Question :1

The expression for $P_{m,p}(n) \hspace{0.1cm} is \hspace{0.2cm} \frac{^mC_p.^{n-m}C_{m-p}}{^nC_m}$

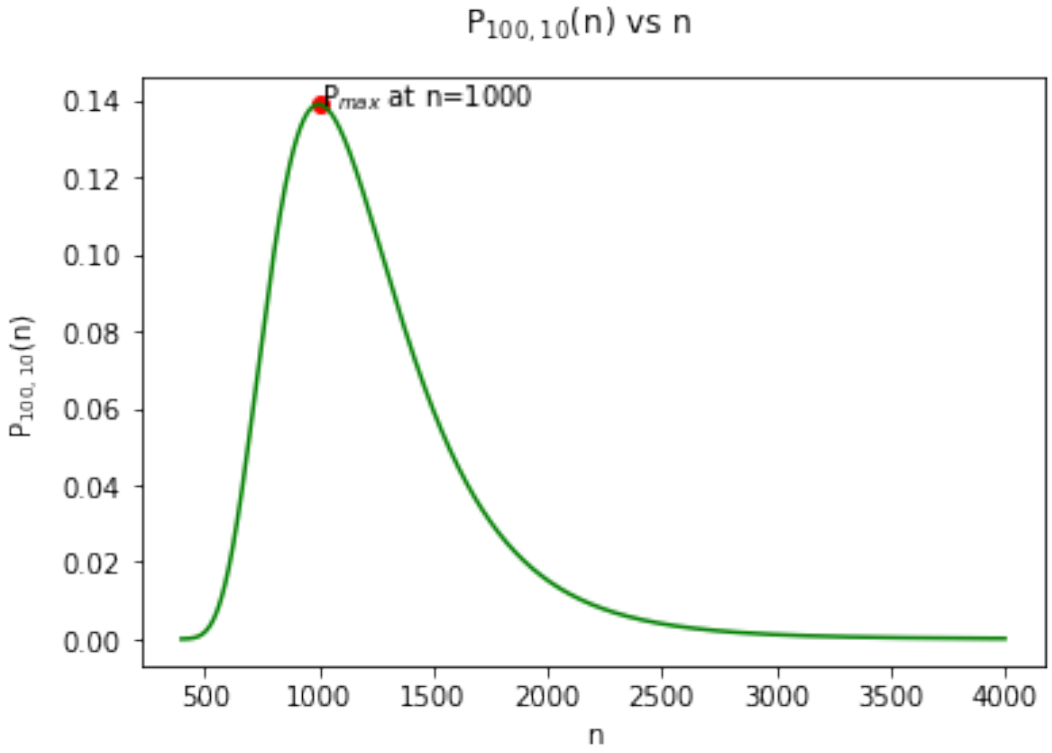
The expression for $P_{m,p}(n)$ is $\frac{^mC_p.^{n-m}C_{m-p}}{^nC_m}$

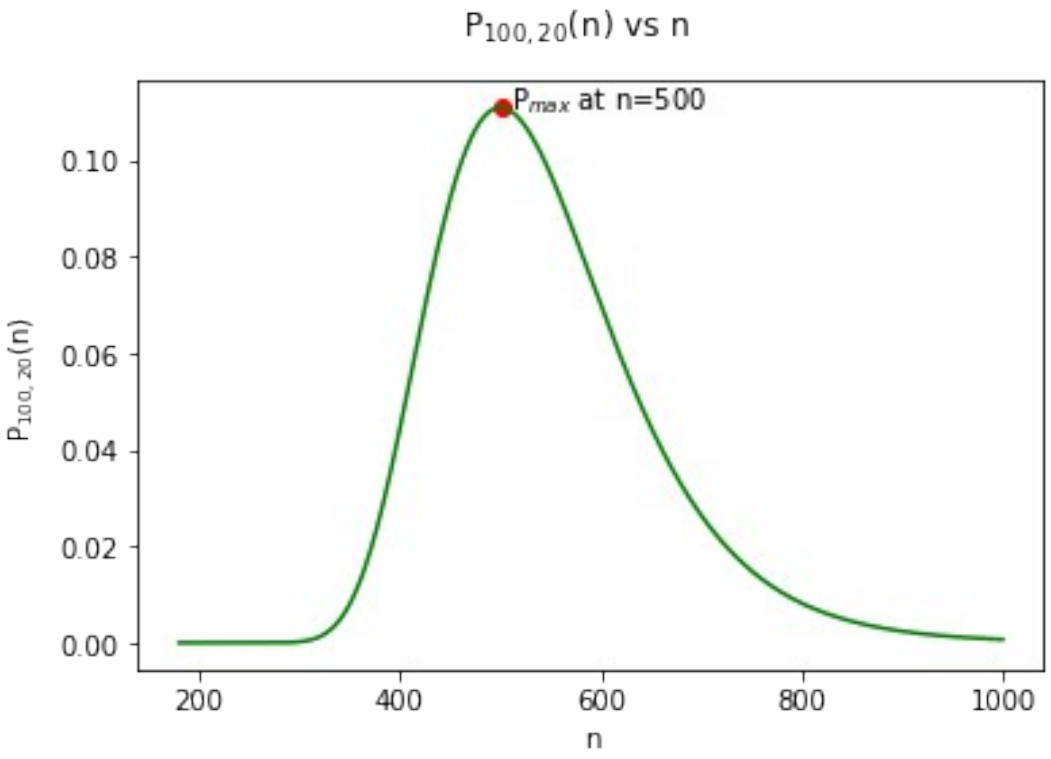
We will use above formulae to plot "probablity" vs "n"(no. of fishesh in lake) graph. And by using that graph we will find the value of "n" for which we have maximum probablity corresponting to given "m" and "p".

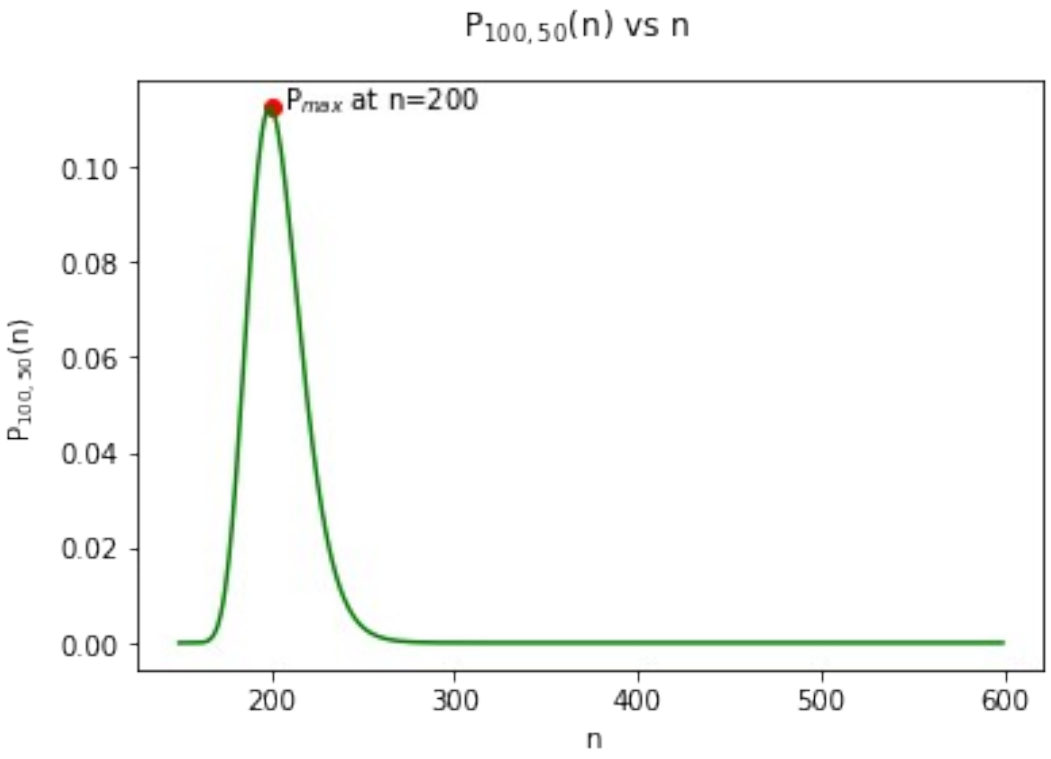
```
import numpy as np
import statistics
import matplotlib.pyplot as plt
import array as arr
import random
import operator as op
from functools import reduce

def ncr(n, r):
    r = min(r, n-r)
    numer = reduce(op.mul, range(n, n-r, -1), 1)
    denom = reduce(op.mul, range(1, r+1), 1)
    return (numer/denom)

m = 100
p = arr.array('b',[10,20,50,75])
n=np.array(range(400, 4000))


#a=ncr(m,p[r])
#b=ncr(n-m,m-p[r])
#c=ncr(n,m)
#Pmn = (a*b)/c

p1 = p[0] #P[0] corresponds to p=10
          #P[1] corresponds to p=20
          #P[2] corresponds to p=50
          #P[3] corresponds to p=75
arr = np.array([(ncr(m,p1)*ncr(i-m,m-p1))/ncr(i,m) for i in n ] )
plt.plot(n , arr , markersize=5,markeredgecolor="black",
markerfacecolor="black",color='green')
max_index=np.argmax(arr)+1
print(n[max_index])
plt.xlabel('n')
```

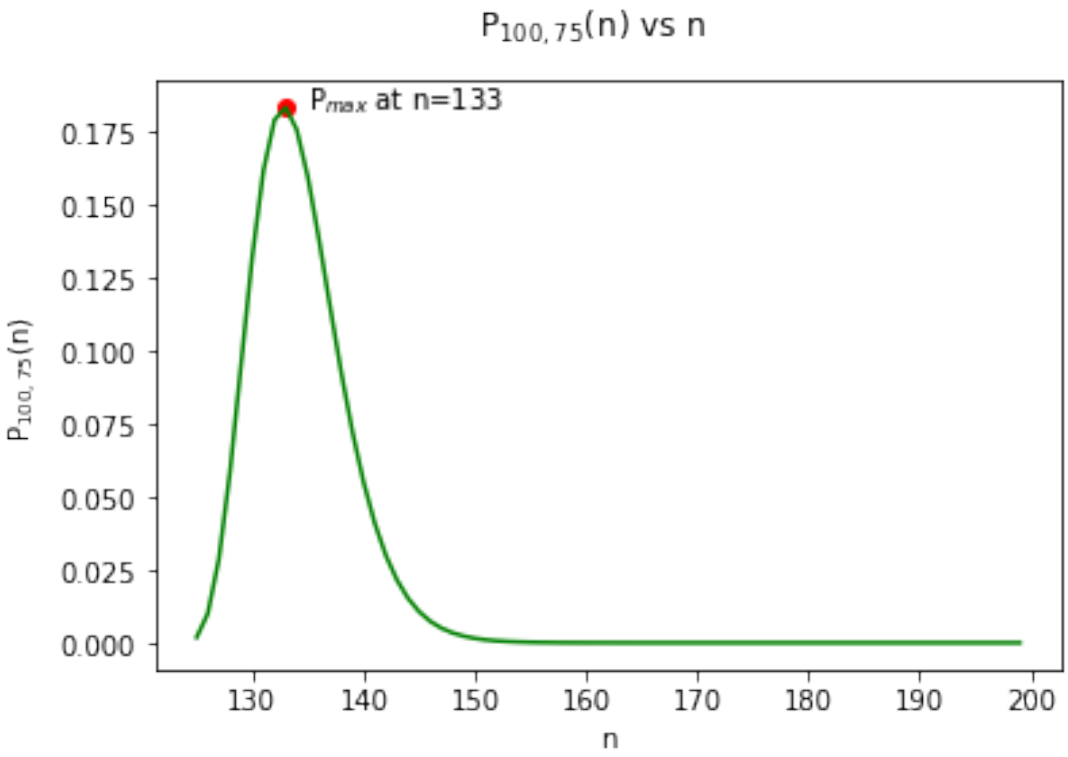
```
plt.ylabel('P$_{100,10}$(n)')
plt.suptitle('P$_{100,10}$(n) vs n')
plt.scatter(n[600],arr[600],color='red')
plt.annotate("P$_{max}$ at n=1000",(n[610],arr[600]))
plt.show()
```

```
1000
```




$P_{100,20}(n)$ vs n
P_{max} at n=500
0.10
0.08
0.06
0.04
0.02
0.00
$P_{100,20}(n)$
200
400
600
800
1000
n


$P_{100,50}(n)$ vs n
P_{max} at n=200
0.10
0.08
0.06
0.04
0.02
0.00
$P_{100,50}(n)$
200
300
400
500
600
n



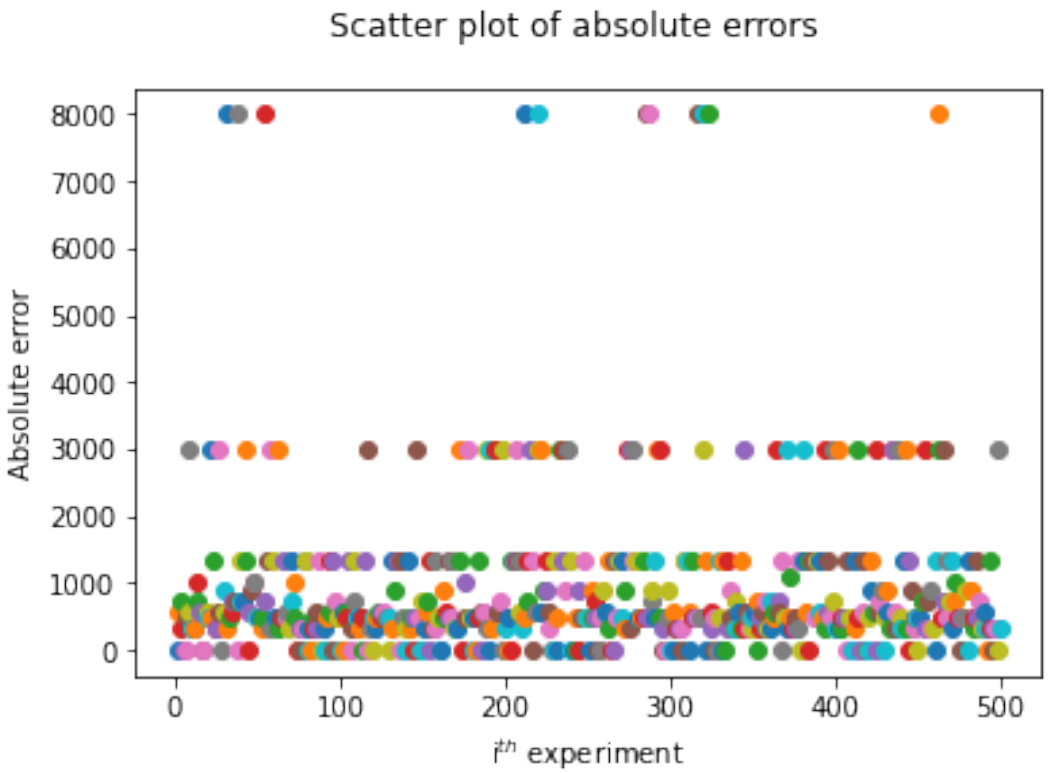
Question :2

When we will repeat this experiment 500 times then there will be the cases when P=0 i.e, no marked fish was found when we caught fish for the second time and in this case error is infinity.Therefore in this case mean of the error and varience of the error will be infinity.

If we ignore the above case then we will get a finite error.And the value of the errors we will get after this assumption are displayed after the following code.

```
arrerror = np.empty(501,dtype=int)
mean_error = 0
for t in range(1,501):

    fish = np.empty(2000, dtype=int)
    for i in range(0,2000):
        fish[i] = 0;

    #X = np.random.choice(fish, size=100, replace=False)
    #Y = X+1
    fish1 = np.empty(2000, dtype=int)
    for i in range(0,2000):
        fish1[i] = 0;
    # for y in range(0,100):
    # randomfish = random.randrange(len(fish))
```

```
    # fish1[i] = 1;
    #print(fish[500])
    arr=np.random.randint(0,2000,size=100);
    #print(arr)
    for i in range(0,100):
        fish[arr[i]] = 1;
    #for i in range(0,50):
    #    print(fish[i])
    arr1=np.random.randint(0,2000,size=100);
    #print(arr1)
    p = 0
    for i in range(0,100):
        if (fish[arr1[i]] == 1):
          p = p+1;
    #print(p)
    error1=0
    m=100
    n1=np.array(range(250, 11000))
    arr5 = np.array([(ncr(m,p)*ncr(i-m,m-p))/ncr(i,m) for i in n1 ] )
    if p !=0:
       max_index=np.argmax(arr5)+1
       error = n1[max_index] - 2000
    else :
         error = 0   #for case when P=0;
    arrerror[t] = error
    error1 = error1 + abs(error);
    #print(n1[max_index])
    #print(error)

    plt.scatter(t,abs(error),);
plt.xlabel('i$^{th}$ experiment')
plt.ylabel('Absolute error')
plt.suptitle('Scatter plot of absolute errors')
plt.show()

mean_error = error1/500
varience = 0
for f in range(1,501):
    z=(arrerror[f] - mean_error)
    varience = (z*z)/499 + varience

print("Mean Error is :",mean_error)
print("Sample Varience is :",varience)
```



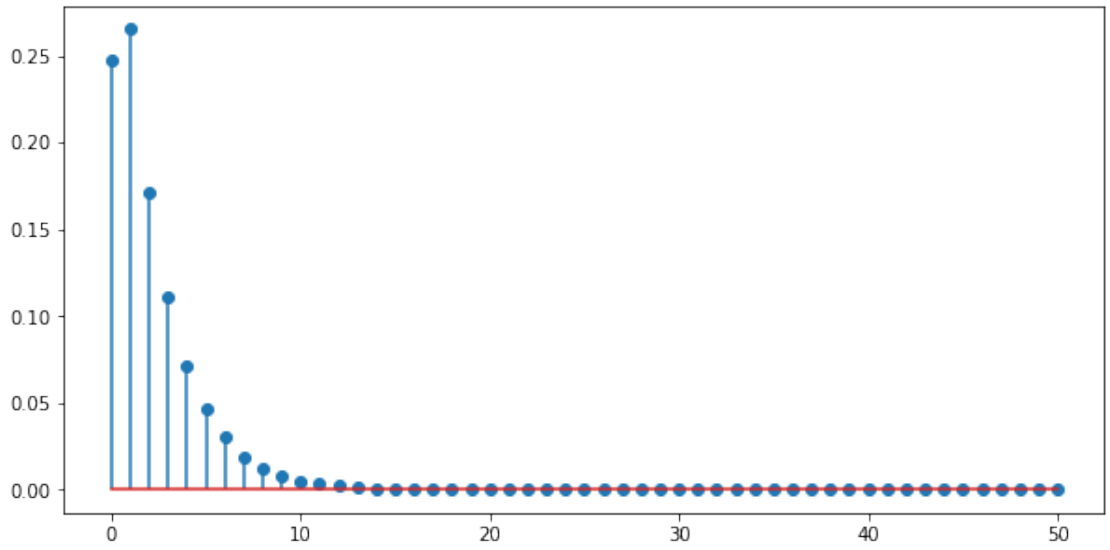

```
Mean Error is : 0.666
Sample Varience is : 2693914.424220429
```

Question 3:

```
from tkinter import Y
import matplotlib.pyplot as plt;
import numpy as np;
import random;
plt.rcParams['figure.figsize']= [10,5]
import csv

noofexp = 1000000

def findavg(k):
    s = 0
    for i in range(51):
        s = i*k[i]+s
    return(s/1000000)

def give():
    return(random.random())

def gen():
    x, k = [],[]
```

```
    z = 0
    y = [0 for x in range(51)]
    for i in range(noofexp):
        z = give()
        if z<=0.4:
            if len(x)>0:
                x.pop()
        w = give()
        if w<=0.3:
            x.append(5)
        y[len(x)] = y[len(x)]+1
        if (len(x)==0):
            k.append(i)
    z = findavg(y)
    for j in range(51):
        y[j]=y[j]/noofexp
    #return(y, k[len(k)-1])
    return(y,z)

y1, k = gen()
x1 = [x for x in range(51)]

print('Time average of number of packets in the memory:', k)
plt.stem(x1, y1, use_line_collection = True)
plt.show()
```

```
Time average of number of packets in the memory: 2.138006
```

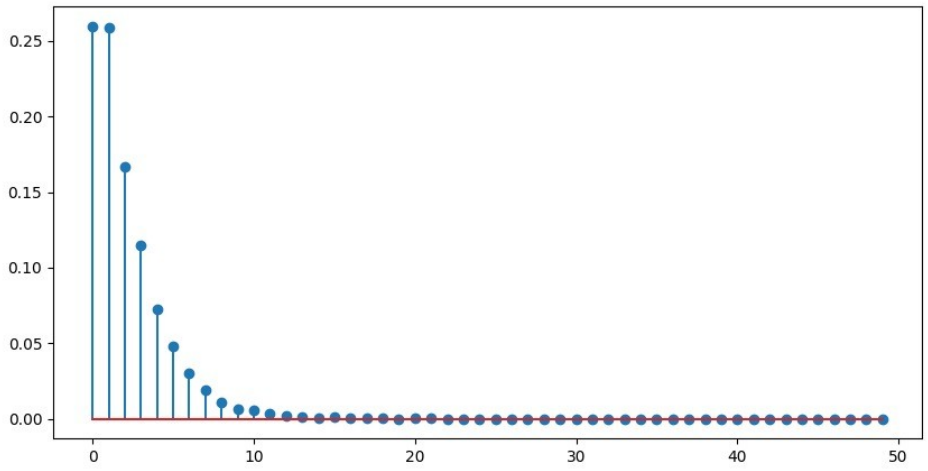


Question 4:

```
from tkinter import Y
import matplotlib.pyplot as plt;
```

```
import numpy as np;
import random;
plt.rcParams['figure.figsize']= [10,5]
import csv

noofexp = 1000000
mypt = [0 for i in range(100000)]
for i in range(100000):
    mypt[i] = random.randint(0,99999)

def findavg(k):
    s = 0
    for i in range(50):
        k[i] = k[i]*10000
        s = i*k[i]+s
    return(s/10000)

def give():
    return(random.random())

def gen(startpt, m):
    x = []
    k = [0 for i in range(10000)]
    j = 0
    if startpt>0:
        x = [0 for x in range(startpt)]
    y = [0 for x in range(50)]
    for i in range(noofexp):
        z = give()
        if z<=0.4:
            if len(x)>0:
                x.pop()
        w = give()
        if w<=0.3:
            x.append(5)
        if noofexp==1000000:
            y[len(x)] = y[len(x)]+1
        if noofexp==10000:
            if (i==m[j]):
                k[j] = len(x)
                j = j+1
    if noofexp==1000000:
        return(y, k)
    else:
        return(len(x))

y1, startpt = gen(0, mypt)
x1 = [x for x in range(50)]

k = [0 for i in range(50)]
```

```
noofexp=200
for i in range(10000):
    j = gen(startpt[i], 0)
    k[j] = k[j]+0.0001

plt.stem(x1, k, use_line_collection = True)
plt.show()

print('Sample Average', findavg(k))
```



After simulating 1000000 timesteps for 50 times, here are some observations I made:

1. The number of packets in the buffer does not depend on the timesteps remaining. Therefore, having n=0 is as good as being right back at the start. The final timestep where n=0 happened was always within the last 200 timesteps.Conducting the simulations 50 times, I found the average of this final timestep to be 999983.5 . The lowest of these numbers was 999861.

2. The final timestep where n=0 happened was always within the last 200 timesteps. Conducting the simulations 50 times, I found the average of this final timestep to be 999983.5 . The lowest of these numbers was 999881.

3. I compared this result with 1000 timesteps and 200 timesteps. The observations are recorded below.

Timesteps	T max for n=0	T min for n=0	T avg for n=0	N max avg
1000000	999	999861	999983.5	28.1
11000	999	844	958	13.3
200	199	122	177.5	9.7

4. The issue that comes with this is that there is not enough time for other intermediate values like n=10 or n=20 to come in. To compensate for that, we randomly assign a starting point according to the fraction of n that has occurred so far.