Python Practice 2

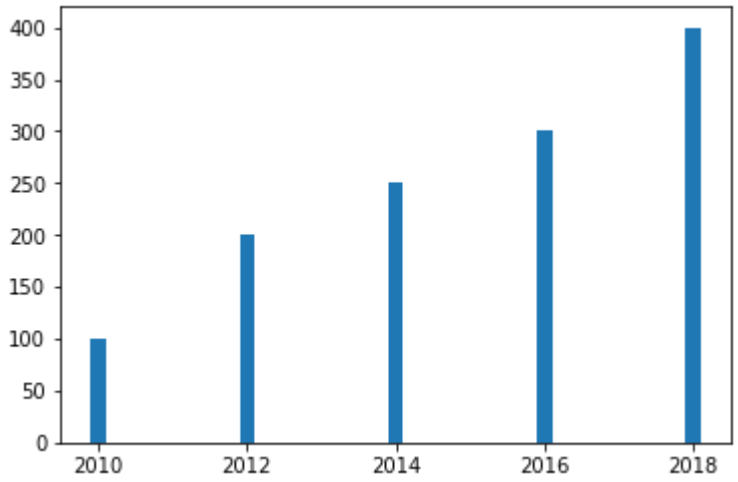
Probability and Statistics Programming (Sejong University)

Date: 2019.04.08 (By: S.M. Riazul Islam)

Ploting Bar Diagram

In [131]:
```
import numpy as np
import matplotlib.pyplot as plt
year=[2010, 2012, 2014, 2016, 2018]
visitors=[100, 200, 250, 300, 400]
plt.bar(x=year, height=visitors, width=0.2, align='center')
```

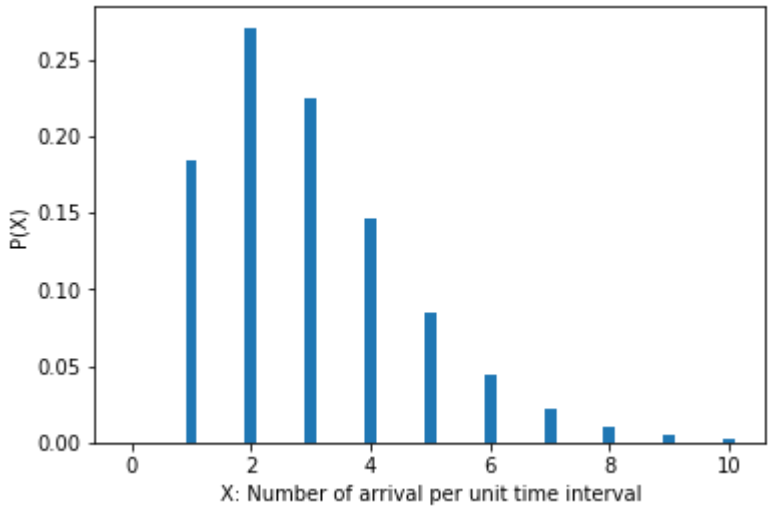
Out[131]: <BarContainer object of 5 artists>



Presenting Poisson pmf as a bar Diagram (Hint: RV is Discrete)

In [9]:
```
import numpy as np
import scipy.stats as sp
import matplotlib.pyplot as plt
rate=2
x=np.arange(11)
ppoisson=sp.poisson.pmf(rate,x)
plt.bar(x,ppoisson, width=0.2, align='center')
plt.xlabel("X: Number of arrival per unit time interval")
plt.ylabel("P(X)")
```

Out[9]:
```
Text(0, 0.5, 'P(X)')
```



Random Number Generation Following Normal Distribution

In [14]:
```
# A single random number generation
import numpy as np
np.random.normal(loc=3,scale=1) # loc=mean, scale=standard deviation
```

Out[14]:
```
3.145760919448254
```

In [15]:
```
# An array of random numbers generation
import numpy as np
np.random.normal(loc=3,scale=1, size=10)
```

Out[15]:
```
array([1.62477417, 3.60192751, 1.97912455, 3.49715414, 2.28988237,
       1.3978047 , 4.59500144, 4.29437805, 1.64832197, 0.98200888])
```

In [48]:
```
# Another way for normal random number generation
import scipy.stats as sp
a=sp.norm.rvs(loc=3, scale=1, size=10)
a
```

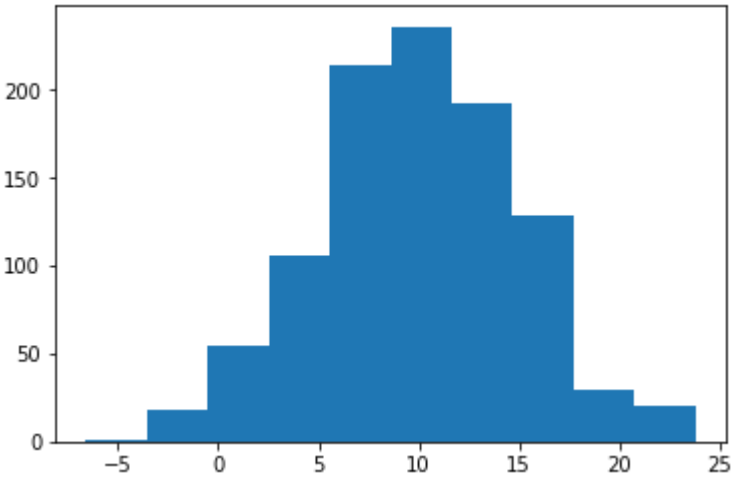
Out[48]:
```
array([3.334115  , 2.13262805, 3.27138645, 3.04678519, 4.9121847 ,
       3.75841243, 2.6080921 , 3.80310042, 4.44884336, 3.12645203])
```

In [26]:
```
# Check the mean
# An array of random numbers generation
import numpy as np
n=1000;   # Try with different values of n (n=10, 100, 1000)
a=np.random.normal(loc=3,scale=1, size=n) # try with different values of mean and
average=sum(a)/n
print("Calculated Mean: "+str(average))
```

```
Calculated Mean: 2.9734494514339387
```

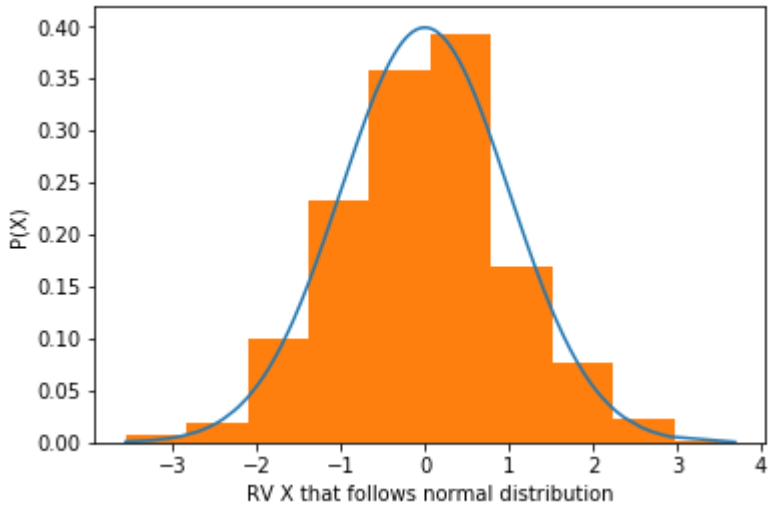
In [29]:
```
# Check Histogram
import numpy as np
import matplotlib.pyplot as plt
n=1000;   # Try with different values of n (n=10, 100, 1000)
a=np.random.normal(loc=10,scale=5, size=n) # try with different values of mean ar
plt.hist(a)
```

Out[29]:
```
(array([  1.,  18.,  55., 106., 214., 236., 192., 129.,  29.,  20.]),
 array([-6.59472134, -3.55578781, -0.51685429,  2.52207924,  5.56101276,
         8.59994629, 11.63887981, 14.67781334, 17.71674686, 20.75568039,
        23.79461391]),
 <a list of 10 Patch objects>)
```

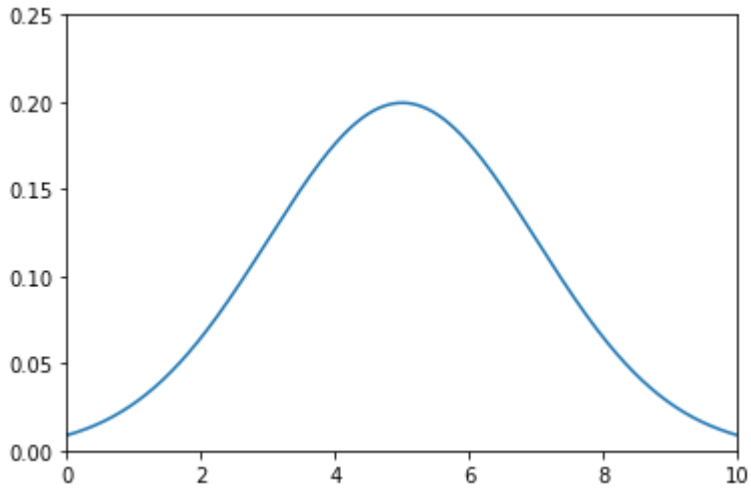


Plotting Probability of a Normal RV

In [7]:
```
# we will verify that the generated random numbers are follwing normal distribut

import numpy as np
import matplotlib.pyplot as plt
import scipy.stats as sp
x = sp.norm.rvs(loc=0, scale=1, size=1000)
px= sp.norm.pdf(np.sort(x),loc=0,scale=1)
plt.plot(np.sort(x),px)
plt.hist(x, density=True)
plt.xlabel("RV X that follows normal distribution")
plt.ylabel("P(X)")
plt.show()
```



In [2]:
```
# Just for Graphical representations

import matplotlib.pyplot as plt
import scipy.stats as sp
import numpy as np

xmin=0
xmax = 10

mean = 5
sd = 2.0

x = np.linspace(xmin, xmax, 1000) # x is not follwoing normal distribution
y = sp.norm.pdf(x,mean,sd)

plt.plot(x,y)
plt.xlim(xmin,xmax)
plt.ylim(0,0.25)
plt.show()
```



Calculation of CDF for Normal RV

In [127]:
```
import scipy.stats as sp
import numpy as np
x=0;                    # Try with different values of x, mean and sd
mean=0
sd=1
sp.norm.cdf(x,loc=mean,scale=sd)
```

Out[127]: 0.5

Calculation of Some Probabilities of a normal RV using CDF

In [14]:
```
# P(X>=12) where X~N(10,4)
import scipy.stats as sp
import numpy as np
x=12
mean=10
sd=2
1-sp.norm.cdf(x,loc=mean,scale=sd)
```

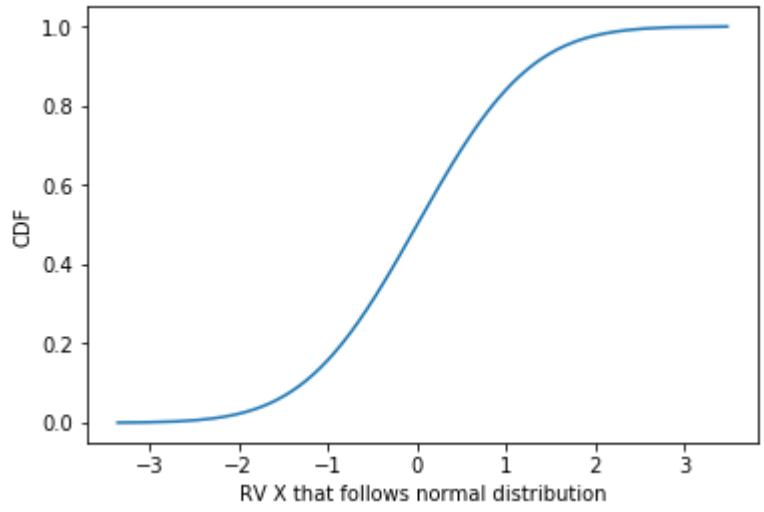
Out[14]: 0.15865525393145707

In [15]:
```
# P(8 <= X <= 12) where X~N(10,4)
import scipy.stats as sp
import numpy as np
x1=8
x2=12
mean=10
sd=2
sp.norm.cdf(x2,loc=mean,scale=sd)-sp.norm.cdf(x1,loc=mean,scale=sd)
```

Out[15]: 0.6826894921370859

Plotting CDF for Normal RV

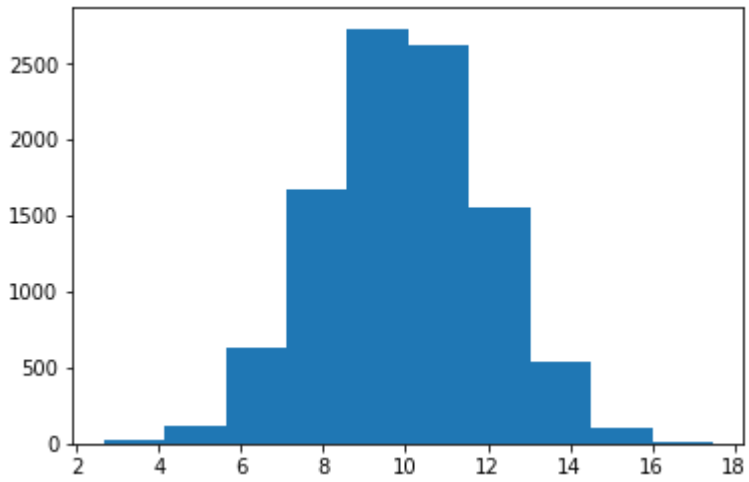
In [8]:
```
import scipy.stats as sp
import numpy as np
import matplotlib.pyplot as plt
mean=0
sd=1
x = sp.norm.rvs(loc=mean, scale=sd, size=1000)
px = sp.norm.cdf(np.sort(x),loc=mean, scale=sd)
plt.plot(np.sort(x),px)
plt.xlabel("RV X that follows normal distribution")
plt.ylabel("CDF")
plt.show()
```



Generating non-standard RVs using standard RVs

In [29]:
```
import numpy as np
import scipy.stats as sp
import matplotlib.pyplot as plt
x=sp.norm.rvs(loc=0,scale=1, size=10000) # X is a standard normal RV
#plt.hist(x)
mean=10
sd=2
y=mean+np.multiply(sd,x) # Y is a non-standard normal RV
plt.hist(y)
```

Out[29]:
```
(array([  19.,  119.,  631., 1672., 2731., 2621., 1553.,  539.,  101.,
         14.]),
 array([ 2.64680671,  4.13067575,  5.61454478,  7.09841382,  8.58228285,
        10.06615189, 11.55002092, 13.03388996, 14.51775899, 16.00162803,
        17.48549706]),
 <a list of 10 Patch objects>)
```



Assume that the test score of a college entrance exam fits a normal distribution. Furthermore, the mean test scoore is 72, and the standard deviation is 15.2. What is the percentage of students scoring 84 or more in the exam?

In [45]:
```
import numpy as np
import scipy.stats as sp

1-sp.norm.cdf(84, loc=72, scale=15.2)
```

Out[45]:
```
0.21491760231127244
```

Suppose there are twelve multiple choice questions in a class quiz. Each question has five possible answers, and only one of them is correct. What is the probability that at most four answers will be correct if a student attempts to answer every question at random?

In [33]:
```
# P(X<=4)

import numpy as np
import scipy.stats as sp
n=12
p=0.2

#sp.binom.pdf(x,n,p)
sp.binom.pmf(0,12,0.2)+sp.binom.pmf(1,12,0.2)+sp.binom.pmf(2,12,0.2)+sp.binom.pm

#or
#sp.binom.cdf(4,12,0.2)
```

Out[33]: 0.9274445004799988

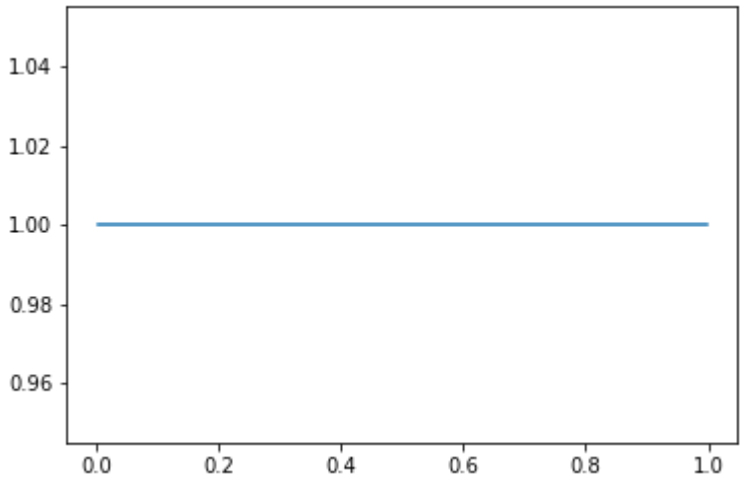
Suppose that there are twelve cars crossing a bridge per minute on average. What is the probability of having seventeen or more cars crossing the bridge in a particular minute?

In [35]:
```
# P(X >= 17)

import numpy as np
import scipy.stats as sp
rate=12

#sp.poisson.pmf(x,rate)
#sp.poisson.cdf(x,rate)
1-sp.poisson.cdf(16,rate)
```

Out[35]: 0.10129100743983788

Random Number Generation using Uniform Distribution

In [43]:
```
# Standard numbers [0,1]

import numpy as np
import scipy.stats as sp
import matplotlib.pyplot as plt
x=sp.uniform.rvs(size=1000)
x=np.sort(x)
px=sp.uniform.pdf(x)
plt.plot(x,px)
plt.show()
```



In [44]:
```
# numbers [loc, loc+scale]
import numpy as np
import scipy.stats as sp
import matplotlib.pyplot as plt
x=sp.uniform.rvs(loc=10, scale=20, size=1000)
x=np.sort(x)
px=sp.uniform.pdf(x)
plt.plot(x,px)
plt.show()
```