In [2]:

```
import numpy as np
import matplotlib.pyplot as plt
import scipy as sp
import hamhelper.plotting as hp
# import hamhelper.colours as hc

# cmap = hc.HamColour.from_name('ptol_bright').cmap()
# print(cmap)
```

Q1

> Consider a gas of N0 non-interacting atoms enclosed in a container of volume V0. Focus attention on any sub-volume V of this container and denote by N the number of molecules located within this sub-volume. Each atom is equally likely to be located anywhere within the container; hence the probability that a given atom is located within the sub-volume V is equal to V/V0. Using the ideas of the binomial distribution, answer the following questions:

(A) Mean number of atoms within sub-volume V

$\langle N \rangle$ within V is given by: $\frac{\langle N \rangle}{V} = \frac{N_0}{V_0} \rightarrow \langle N \rangle = N_0 \frac{V}{V_0}$

(B) Relative fraction in the number of atoms within V

$$\frac{\left\langle (N - \langle N \rangle)^2 \right\rangle}{\langle N \rangle^2} = \frac{\sigma^2}{\langle N \rangle^2}.$$

We can then use $\sigma^2 = N_0 p(1-p)$, where p is the probability of a particle being contained within the sub-volume, given by V/V_0 in part (a). Substituting the result from (a) we show

$$\frac{\sigma^2}{\langle N \rangle^2} = \frac{N_0 \frac{V}{V_0}(1 - \frac{V}{V_0})}{(N_0 \frac{V}{V_0})^2} = \frac{1}{N_0}\left(\frac{V_0}{V} - 1\right) \square$$

(C) The limit as $V << V_0$

Taking the limit as V shrinks small compared to V_0 we end up with the V_0/V term blowing up in our expression above

$$\frac{\sigma^2}{\langle N \rangle^2} \rightarrow \frac{V_0}{N_0 V} = \frac{1}{\langle N \rangle}$$

(D) The limit as $V \rightarrow V_0$

Taking the limit as V approaches the full volume V_0, we observe

$$\sigma^2 = N_0 \frac{V}{V_0}\left(1 - \frac{V}{V_0}\right) \xrightarrow[2^{\text{nd}}\text{order}]{} N_0\left(\frac{V_0}{V} - 1\right) \xrightarrow[1^{\text{st}}\text{order}]{} 0$$

meaning that the relative fraction approaches

$$\frac{\sigma^2}{\langle N \rangle^2} \to \frac{0}{N_0} = 0.$$

This makes sense as we then have all particles of the volume contained within the sub-volume with 100% certainty as they occupy the same space.

Q2

> Six monkeys, set to strum unintelligibly on typewriters for millions of years, would be bound in time to write all the books in the British Museum." - Aldous Huxley This is a rather misleading statement. Could all the monkeys in the world have typed out ANY book during the age of the universe? Calculate the likelihood of the monkeys producing an 'average' book, Shakespeare's Hamlet.

(A) A typewriter has 44 keys, assuming the length of Hamlet is 10^5 characters, what is the probability that an exact sequence of those characters forms?

This is asking if we have 10^5 characters chosen from 44 uncorrelated options, what is the correct they are the exact right sequence. The probability comes from the number of permutations possible $\to 44^{(10^5)} \approx 10^{164345}$. This is large. The exact sequence is one of these many options,

$$\therefore P = \frac{1}{44^{(10^5)}} \approx 0$$

(B) What is the probability this will be typed over the age of the universe, assuming three monkeys per human all typing 10 keys per second?

Lets assume

$$t = 10^{18} \text{ seconds,}$$
$$R = 10 \text{ characters per second, and}$$
$$N = 3 \times 7.48 \times 10^9 \text{ monkeys.}$$

Again, we consider the odds of choosing 10^5 exact characters from a set of 44 options. This will be sampled from $L = t \times R \times N = 2.244 \times 10^{30}$ characters. We can calculate the number of substrings as

$$N_{\text{sub}} = L - (10^5 - 1) = 2.244 \times 10^{30} - 10^5 + 1 \approx 2.244 \times 10^{30} \text{ within computer error!}$$

So, if each of these substrings has a chance of being the correct version of Hamlet, what is the probability considering all strings we get hamlet. We find

$$P_{\text{tot}} = N_{\text{sub}} \times P = \frac{2.244 \times 10^{30}}{44^{(10^5)}} \approx 0,$$

so it does not matter! The number of ways to not arrange Hamlet are so many that there is no physical way for monkeys to do it (statistically).

Q3

> There are an equal number of ripe (yellow) and unripe (green) bananas in a very large barrel. If a blindfolded monkey pulls bananas from the barrel, what is the probability that the monkey will get 215 ripe bananas if it grabs 400 bananas?

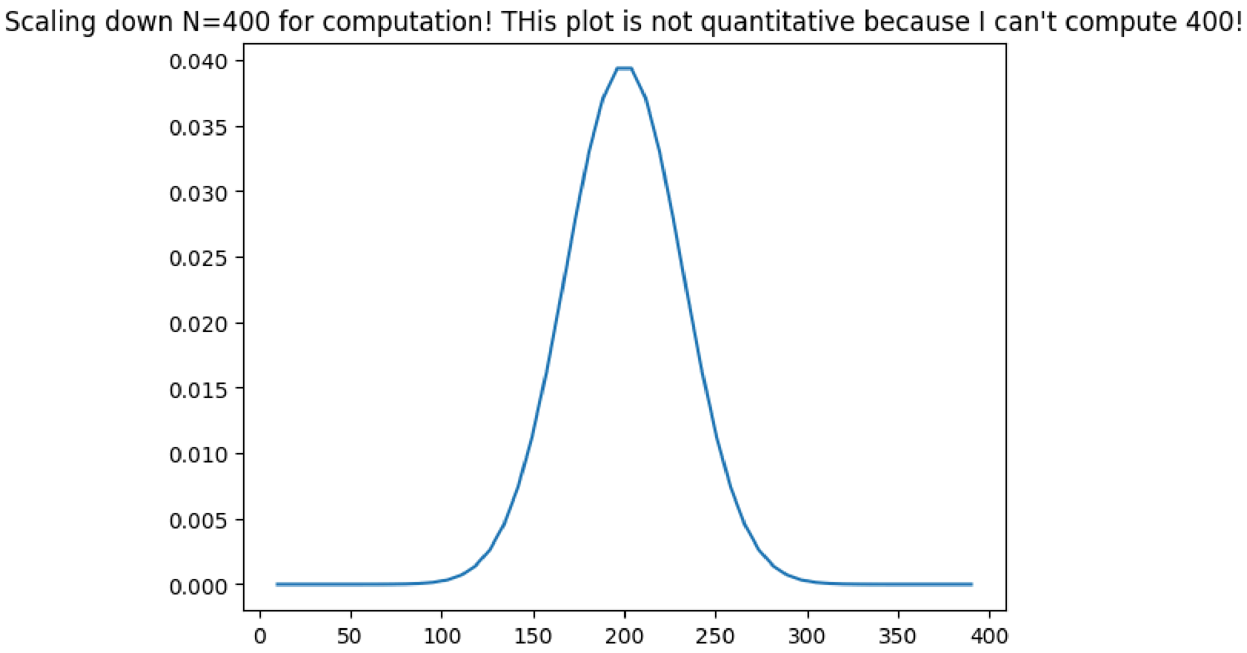
From a large bath of bananas, there is equal probability ($p = 0.5 = q$) for pulling a yellow or green banana respectively. This is essentially a random walk process of $n = 400$ "steps" where we want to find the probability of being some total "steps" to the ripe banana side n_R there are in total.

$$P\left(n_R\right) = \frac{n!}{n_R!\left(n - n_R\right)!} p^{n_R} q^{n-n_R} = \frac{400!}{215!\left(400 - 215\right)!} 0.5^{215} 0.5^{400-215} = 0.012966 \ \approx \frac{1.3}{100}$$

This is a pretty low chance! We would expect to see a result much closer to 200 flat.

In [3]:

```
N = 400
ns = np.linspace(1,39)

def P(n, N=40):
        return (sp.special.factorial(N)/(sp.special.factorial(n) * sp.special.factorial

p = P(ns)
# print(p)

ns *= 10
p *= np.sqrt(40)/np.sqrt(400)
plt.plot(ns, p, '-')
plt.title("Scaling down N=400 for computation! THis plot is not quantitative because I
plt.show()
```



Q4

> Consider a collection of seventy noninteracting spins that may either point up or down. How many microstates are there in this system? How can we label the macrostates in this system? If each microstate is equally probable, what is the most likely macrostate to find the system in, and what is the probability that we will find the system in this configuration? (From Kennett).

Letting

$$N = 10 \text{ non-interacting spins,}$$
$$s = 2 \text{ states per spin,}$$

meaning we have $s^N = 2^{70}$ microstates. We can label the macrostates by the number of spin-up particles n_u. Assuming both up and down equally probably for all spins, the most likely macrostate is the one with the higher multiplicity Ω. In other words when the entropy is highest. This occurs when $n_u = 35 = N/2$. We can then calculate the probability of this system using the Binomial distribution

$$P(n_u) = \frac{N!}{n_u!(N-n)!}(0.5)^{n_u}(0.5)^{N-n_u} = 0.0950 \approx \frac{1}{10},$$

whose maximum occurs at $\frac{N}{2} = 35$.

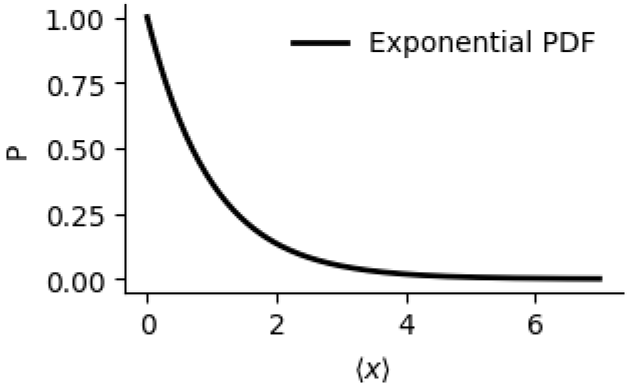
Q5

> Perform a numerical simple calculation using **python** (or whatever program you like) of the random walk problem worked out in class but where the step length is given

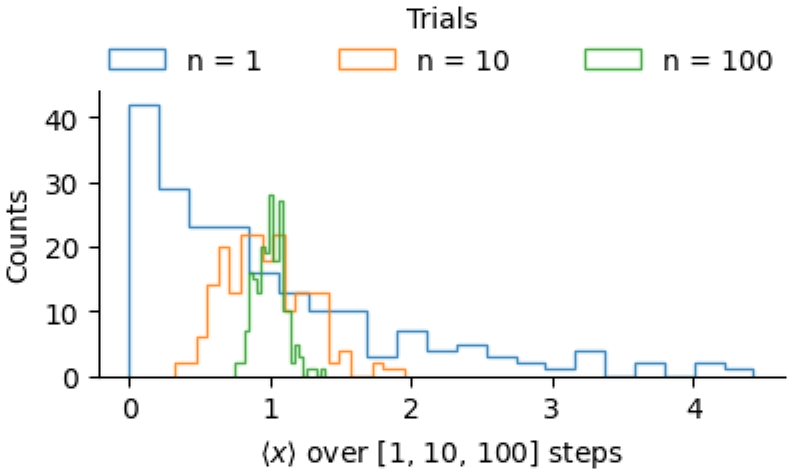
> by a continuous exponential distribution of the form $P(x) = e^{-x}$ where x is the step length in metres and varies from zero to infinity.
>
> 1. First, calculate the step length $\langle x \rangle$ and the variance of this distribution.
> 2. Then plot a histogram of 200 trials for n random steps where $n = 1,\ 10,\ \text{and } 100.$
> 3. Comment on the shape of the observed distribution for the three cases.
> 4. Is the width of the distributions consistent wth theory?

In [4]:

```
import scipy.stats as st
import hamhelper.plotting as hp
#noinput

def step_PDF(x):
        # Distribution of step lengths, continuous for x > 0
        return st.expon.pdf(x)

def draw_from_PDF(N=1):
        return st.expon.rvs(size=N)

fig, ax = plt.subplots(1, 1, figsize=np.array([3.3,2.2]))
x = np.linspace(0, 7, 200)
ax.plot(x, st.expon().pdf(x), 'k-', lw=2, label='Exponential PDF')
ax.legend(framealpha=0)
ax.set(xlabel=r'$\langle x \rangle$', ylabel='P')
hp.despine(ax)
plt.tight_layout()
plt.savefig('plots/A1_Q5_PDF.pdf', bbox_inches='tight')
plt.savefig('plots/A1_Q5_PDF.png', bbox_inches='tight', dpi=300)
plt.show()

# Get mean step lenth and distribution
mean, var, skew, kurt = st.expon.stats(moments='mvsk')
print(f"Mean = {mean:.5f}")
print(f"Variance = {var:.5f}")

N = 200  # Trials
ns = [1, 10, 100] #, 10, 100]#, 10, 100]  # Steps

# Generate trials
fig, ax = plt.subplots(1, 1, figsize=np.array([4.2,2.7]))
nbins = 41
bins = np.linspace(0,7,nbins+1)
bins=21
for n in ns:
    disp_array = []
    for trial in range(N):
        x_array = draw_from_PDF(n).copy()
        mean_disp = np.mean(x_array)
        stdev = np.std(x_array)
        var = stdev**2
        disp_array.append(mean_disp)

    disp_array = np.array(disp_array)
    # print(disp_array)
    counts, _bins = np.histogram(disp_array, bins=bins, density=False)
    ax.hist(_bins[:-1], _bins, weights=counts, label=f'n = {n}', histtype='step', alpha
ax.set(xlabel=r'$\langle x\rangle$ over'+f' {ns} steps', ylabel='Counts')
ax.legend(framealpha=0, title='Trials', bbox_to_anchor=(0, 1.02, 1, 0.2), loc="lower le
                  mode="expand", borderaxespad=0, ncol=3)
hp.despine(ax)
plt.tight_layout()
ax.set(facecolor = "none")
plt.savefig('plots/A1_Q5_distributionOfDisplacements.pdf', bbox_inches='tight')
plt.savefig('plots/A1_Q5_distributionOfDisplacements.png', bbox_inches='tight', dpi=300
plt.show()
```



```
Mean = 1.00000
Variance = 1.00000
```



The three distributions are histogrammed using the same bins and are not normalized. We note that in the case where $n = 1$, we rebuild the original exponential distribution. This makes sense as we are sampling it directly and re-plotting it as a PDF as we take $N \rightarrow \inf$ samples. For larger Ns, we approach a normal distribution under the Central Limit Theorem. For $n = 10$ we already get close, however it is broad, skews left, and has a tail to the right. With $n = 100$ we see a sharper and more symmetric distribution that looks closer to a normal distibution!

I chose not to normalize the histogram as it makes the behaviour of $n = 1$ hard to see.

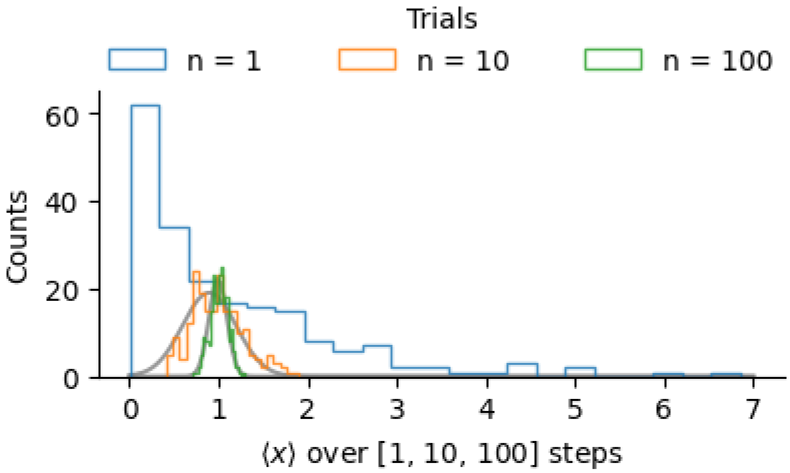
The width of the distributions decreases as we increase n that we average over, agreeing with theory. We can take this a step further by finding the FWHM of each!

In [5]:

```
x = np.linspace(0, 7, 200)
N = 200  # Trials
```

```python
ns = [1, 10, 100] #, 10, 100]#, 10, 100]  # Steps

# Generate trials
fig, ax = plt.subplots(1, 1, figsize=np.array([4.2,2.7]))
nbins = 41
bins = np.linspace(0,7,nbins+1)
bins=21
for n in ns:
    disp_array = []
    for trial in range(N):
        x_array = draw_from_PDF(n).copy()
        mean_disp = np.mean(x_array)
        stdev = np.std(x_array)
        var = stdev**2
        disp_array.append(mean_disp)

    disp_array = np.array(disp_array)
    # print(disp_array)
    counts, _bins = np.histogram(disp_array, bins=bins, density=False)
    ax.hist(_bins[:-1], _bins, weights=counts, label=f'n = {n}', histtype='step', alpha


    def gaus(x,a,x0,sigma):
        return a*np.exp(-(x-x0)**2/(2*sigma**2))
    try:
        popt, pcov = sp.optimize.curve_fit(gaus,_bins[:-1], counts, p0=[20, 1, 1])
        print(f'n = {n}\tsigma = {np.abs(popt[2])}')

        y_model = gaus(x, *popt)
        ax.plot(x, y_model, color='grey', alpha=0.8, zorder=1)

    except Exception as e:
        print(f'{n} failed fit')
        print(e)


ax.set(xlabel=r'$\langle x\rangle$ over'+f' {ns} steps', ylabel='Counts')
ax.legend(framealpha=0, title='Trials', bbox_to_anchor=(0, 1.02, 1, 0.2), loc="lower le
                  mode="expand", borderaxespad=0, ncol=3)
hp.despine(ax)
plt.tight_layout()
ax.set(facecolor = "none")
plt.savefig('plots/A1_Q5_distributionOfDisplacements_fit.pdf', bbox_inches='tight')
plt.savefig('plots/A1_Q5_distributionOfDisplacements_fit.png', bbox_inches='tight', dpi
plt.show()
```

```
1 failed fit
Optimal parameters not found: Number of calls to function has reached maxfev = 800.
n = 10  sigma = 0.30106818585991935
n = 100 sigma = 0.10008791977025665
```



We see:

$\sigma_{n=1} = 2.7$,

$\sigma_{n=10} = 0.33$,

$\sigma_{n=100} = 0.10 \approx \sigma_{n=10} \frac{\sqrt{10}}{\sqrt{100}}$,

so while the $n = 1$ case doesn't follow the fit for the Gaussian distribution, the $n = 10$ and $n = 100$ distributions seem to follow the scaling low of their relative widths falling as $1/\sqrt{N}$!

Q6

> A helium atom in a gas moves equal distances l between collisions with equal probability in any direction. After a total of N such displacements, what is the root mean square displacement $\sqrt{\langle \Delta R^2 \rangle}$ from its starting point? Treat the problem as 3 dimensional. After N collisions, the vector displacement is $\vec{R} = \sum_{n=1}^{N} l\vec{r}_i$, where the direction of $\vec{r}_i$ is random (Adapted from Reif.).

The atom can move in any direction in 3D space by distance l. Lets start by writing ΔR^2 as

$$\Delta R^2 = \left(\vec{R} - \langle\vec{R}\rangle\right)^2 = \left(\sum_{r=i}^{N} l\vec{r}_i - \langle\vec{R}\rangle\right)^2 = \left(\sum_{i=1}^{N}\Delta(l\vec{r}_i)\right), \text{ then}$$

$$\begin{aligned}
\langle\Delta R^2\rangle &= \left\langle \sum_{i=1}^{N}\Delta(l\vec{r}_i)\cdot\sum_{j=1}^{N}\Delta(l\vec{r}_j)\right\rangle \\
&= \left\langle \sum_{i=1,i=l}^{N}\Delta(l\vec{r}_i)^2\right\rangle + \text{\textbackslash cancel}\cancel{\left\langle \sum_{i=1,j=1,i\neq j}^{N}\Delta(l\vec{r}_i)\cdot\Delta(l\vec{r}_j)\right\rangle}, \; i \text{ and } j \text{ are independant vari} \\
&= N\left\langle\Delta(l\vec{r}_i)^2\right\rangle \\
&= N\left\langle(l\vec{r}_i - \langle\vec{R}\rangle)^2\right\rangle = N\sigma^2 \\
&= Nl^2\left(\langle\vec{r}_i^{\,2}\rangle - \langle\vec{r}_i\rangle^2\right) \\
&= Nl^2\,(1-0)
\end{aligned}$$

$$\therefore \text{ The RMS displacement is } \sqrt{\langle\Delta R^2\rangle} = \sqrt{N}l$$

Formatting cut off this text. It reads:

i and j are independant variables, therefore average to zero when multiplied together .

I plan to not use a Jupyter Notebook for future assignments, glad I tried it though