In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
from scipy import stats, special
```

The χ^2 distribution and its properties

The χ^2 distribution of a variable x is given by

$$f(x|\nu) = \frac{x^{\frac{\nu}{2}-1}e^{-\frac{x}{2}}}{2^{\frac{\nu}{2}}\Gamma(\frac{\nu}{2})}$$

for ν = number of degrees of freedom. It represents the **distribution of the variances of samples taken from a gaussian distribution**. The mean and variance of this PDF are given by:

$$\mu = \nu, \sigma^2 = 2\nu$$

$\Gamma(n)$ is the gamma function.

In [2]:
```
for i in np.arange(10):
    n=0.5*i
    print("gamma(%3.1f) = %8.4f" % (n, special.gamma(n)))
```

```
gamma(0.0) =      inf
gamma(0.5) =   1.7725
gamma(1.0) =   1.0000
gamma(1.5) =   0.8862
gamma(2.0) =   1.0000
gamma(2.5) =   1.3293
gamma(3.0) =   2.0000
gamma(3.5) =   3.3234
gamma(4.0) =   6.0000
gamma(4.5) =  11.6317
```

Let's plot the χ^2 distributions for several degrees of freedom.

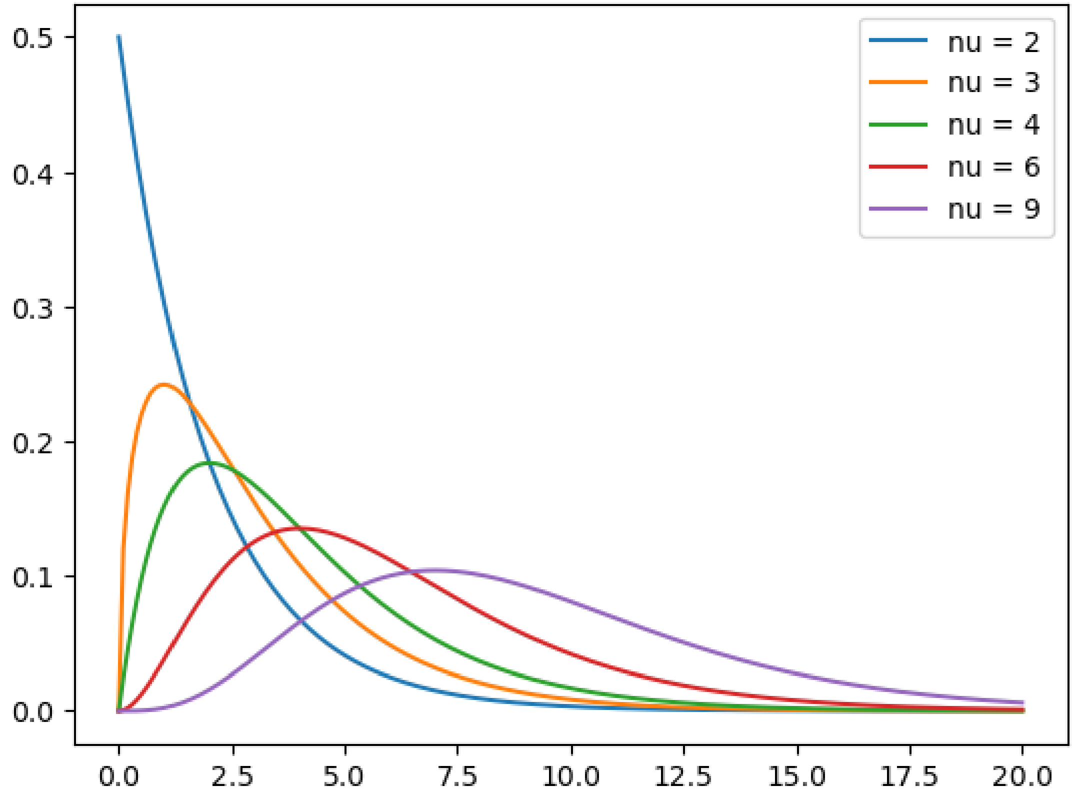
In [3]:
```
x = np.linspace(0, 20, 200)   # 201 bins from 0 to 20
```

In [4]:
```
y1 = stats.chi2.pdf(x, 1)   # 1 degree of freedom
y2 = stats.chi2.pdf(x, 2)   # 2 dof
y3 = stats.chi2.pdf(x, 3)   # 3
y4 = stats.chi2.pdf(x, 4)
y6 = stats.chi2.pdf(x, 6)
y9 = stats.chi2.pdf(x, 9)   # 9
```

In [5]:
```
#plt.plot(x, y1, label='nu = 1')
plt.plot(x, y2, label='nu = 2')
plt.plot(x, y3, label='nu = 3')
plt.plot(x, y4, label='nu = 4')
plt.plot(x, y6, label='nu = 6')
```

```
plt.plot(x, y9, label='nu = 9')
plt.legend()
plt.show
```

Out[5]: `<function matplotlib.pyplot.show(close=None, block=None)>`



What do we mean by the distribution of variances of samples taken from a gaussian distribution? If we drew ν random numbers $\{x_i\}$ from a gaussian distribution with mean μ and variance σ^2, then the following sum:

$$Y = \sum_{i=1}^{\nu} \left(\frac{x_i - \mu}{\sigma} \right)^2 = \sum_{i=1}^{\nu} Z_i^2$$

follows the χ^2 distribution with ν degrees of freedom.

You can integrate the PDF to compute the probability that, for example, χ^2 exceeds some value. e.g.,

$$p(\chi^2 \geq 9|\nu = 5) = \int_9^\infty f(Y|\nu = 5)dY = 0.109$$

i.e., if you were to repeat this measurement many times, you will get a $\chi^2 \geq 9$ 10.8% of the time.

In [6]:
```
# stats.chi2.cdf(x, nu) gives the CDF up to x, so 1 minus that gives you the ir
print('p(chi^2 >= 9) = %8.5f' % (1-stats.chi2.cdf(9, 5)))
```

```
p(chi^2 >= 9) =  0.10906
```

Here is a demonstration of **Theorem 1**.

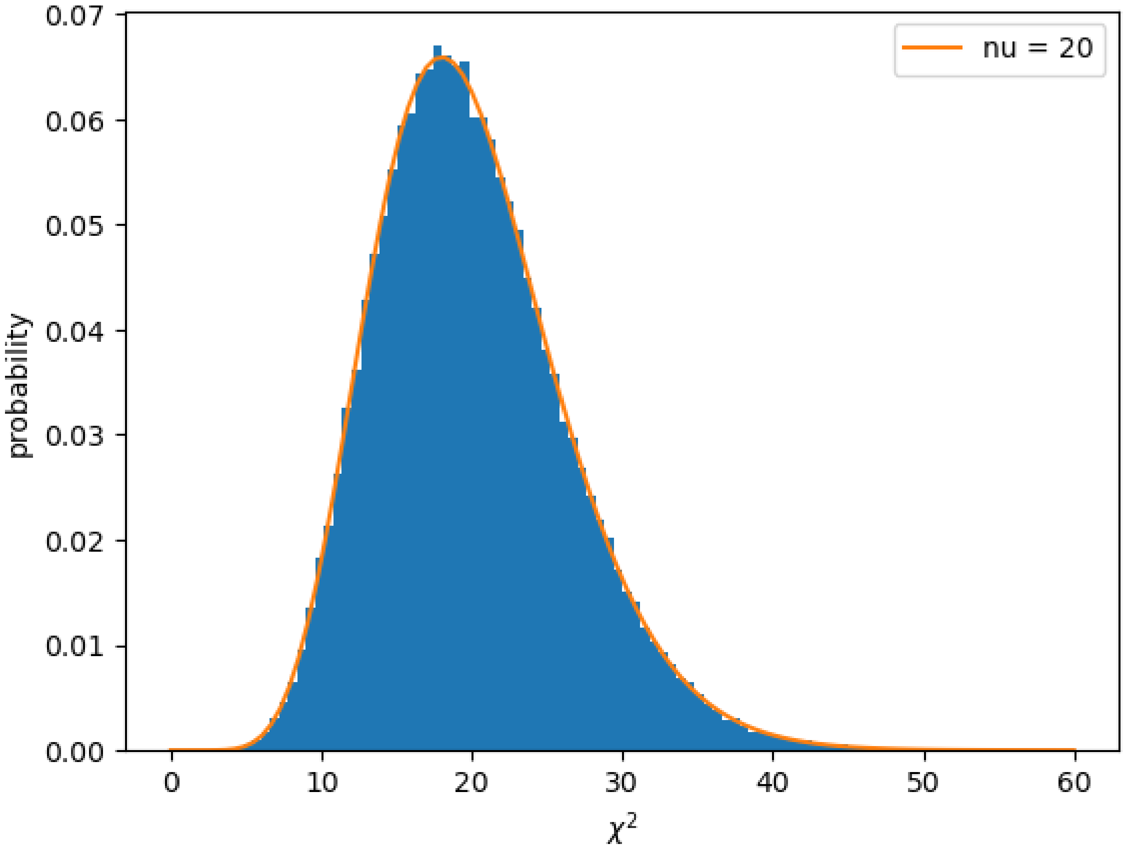
Let $\{x_i\} = x_1, x_2, \ldots, x_n$ be an indepent and identically distributed (IID) sample from a normal distribution $\mathcal{N}(\mu, \sigma)$. Let

$$Y = \sum_{i=1}^{n} \left(\frac{x_i - \mu}{\sigma}\right)^2 = \sum_{i=1}^{n} Z_i^2,$$

where Z_i are standard random variables. Then Y has a chi-squared (χ^2_n) distribution with n degrees of freedom.

In [7]:
```
# We will pick a mean and standard deviation for the parent gaussian
# distribution, but the results are independent of these values.
# Try changing them!
mu  = 5.0   # parent population mean
sig = 2.0   # parent population standard deviation

# Let's draw this many random numbers each simulation.
nu  = 20    # degrees of freedom

# and many simulations so we can plot the distribution.
nsims = 100000

chi2vals = np.zeros(nsims)
for i in range(nsims):
    x = np.random.normal(loc=mu, scale=sig, size=nu)
    chi2vals[i] = np.sum(((x-mu)/sig)**2)

# plot the histogram of the nsims realizations
xmax = nu*3  # x axis to 3 times the mean (=nu)
a,b,c = plt.hist(chi2vals, range=[0, xmax], bins=100, density=True)

# and overplot the theoretical chi-squared distribution with
# the same nu degrees of freedom
xgrid = np.linspace(0, xmax, 100)
chi2pdf = stats.chi2.pdf(xgrid, nu)
plt.plot(xgrid, chi2pdf, label='nu = %d' %nu)
plt.xlabel('$\chi^2$')
plt.ylabel('probability')
plt.legend()
```

Out[7]:
```
<matplotlib.legend.Legend at 0x10dea04f0>
```



Demonstration of the **Example of Theorem 1**.

Given an IID sample $\{x_i\}$ with n elements drawn from an arbitrary distribution with mean μ and variance σ^2 with the sample mean given by

$$\bar{x} = \frac{1}{n}\sum_i x_i,$$

We saw that the distribution of $\bar{x}$ approaches a gaussian with mean μ and variance equal to σ^2/n or $\mathcal{N}(\mu, \sigma/\sqrt{n})$. Therefore, the distribution of

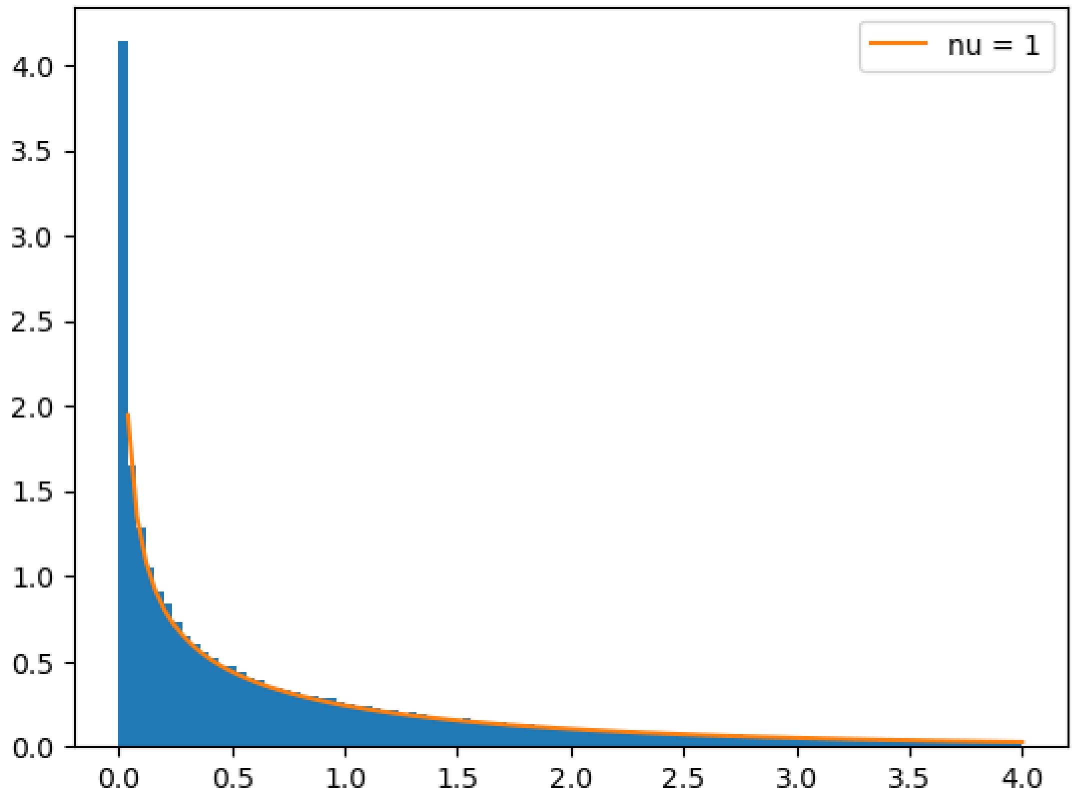
$$Y = \left(\frac{\bar{x}-\mu}{\sigma/\sqrt{n}}\right)^2$$

follows a chi-squared distribution (χ_1^2) with 1 degree of freedom.

In [8]:
```
mu  = 5.0
sig = 2.0

nu  = 20    # degrees of freedom

nsims = 100000

chi2vals = np.zeros(nsims)
for i in range(nsims):
    x = np.random.normal(loc=mu, scale=sig, size=nu)
    # this time we want to compute (mean(x)-mu)*sqrt(n)/sigma
```

```
    chi2vals[i] = ((np.mean(x)-mu)*np.sqrt(nu)/sig)**2

xmax = nu/5
a,b,c = plt.hist(chi2vals, range=[0, xmax], bins=100, density=True)

# chi-squared distribution with 1 dof
xgrid = np.linspace(0, xmax, 100)
chi2pdf = stats.chi2.pdf(xgrid, 1)
plt.plot(xgrid, chi2pdf, label='nu = 1')
plt.legend()
```

Out[8]: <matplotlib.legend.Legend at 0x1bf639d80>



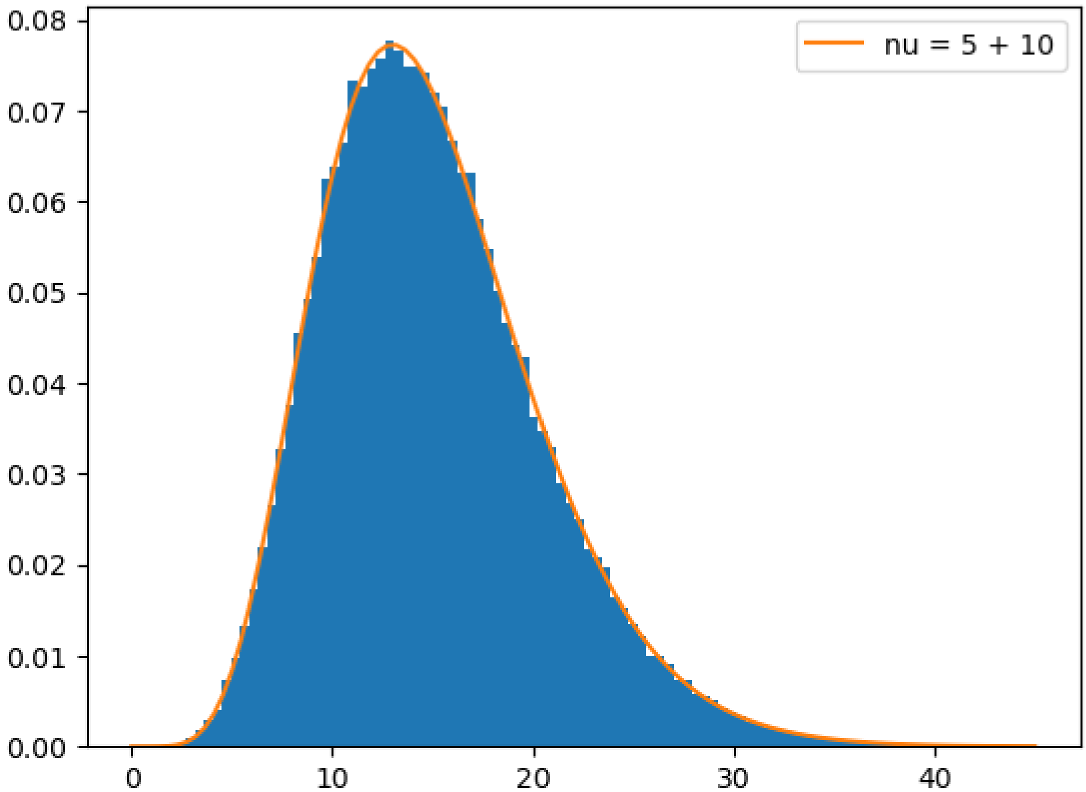
Demonstration of **Theorem 2**.

If Y_1 and Y_2 are two independent χ^2-distributed random variables with ν_1 and ν_2 degrees of freedom, then $Y = Y_1 + Y_2$ is also χ^2-distributed with $\nu_1 + \nu_2$ degrees of freedom.

In [9]:
```
mu1  = 5.0
sig1 = 2.0
nu1  = 5    # degrees of freedom

mu2  = 3.0
sig2 = 4.0
nu2  = 10    # degrees of freedom

nsims = 100000

chi2vals = np.zeros(nsims)
for i in range(nsims):
    x1 = np.random.normal(loc=mu1, scale=sig1, size=nu1)
    x2 = np.random.normal(loc=mu2, scale=sig2, size=nu2)
```

```
    chi2vals[i] = np.sum(((x1-mu1)/sig1)**2) + np.sum(((x2-mu2)/sig2)**2)

xmax = (nu1+nu2)*3
a,b,c = plt.hist(chi2vals, range=[0, xmax], bins=100, density=True)

xgrid = np.linspace(0, xmax, 100)
chi2pdf = stats.chi2.pdf(xgrid, nu1+nu2)
plt.plot(xgrid, chi2pdf, label='nu = %d + %d' % (nu1, nu2))
plt.legend()
```

Out[9]: <matplotlib.legend.Legend at 0x1bf9f21a0>



We define the **sample variance** S^2 to be

$$S^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2.$$

Theorem 3 says that the sampling distribution of

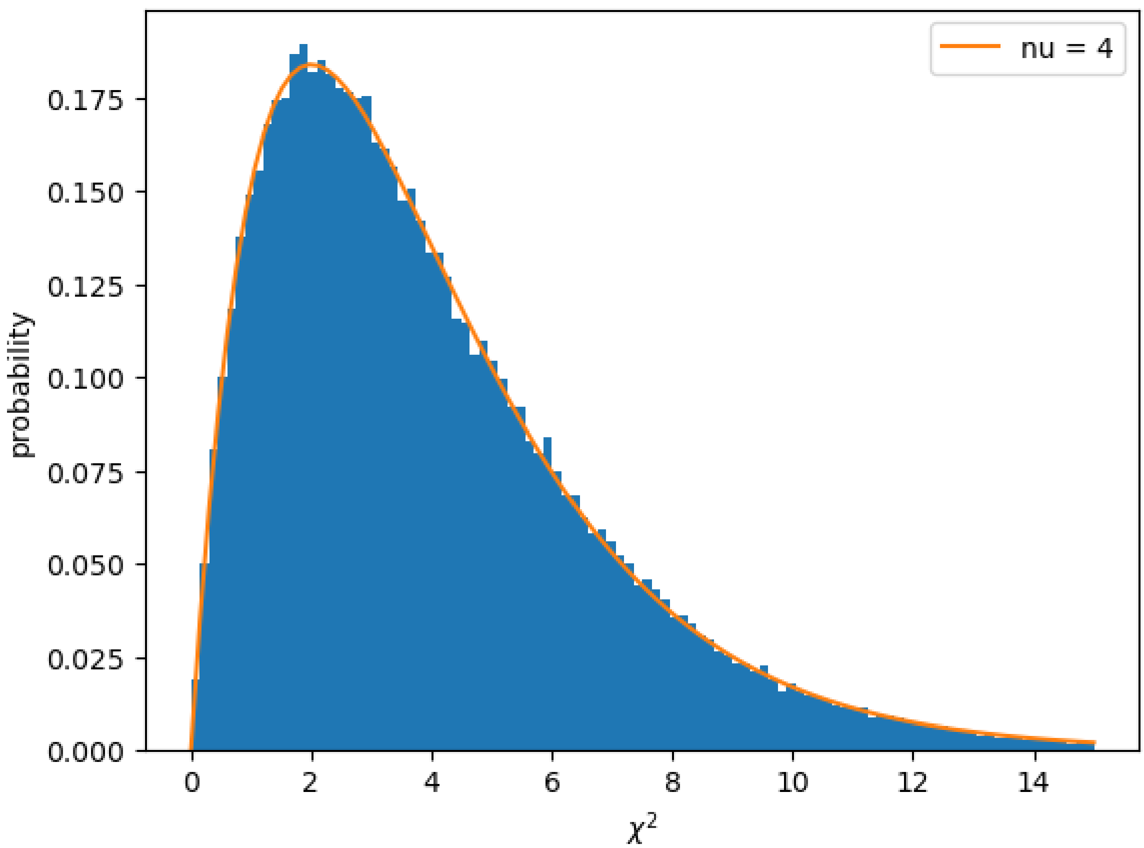
$$(n-1)\frac{S^2}{\sigma^2}$$

is χ^2_{n-1} with $(n-1)$ degrees of freedom. Here is a demonstration.

In [10]:
```
nu  = 5    # degrees of freedom
mu  = 5.0
sig = 2.0

xmax = nu*3

nsims = 100000
```

```
chi2vals = np.zeros(nsims)
for i in range(nsims):
    x = np.random.normal(loc=mu, scale=sig, size=nu)
    smu  = np.mean(x)
    svar = np.sum((x-smu)**2)/(nu-1)
    chi2vals[i] = (nu-1)*svar/(sig**2)

a,b,c = plt.hist(chi2vals, range=[0, xmax], bins=100, density=True)

xgrid = np.linspace(0, xmax, 100)

chi2pdf = stats.chi2.pdf(xgrid, nu-1)
plt.plot(xgrid, chi2pdf, label='nu = %d' % (nu-1))
plt.xlabel('$\chi^2$')
plt.ylabel('probability')
plt.legend()
```

Out[10]: <matplotlib.legend.Legend at 0x1bf88ee30>



If we replace the sample mean $\bar{x}$ with the true mean μ (i.e., if we somehow knew it), we can see that the sample variance is given instead by

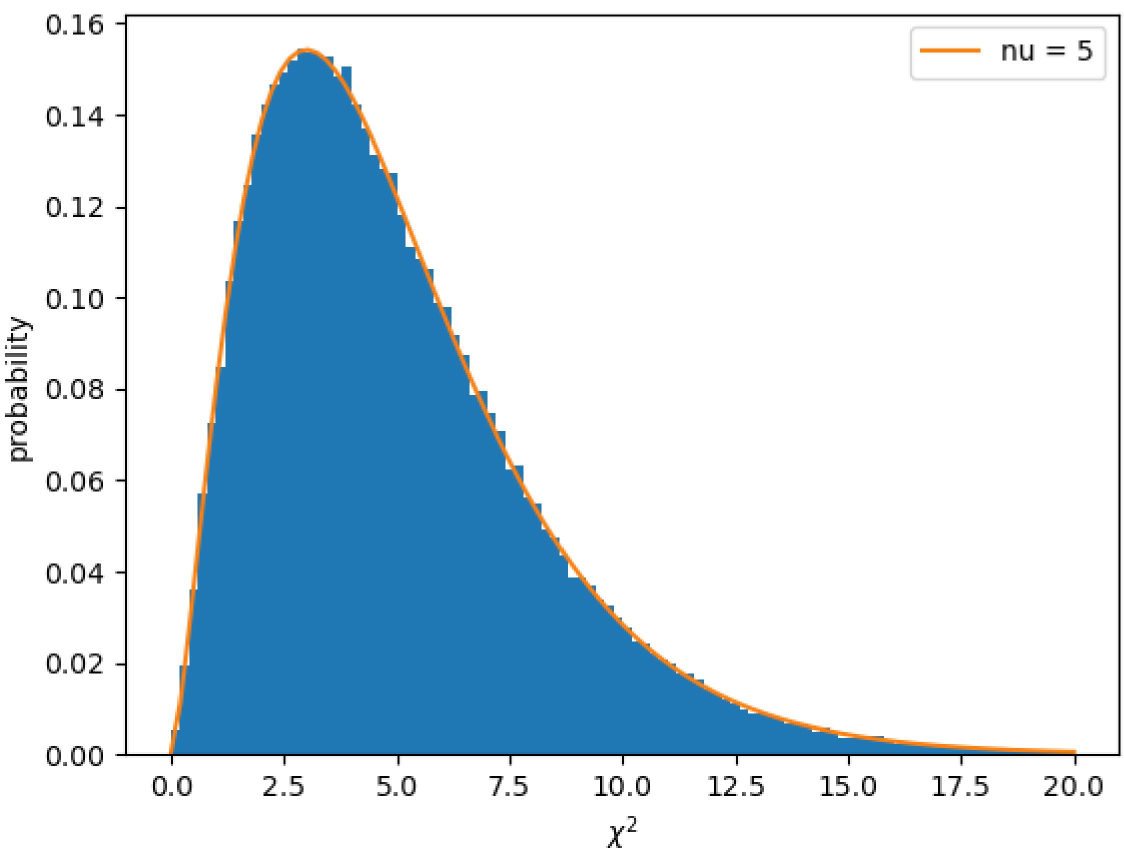
$$S^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2$$

i.e., we do **not** lose one degree of freedom.

In [11]:
```
mu  = 5.0
sig = 2.0
```

```
nu  = 5    # degrees of freedom

nsims = 100000

chi2vals = np.zeros(nsims)
for i in range(nsims):
    x = np.random.normal(loc=mu, scale=sig, size=nu)
    svar = np.sum((x-mu)**2)/nu
    chi2vals[i] = nu*svar/(sig**2)

xmax = nu*4
a,b,c = plt.hist(chi2vals, range=[0, xmax], bins=100, density=True)

xgrid = np.linspace(0, xmax, 100)
chi2pdf = stats.chi2.pdf(xgrid, nu)
plt.plot(xgrid, chi2pdf, label='nu = %d' % nu)
#chi2pdfwrong = stats.chi2.pdf(xgrid, nu-1)
#plt.plot(xgrid, chi2pdfwrong, label='nu = %d' % (nu-1))
plt.xlabel('$\chi^2$')
plt.ylabel('probability')
plt.legend()
```

Out[11]: <matplotlib.legend.Legend at 0x1bfa3cb20>

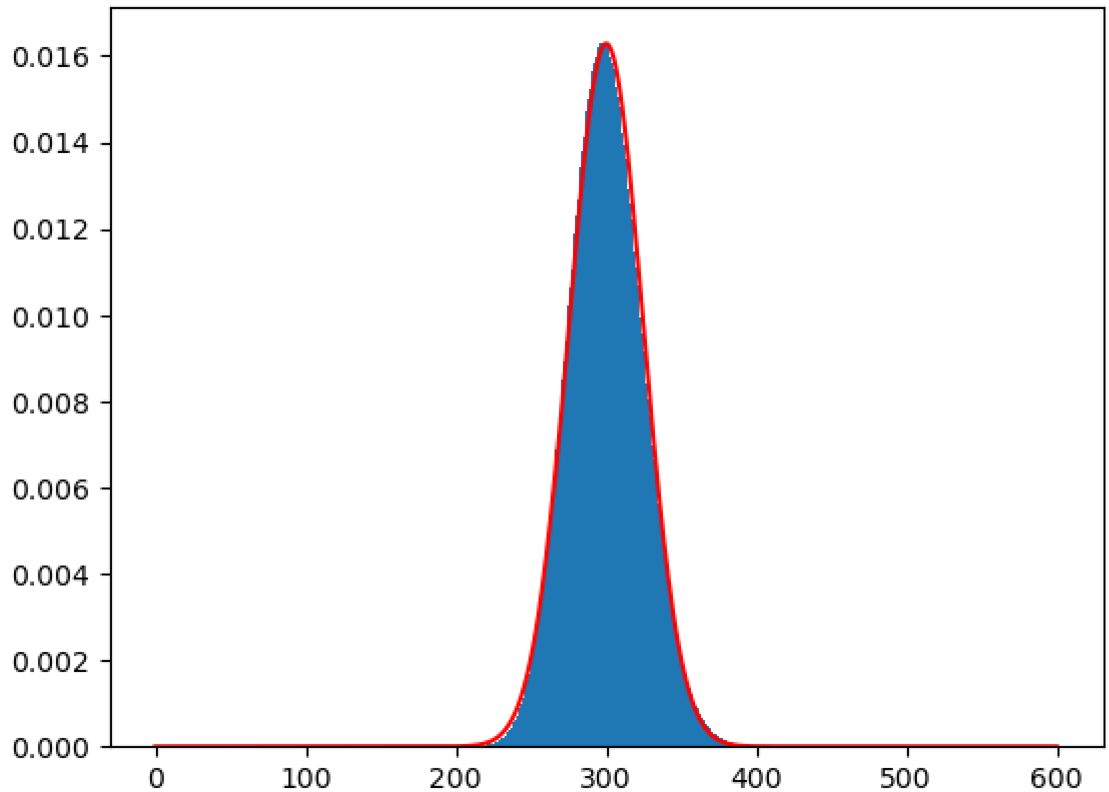


Due to the central limit theorem, the χ^2 distribution also approaches a gaussian distribution at (rather) large values of ν. Recalling that the mean and standard deviation of the χ^2 distribution are $\mu = \nu$ and $\sigma = \sqrt{2\nu}$.

In [12]:
```
nu = 300
```

```
x = np.arange(2*nu+1)    # = [0 1 .. 150]
p = stats.chi2.pdf(x, nu)

mean = nu
std  = np.sqrt(2*nu)
nx = np.linspace(0, 2*nu, 2*nu)
ny = stats.norm.pdf(nx,mean,std)

plt.bar(x, p, width=1)
plt.plot(nx, ny, color='r')
```

Out[12]: [<matplotlib.lines.Line2D at 0x1bfd39d50>]



In [ ]: