In [1]:
```
%pylab inline
from scipy import stats

from scipy.stats import binom

from scipy.stats import poisson

from scipy.stats import norm
```

Populating the interactive namespace from numpy and matplotlib

In [2]:
```
#Austin Jones
#Matthew Katz
#PHY 80 Winter 2020
#Wednesday, February 11
#Lab 9 Central Limit Theorem
```

In [3]:
```
NEXP = 1000000
NBINS = 60
MAX = 3
r = np.random.uniform(low=-1,high=1.0,size=NEXP)
h,edges = np.histogram(r,bins = NBINS,range =(-MAX,MAX))
cbins = (edges[:-1] + edges[1:])/2.0
plt.plot(cbins,h,"k-")

plt.xlabel("$x$")
plt.ylabel("Entries")
plt.show()
print("Figure 9.1 from Sampling from the uniform distribution")
```

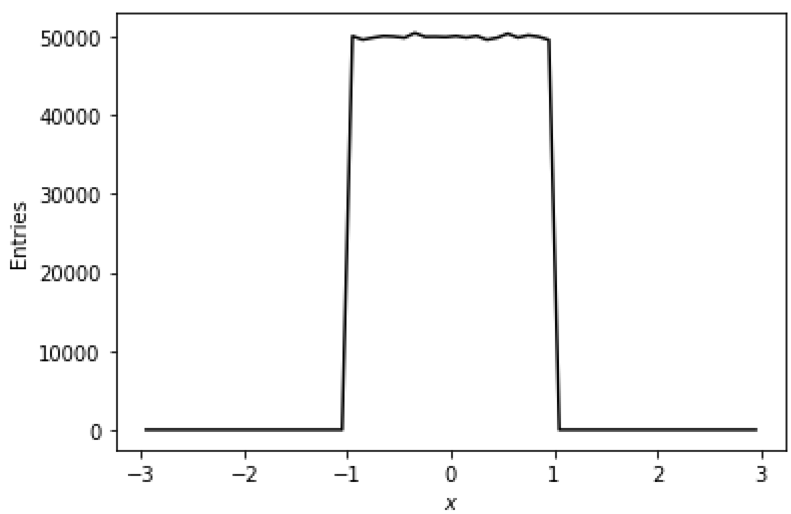


Figure 9.1 from Sampling from the uniform distribution

In [4]:
```
NEXP = 2000000
NBINS = 60
MIN =0
MAX = 10
MEAN = 5.0
SIGMA = 1.5
```

```
r = np.random.normal(loc= MEAN,scale = SIGMA,size = NEXP)
h,edges = np.histogram(r,bins=NBINS,range=(MIN,MAX))
cbins = (edges[:-1] + edges[1:])/2.0
plt.plot(cbins,h,"k-",label = "Monte Carlo")

binsize = float(MAX-MIN)/NBINS
x = np.linspace(MIN,MAX,100)
y = NEXP*binsize*stats.norm.pdf(x,loc=MEAN,scale=SIGMA)
plt.plot(x,y,"r--",label="PDF")
plt.xlabel("$x$")
plt.ylabel("Entries")
plt.semilogy()

plt.legend()
plt.show()
print("Figure 9.2: Sampling from the Gaussian distribution and comparison with t
```

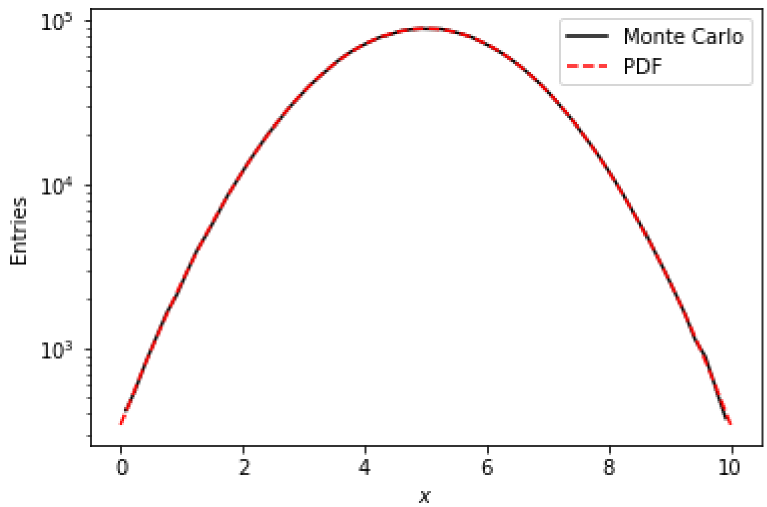
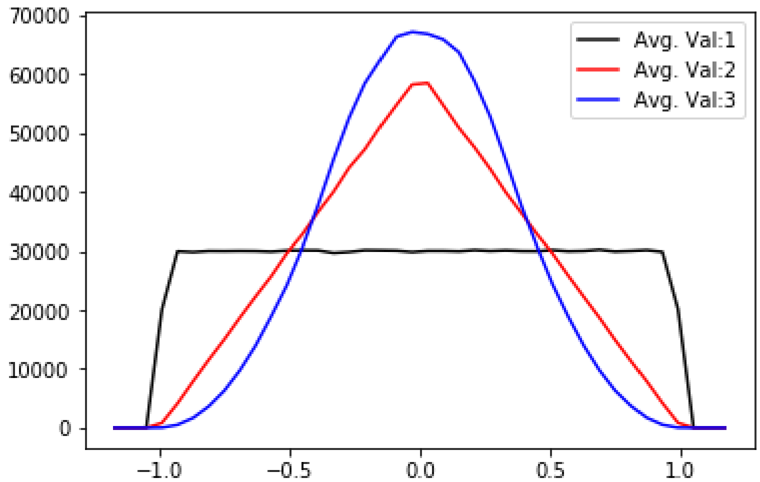


Figure 9.2: Sampling from the Gaussian distribution and comparison with the Gaus
sian PDF

In [5]:

```
#Demonstration of the Central Limit Theorem
#Jupyter Notebook 9.1
NEXP = 1000000
NBINS = 40
MIN =-1.2
MAX = 1.2

#average value of 1
NAVG = 1
r = np.random.uniform(low=-1.0,high=1.0,size=(NEXP,NAVG))
x1 = np.sum(r,axis=1)/float(NAVG)
h,edges = np.histogram(x1,bins=NBINS,range=(MIN,MAX))
cbins = (edges[:-1] + edges[1:])/2.0
plt.plot(cbins,h,"k-",label = "Avg. Val:1")

#average value of 2
NAVG = 2
r = np.random.uniform(low=-1.0,high=1.0,size=(NEXP,NAVG))
x2 = np.sum(r,axis=1)/float(NAVG)
h,edges = np.histogram(x2,bins=NBINS,range=(MIN,MAX))
cbins = (edges[:-1] + edges[1:])/2.0
plt.plot(cbins,h,"r-",label = "Avg. Val:2")
```

```
#average value of 3
NAVG = 3
r = np.random.uniform(low=-1.0,high=1.0,size=(NEXP,NAVG))
x3 = np.sum(r,axis=1)/float(NAVG)
h,edges = np.histogram(x3,bins=NBINS,range=(MIN,MAX))
cbins = (edges[:-1] + edges[1:])/2.0
plt.plot(cbins,h,"b-",label = "Avg. Val:3")

plt.legend()
plt.show()
print("Notebook 9.1")
print("Demonstration of the Central Limit Theorem of three  Averages")
```

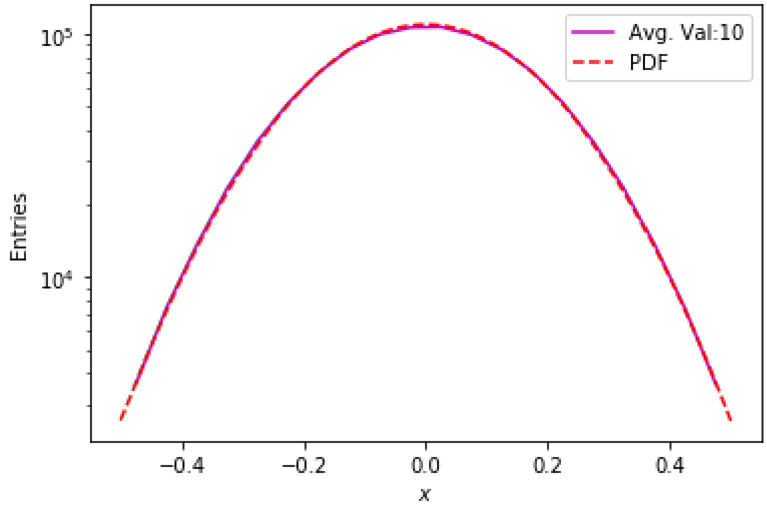



```
Notebook 9.1
Demonstration of the Central Limit Theorem of three  Averages
```

In [6]:

```
#Jupyter Notebook 9.2
NEXP = 1000000
NBINS = 20
MIN =-0.5
MAX = 0.5

#average value of 10
NAVG = 10
r = np.random.uniform(low=-1.0,high=1.0,size=(NEXP,NAVG))
x1 = np.sum(r,axis=1)/float(NAVG)
h,edges = np.histogram(x1,bins=NBINS,range=(MIN,MAX))
cbins = (edges[:-1] + edges[1:])/2.0
plt.plot(cbins,h,"m-",label = "Avg. Val:10")

MEAN = np.mean(x1)
SIGMA = np.var(x1)**0.5

binsize = float(MAX-MIN)/NBINS
x = np.linspace(MIN,MAX,100)
y = NEXP*binsize*stats.norm.pdf(x,loc=MEAN,scale=SIGMA)

plt.plot(x,y,"r--",label="PDF")
plt.xlabel("$x$")
plt.ylabel("Entries")
plt.semilogy()

plt.legend()
plt.show()
```

```
print("Notebook 9.2")
print("Demonstration of the Central Limit Theorem of Average 10")
```



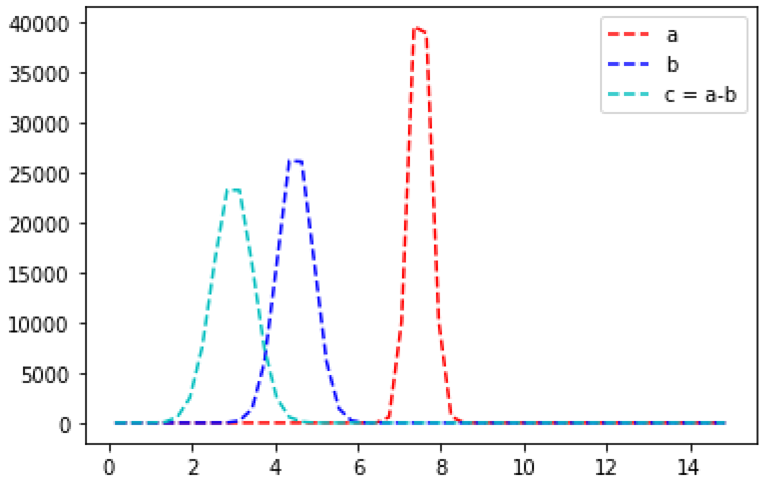

```
Notebook 9.2
Demonstration of the Central Limit Theorem of Average 10
```

In [7]:

```
#Propagation of Uncertanties
#Jupyter Notebook 9.3

#arbitrary values for a,b, and their uncertainties

NEXP = 100000
NBINS = 50
MIN =0.0
MAX = 15

a = 7.5
ø_a = 0.24

b = 4.5
ø_b = 0.42

c = a-b
ø_c = (ø_a**2 + ø_b**2)**0.5

print(a,b,c)
print(ø_a,ø_b,ø_c)

#list of samples from a Gaussian Distribution
list_a = np.random.normal(loc= a,scale = ø_a,size = NEXP)
list_b = np.random.normal(loc= b,scale = ø_b,size = NEXP)

list_c = list_a - list_b

h_a,edges_a = np.histogram(list_a,bins = NBINS,range = (MIN,MAX))
cbins_a = (edges_a[:-1] + edges_a[1:])/2.0
plt.plot(cbins_a,h_a,"r--",label = "a")

h_b,edges_b = np.histogram(list_b,bins = NBINS,range = (MIN,MAX))
cbins_b = (edges_b[:-1] + edges_b[1:])/2.0
plt.plot(cbins_b,h_b,"b--",label = "b")

h_c,edges_c = np.histogram(list_c,bins = NBINS,range = (MIN,MAX))
```

```
cbins_c = (edges_c[:-1] + edges_c[1:])/2.0
plt.plot(cbins_c,h_c,"c--",label = "c = a-b")

plt.legend()
plt.show()

print("Notebook 9.3: Simulation of many measurements of the quantity c = a - b")
```

```
7.5 4.5 3.0
0.24 0.42 0.48373546489791297
```




```
Notebook 9.3: Simulation of many measurements of the quantity c = a - b
```

In [8]:

```
#Propagation of Uncertanties
#Jupyter Notebook 9.4

#arbitrary values for a,b, and their uncertainties

NEXP = 100000
NBINS = 50
MIN =0.0
MAX = 15

a = 6.3
ø_a = 0.36

b = 8.4
ø_b = 0.12

c = a/b
ø_c = (((ø_a/a)**2 + (ø_b/b)**2)**0.5)*c

print(a,b,c)
print(ø_a,ø_b,ø_c)
print()

#list of samples from a Gaussian Distribution
list_a = np.random.normal(loc= a,scale = ø_a,size = NEXP)
list_b = np.random.normal(loc= b,scale = ø_b,size = NEXP)

list_c = list_a / list_b

h_a,edges_a = np.histogram(list_a,bins = NBINS,range = (MIN,MAX))
cbins_a = (edges_a[:-1] + edges_a[1:])/2.0
plt.plot(cbins_a,h_a,"r--",label = "a")
```

```
h_b,edges_b = np.histogram(list_b,bins = NBINS,range = (MIN,MAX))
cbins_b = (edges_b[:-1] + edges_b[1:])/2.0
plt.plot(cbins_b,h_b,"b--",label = "b")

h_c,edges_c = np.histogram(list_c,bins = NBINS,range = (MIN,MAX))
cbins_c = (edges_c[:-1] + edges_c[1:])/2.0
plt.plot(cbins_c,h_c,"c--",label = "c = a/b")

print("Notebook 9.4: Simulation of many measurements of the quantity c = a/b")
```

```
6.3 8.4 0.75
0.36 0.12 0.044176131703O4636

Notebook 9.4: Simulation of many measurements of the quantity c = a/b
```