:

Negative binomial log-likelihood. The negative binomial distribution describes a Poisson random variable whose rate parameter is gamma distributed. PyMC's chosen parameterization is based on this mixture interpretation.

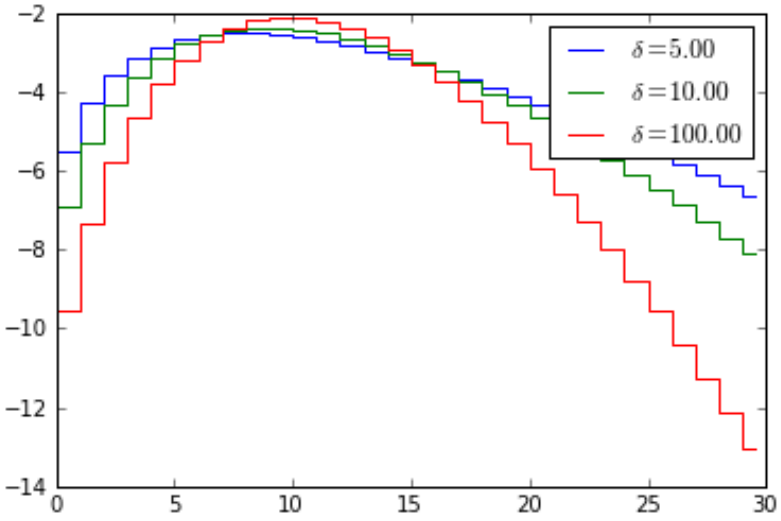
$$f(x \mid \mu, \alpha) = \frac{\Gamma(x+\alpha)}{x!\Gamma(\alpha)} (\alpha/(\mu+\alpha))^{\alpha} (\mu/(\mu+\alpha))^{x}$$

I like calling the first parameter μ for mean, but I like calling the second parameter δ for dispersion. And let's call the value k instead of x, since it is a count:

$$f(k \mid \mu, \delta) = \frac{\Gamma(k+\delta)}{k!\Gamma(\delta)} (\delta/(\mu+\delta))^{\delta} (\mu/(\mu+\delta))^{k}$$

In [2]:
```
# plot the log-likelihood of this distribution for a few values of delta

x = arange(0,30,.5)
mu = 10.
for delta in [5., 10., 100.]:
    y = [mc.negative_binomial_like(x_i, mu, delta) for x_i in x]
    plot(x, y, label='$\\delta=%.2f$'%delta, drawstyle='steps-post')
legend()
```

Out[2]: <matplotlib.legend.Legend at 0x9c28c0c>



I am sure that there is a representation of the negative binomial distribution as a hierarchical model:

$$Y \sim \text{Poisson}(\lambda)$$

$$\lambda \sim \text{Gamma}(\alpha, \beta)$$

The PyMC docstring for pymc.gamma_like says the distribution is

$$f(x \mid \alpha, \beta) = \frac{\beta^{\alpha} x^{\alpha-1} e^{-\beta x}}{\Gamma(\alpha)}$$

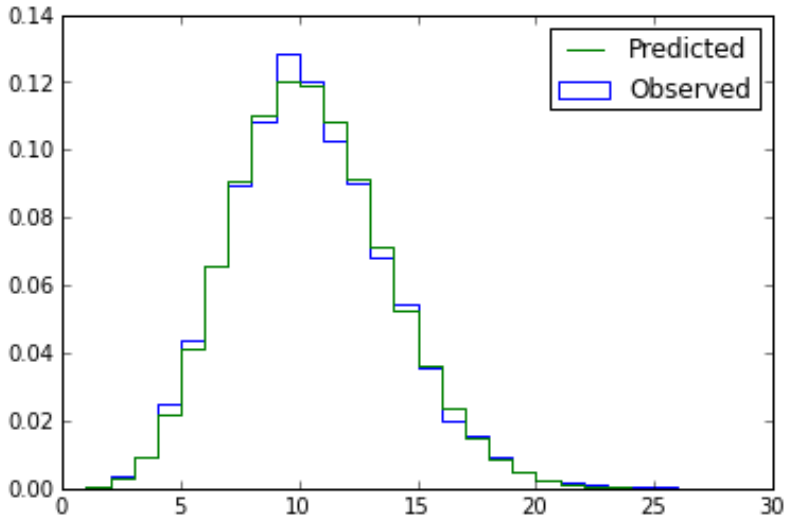
But since I want to represent the negative binomial as a Poisson with a Gamma-distributed rate, I need to write the distribution of Gamma in terms of μ and δ.

I believe that $\alpha = \delta$ and $\beta = \delta/\mu$ is the appropriate transformation. Let's check.

In [3]:
```
lmb = mc.Gamma('lmb', alpha=delta, beta=delta/mu)
Y = mc.Poisson('Y', mu=lmb)

mc.MCMC([lmb, Y]).sample(10000, progress_bar=False)
```

In [4]:
```
# compare hierarchical model to closed form distribution
hist(Y.trace(), range(30), histtype='step', label='Observed', normed=True)

x = arange(0,30,.5)
y = [exp(mc.negative_binomial_like(x_i, mu, delta)) for x_i in x]
plot(x, y, drawstyle='steps-post', label='Predicted')

legend()
```

Out[4]: <matplotlib.legend.Legend at 0xa1f42ac>



In [ ]: