In [10]:
```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
import statistics
```

In [3]:
```python
# 1
x = (6, 7, 5, 7, 7, 8, 7, 6, 9, 7, 4, 10, 6, 8, 8, 9, 5, 6, 4, 8)
print("Mean: ", np.mean(x))
print("Median: ", np.median(x))
print("Mode: ", statistics.mode(x))
print("Standard Deviation: ", np.std(x))
```

```
Mean:  6.85
Median:  7.0
Mode:  7
Standard Deviation:  1.5898113095584647
```

In [4]:
```python
# 2
y = (28, 122, 217, 130, 120, 86, 80, 90, 140, 120, 70, 40,
     145, 113, 90, 68, 174, 194, 170,100, 75, 104, 97, 75,
     123, 100, 75, 104, 97, 75, 123, 100, 89, 120, 109)
print("Mean: ", np.mean(y))
print("Median: ", np.median(y))
print("Mode: ", statistics.mode(y))
print("Standard Deviation: ", np.std(y))
```

```
Mean:  107.51428571428572
Median:  100.0
Mode:  75
Standard Deviation:  38.77287080168403
```

In [5]:
```python
# 3
x = (0, 1, 2, 3, 4, 5)
y = (0.09, 0.15, 0.40, 0.25, 0.10, 0.01)
Mean = ((0*0.09) + (1*0.15) + (2*0.40) + (3*0.25) + (4*0.10) + (5*0.01))
Var = ((0-2.15)**2)*0.09 + ((1-2.15)**2)*0.15 + ((2-2.15)**2)*0.40 + ((3-2.15)**2)*0.25 + ((4-2.15)**2)*0.10 + ((5-2.15)**2)*0.01
print("Mean: ", Mean)
print("Variance: ", Var)
```

```
Mean:  2.15
Variance:  1.2275
```

In [6]:
```python
# 4

from scipy import integrate
# PDF (d) = 20e−20(d−12.5)
# d ≥ 12.5
PDF=lambda d:20*(np.exp((-20*(d-12.5))))
x = 12.6
P_x=integrate.quad(PDF,12.6,np.inf)
y = 11
CDF=integrate.quad(PDF,-np.inf,y)
print(f"Proportion of Parts need to scrapped when d >12.6mm is :{P_x[0]}")
print(f"CDF when d= 11mm is:{CDF[0]}")
print(f"Proportion of CDF when d>12.5mm is : {integrate.quad(PDF,12.5,np.inf)[0]}")

# Conclusion- the function is valid only when d >= 12.5
# When d<12.5, the part can be reworked to 12.5 so no scrap in this case.
# PDF is not defined for d=11
```

```
Proportion of Parts need to scrapped when d >12.6mm is :0.13533528323661398
CDF when d= 11mm is:nan
Proportion of CDF when d>12.5mm is : 1.0000000000000024
<ipython-input-6-ec3666497b53>:6: RuntimeWarning: overflow encountered in exp
  PDF=lambda d:20*(np.exp((-20*(d-12.5))))
<ipython-input-6-ec3666497b53>:10: IntegrationWarning: The maximum number of subdivisions (50) has been achieved.
  If increasing the limit yields no improvement it is advised to analyze
  the integrand in order to determine the difficulties.  If the position of a
  local difficulty can be determined (singularity, discontinuity) one will
  probably gain from splitting up the interval and calling the integrator
  on the subranges.  Perhaps a special-purpose integrator should be used.
  CDF=integrate.quad(PDF,-np.inf,y)
```

In [14]:
```python
# 5

import scipy.special
x = 0.3
y = 0.7
df=pd.DataFrame({'a':[int(i) for i in range(7)],
                 'B_a':[scipy.special.comb(6,i)*(x**i)*(y**(6-i)) for i in range(7)]})
print(df.iloc[2])
df['Expected value']=df['a']*df['B_a']
mean=np.round(df['Expected value'].sum())
print('mean = {}'.format(mean))
df['variance']=df['B_a']*(df['a']-mean)**2
std=np.sqrt(df['variance'].sum())
print(f"Standard Deviation : {np.round(std)}")
```

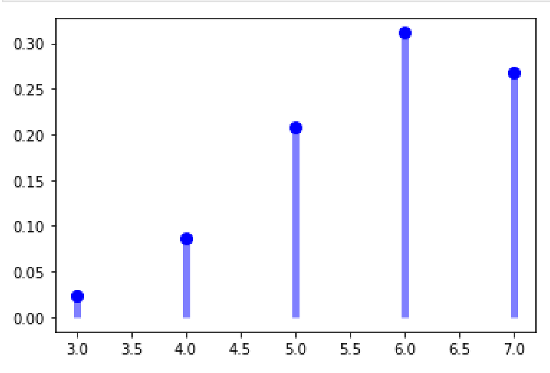
```
a      2.000000
B_a    0.324135
Name: 2, dtype: float64
mean = 2.0
Standard Deviation : 1.0
```

In [15]:
```python
# 6

from scipy.stats import binom
import numpy as np

print(f"Probability of each of them solving 5 questions correctly is:{binom.pmf(5,8,0.75)*binom.pmf(5,12,0.45)}")
print(f"Probability of each of them solving 4,6 questions correctly is:{binom.pmf(4,8,0.75)*binom.pmf(6,12,0.45)}")
```

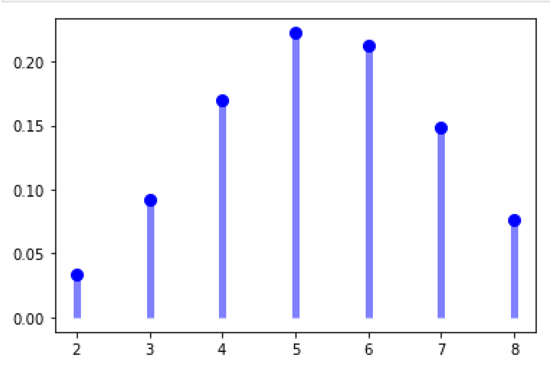
```
Probability of each of them solving 5 questions correctly is:0.04619989057299213
Probability of each of them solving 4,6 questions correctly is:0.018374956477894576
```

In [16]:
```python
def binom_plot(n,p,):
    fig,ax=plt.subplots(1,1)
    x = np.arange(binom.ppf(0.01, n, p),binom.ppf(0.99, n, p))
    ax.plot(x, binom.pmf(x, n, p), 'bo', ms=8, label='binom pmf')
    ax.vlines(x, 0, binom.pmf(x, n, p), colors='b', lw=5, alpha=0.5)
```

In [17]:
```python
# Gaurav
binom_plot(8,0.75)
```

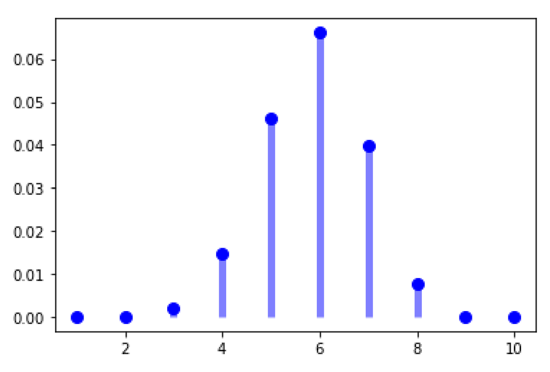


In [18]:
```python
# Barakha
binom_plot(12,0.45)
```



In [19]:
```python
fig,ax=plt.subplots(1,1)
x = np.arange(1,11)
ax.plot(x, binom.pmf(x,8,0.75)*binom.pmf(x,12,0.45), 'bo', ms=8, label='binom pmf')
ax.vlines(x, 0, binom.pmf(x,8,0.75)*binom.pmf(x,12,0.45), colors='b', lw=5, alpha=0.5)
#maximum combined probability observed at 6th question
```

Out[19]:
```
<matplotlib.collections.LineCollection at 0x2328f4cf100>
```



In [20]:
```python
from scipy.stats import binom
binom.pmf(5,8,0.75)*binom.pmf(5,12,0.45)
```

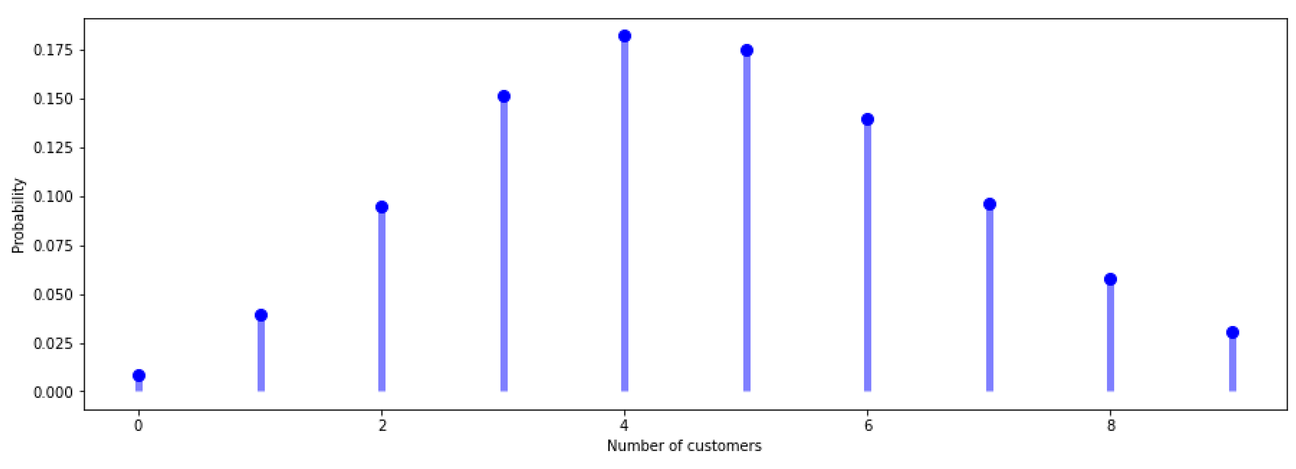
Out[20]:
```
0.04619989057299213
```

In [32]:
```python
# 7

# Average no of customers per minute = 72/60
Avg = 72/60
mu = 4*(72/60) #customers come per 4 minutes
print(f"Probability of 5 cutomers in 4 minutes is : {poisson.pmf(k=5,mu=mu)}")
print(f"Probability of not more than 3 customers in 4 minutes is : {poisson.pmf(k=3, mu=mu)}")
print(f'Probability of more than 3 customers in 4 minutes is : {1-poisson.cdf(k=3,mu=mu)}')
```

```
Probability of 5 cutomers in 4 minutes is : 0.17474768364388296
Probability of not more than 3 customers in 4 minutes is : 0.15169069760753714
Probability of more than 3 customers in 4 minutes is : 0.7057700835034357
```

In [33]:
```python
x = list(range(0,10))
fig,ax = plt.subplots(1,1,figsize=(15,5))
ax.plot(x, poisson.pmf(x,mu), 'bo', ms=8, label='poisson pmf')
ax.vlines(x, 0, poisson.pmf(x, mu), colors='b', lw=5, alpha=0.5)
plt.xlabel('Number of customers')
plt.ylabel('Probability')
```

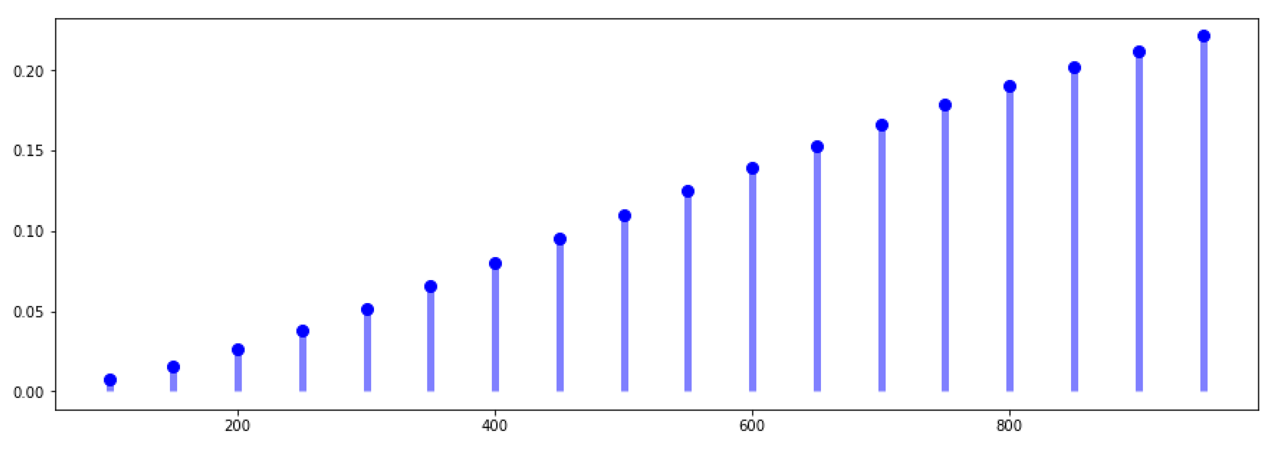
Out[33]:
```
Text(0, 0.5, 'Probability')
```



In [34]:
```python
# 8

from scipy.stats import poisson
# Entering rate  =77/minute
# Error rate = 0.1 per minute
# No of errors per word = 0.1/77
unit_mu=0.1/77
def mu(n):
    return n * unit_mu
print(f"Probability of commiting 2 errors in 455 words: {poisson.pmf(2,mu=mu(455))}")
print(f"Probability of commiting 2 errors in 1000 words: {poisson.pmf(2,mu=mu(1000))}")
print(f"Probability of commiting 2 errors in 255 words: {poisson.pmf(2,mu=mu(255))}")
x=range(100,1000,50)
mu=[i*unit_mu for i in x]
fig,ax = plt.subplots(1,1,figsize=(15,5))
ax.plot(x,poisson.pmf(2,mu), 'bo', ms=8, label='poisson pmf')
ax.vlines(x,0, poisson.pmf(2,mu), colors='b', lw=5, alpha=0.5)
#As the number of words increase probability of getting errors increases
```

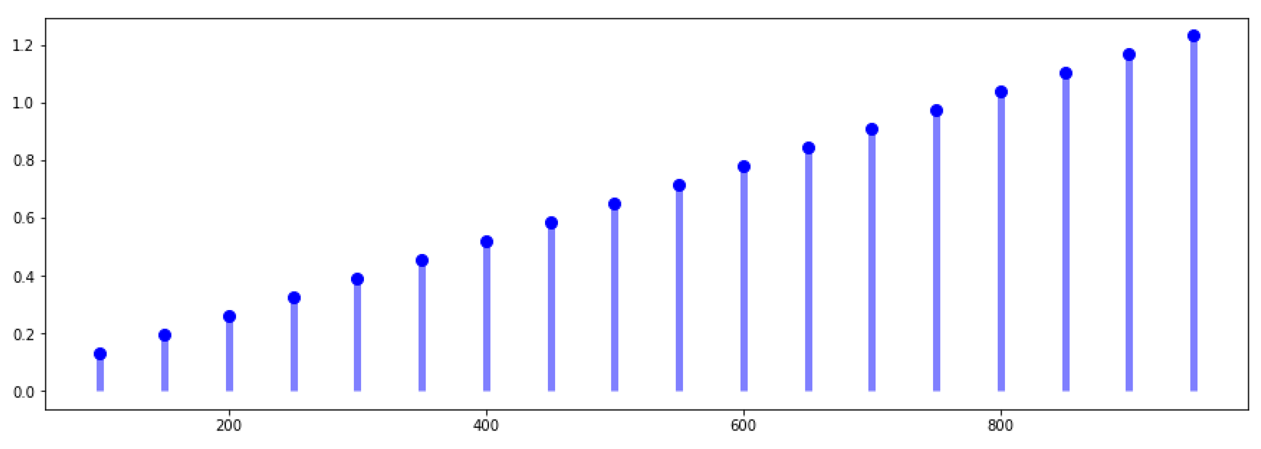
```
Probability of commiting 2 errors in 455 words: 0.09669027375144444
Probability of commiting 2 errors in 1000 words: 0.23012815007300153
Probability of commiting 2 errors in 255 words: 0.039377135392854104
```

Out[34]:
```
<matplotlib.collections.LineCollection at 0x2328f77b640>
```



In [35]:
```python
fig,ax = plt.subplots(1,1,figsize=(15,5))
ax.plot(x,mu, 'bo', ms=8, label='poisson pmf')
ax.vlines(x,0,mu, colors='b', lw=5, alpha=0.5)
#Value of mu keeps on increasing with number of words
```

Out[35]:
```
<matplotlib.collections.LineCollection at 0x23290808820>
```



In [36]:
```python
# 10

from scipy.stats import norm
def P(z,b=-np.inf) :
    return integrate.quad(norm.pdf,b,z)[0]

print('P(Z>1.26) = %.5f'%(1-P(1.26)))
print('P(Z<-0.86) = %.5f'%P(-0.86))
print('P(Z>-1.37) = %.5f'%(1-P(-1.37)))
print('P(−1.25 < Z < 0.37) = %.5f'%P(0.37,b=-1.25))
print('P(Z ≤ −4.6) = %.5f'%P(-4.6))
print('P(Z>z)=0.05 is %.2f'%(-1*norm.ppf(0.05)))
print('P(−z < Z < z) = 0.99 is %.2f'%(abs(norm.ppf(0.005))))
```

```
P(Z>1.26) = 0.10383
P(Z<-0.86) = 0.19489
P(Z>-1.37) = 0.91466
P(−1.25 < Z < 0.37) = 0.53866
P(Z ≤ −4.6) = 0.00000
P(Z>z)=0.05 is 1.64
P(−z < Z < z) = 0.99 is 2.58
```

In [37]:
```python
# 11

mean = 10
std = np.sqrt(4)

def I(z, b=-np.inf):
    z = (z-mean)/std
    return integrate.quad(norm.pdf,b,z)[0]
print(f"Probability of current exceeding 13mA is: {1-I(13)}")
print(f"Probability of current is between 9 mA and 11 mA is : {1-I(11,b=9)}")
```

```
Probability of current exceeding 13mA is: 0.06680720126885797
Probability of current is between 9 mA and 11 mA is : 1.3085375387259144
```

In [40]:
```python
# 12

mean_dia=0.2508
std_dia=0.0005
#specified dia in the range of 0.2485<d<0.2515
#case-1 if mean_dia=0.2508
def I(mean,std,a,b) :
  #gives P(Z<=x)
  a=(a-mean)/std
  b=(b-mean)/std
print(f"Proportion of shafts with dia in range of 0.2485<d<0.2515 when mean diameter:{0.2508,I(0.2508,0.0005,0.2485,0.2515)}")
print(f"Proportion of shafts with dia in range of 0.2485<d<0.2515 when mean diameter:{0.2500,I(0.2500,0.0005,0.2485,0.2515)}")

# Conclusion- Within the range of 0.2485<d<0.2515 A manufacturing process with mean of 0.25 gives maximum proportion of
# required shafts, there by reducing amount of scrap and reprocessing time.
```

```
Proportion of shafts with dia in range of 0.2485<d<0.2515 when mean diameter:(0.2508, None)
Proportion of shafts with dia in range of 0.2485<d<0.2515 when mean diameter:(0.25, None)
```

In [ ]: