PDF-PMF

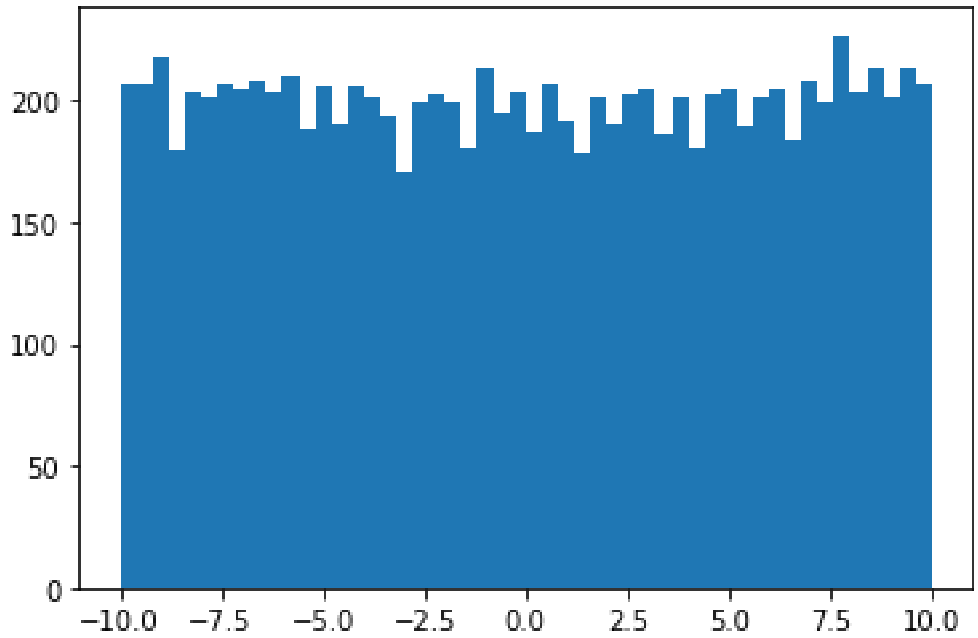
August 21, 2021

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
```

0.1 Uniform Distribution

[2]:
```
#Uniform distribution has equal probability across any given range you define
#there is pretty much equal chnace that value is occuring within that data␣
↪unlike normal distribution
#value occuring i.e concentration of value near mean

#I want uniformly distributed random data that ranges from -10 to +10 having␣
↪10000 sample data
values = np.random.uniform(-10.0,10,10000)
```

[6]:
```
#So a uniform distribution, you would see a flat line of the probability␣
↪distribution function
#because there is basically a constant probability every value, every range of␣
↪values,
#has an equal chance of appearing as any other value

plt.hist(values,50)
plt.show()
```

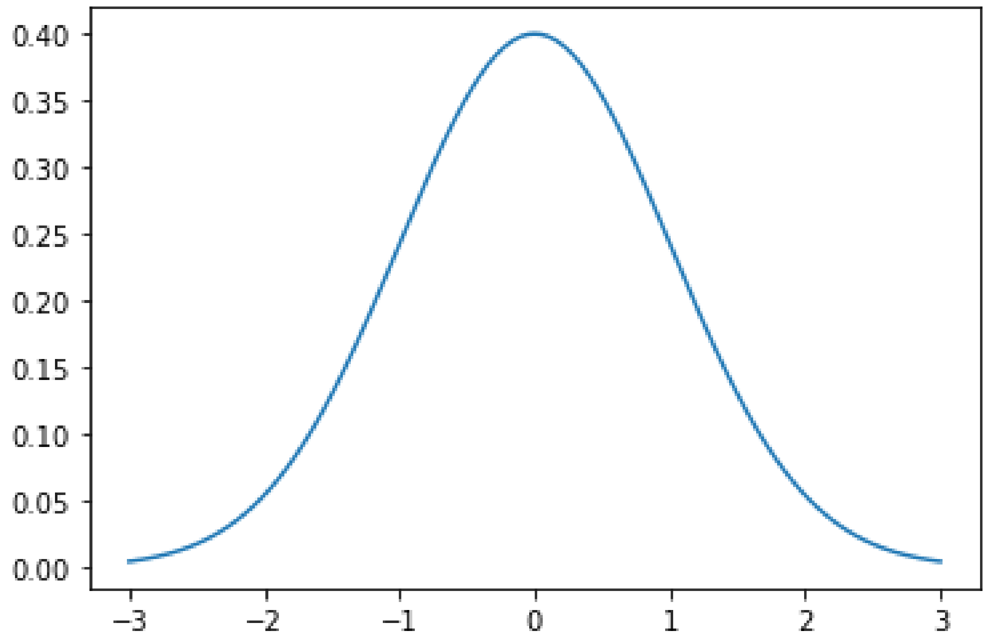


0.2 Normal or Gaussian Distribution

[7]:
```
from scipy.stats import norm

#We're creating a list of x values for plotting that range between - 3 and +3
#with a increment of .001 in between them

#So those are the x values on the graph,and then we're going to plot the x axis

x= np.arange(-3,3,0.0001)
```

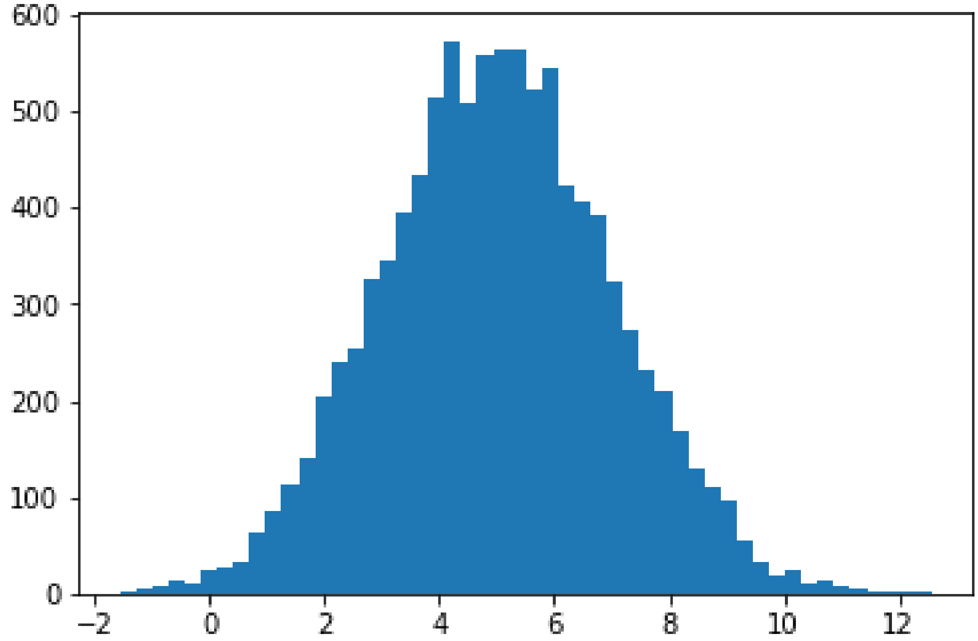
[8]:
```
#So those are the x values on the graph,and then we're going to plot the x axis

#the y axis is going to be the normal function norm.pdf, probability density␣
↪function
#for a normal distribution on those x values

#Where zero(0) will represent the mean, and these numbers (-1,-2,-3,+1,+2,+3)␣
↪are standard deviations.

#norm.pdf(x) -> for a normal distribution on those x values
plt.plot(x,norm.pdf(x))
```

[8]:
```
[<matplotlib.lines.Line2D at 0x7ff4bb472d30>]
```



[10]:
```
#Generate random number within normal distribution

#So This is the way to use a probability distribution function, in this case,␣
↪the normal distribution function,
#to generate a set of random data

mean_mu=5.0
standardDeviation_sigma=2.0

#Again if you use the NumPy package, it has a random.normal function,

#and the first parameter, mu, represents the mean that you want to centre the␣
↪data around.

#Sigma is the standard deviation of that data, which is basically the spread of␣
↪it.

#And then we specify the number of data points that we want using a normal␣
↪probability distribution function
x=np.random.normal(mean_mu,standardDeviation_sigma,10000)

#and it does look more or less like a normal distribution, but since there is a␣
↪random element,
#it's not gonna be a perfect curve
```

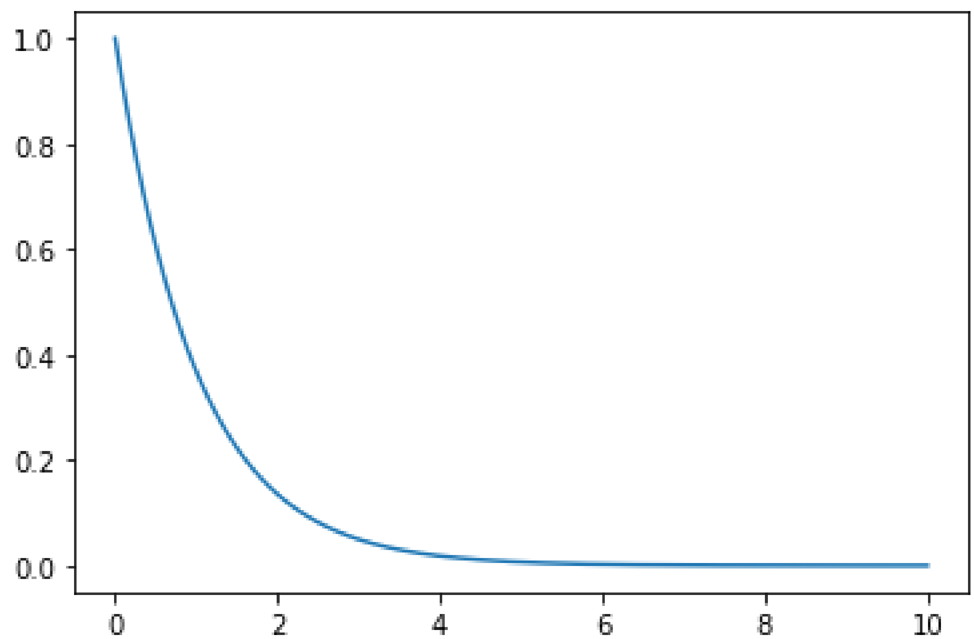
```
plt.hist(x,50)
plt.show()
```



0.3 Exponetial Probability Distribution Function

```
[11]: #Another distribution function you see pretty often is the exponential␣
       ↪probability distribution function,
      #where things fall off in an exponential manner
      from scipy.stats import expon

      #we just create our x values using the NumPy.arange function to create a bunch␣
       ↪of values
      #between zero and 10, with a step size of .001,

      x = np.arange(0,10,0.001)

      #so when you talk about an exponential falloff,you expect to see a curve like␣
       ↪this where it's very likely for
      #something to happen,near zero, but then as you get farther away from it, it␣
       ↪drops off very quickly

      #we also have an expon.pdf for an exponential probability distribution function
      #we plot those x values against the y axis,which is defined as the function␣
       ↪exponential pdf of x
      plt.plot(x, expon.pdf(x))
```

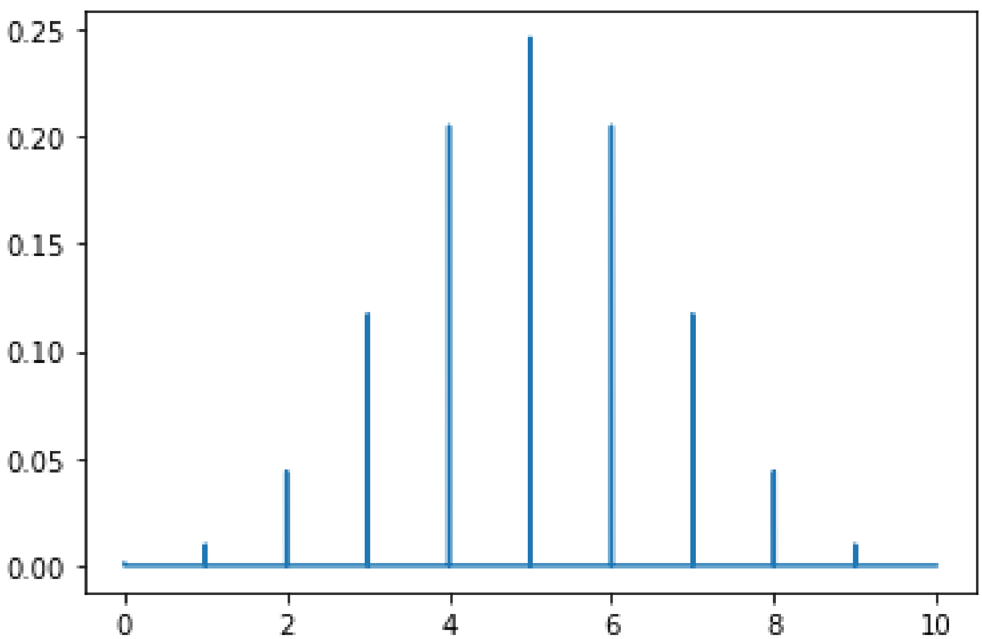
```
[11]: [<matplotlib.lines.Line2D at 0x7ff4b77b0040>]
```



0.4 Binomial Probaility Mass Function

[16]:
```
from scipy.stats import binom

#We can also visualise probability mass functions, it's called the binomial␣
↪probability mass function

#And again, a reminder, probability mass function deals with discrete data, and␣
↪in this case we are dealing with
#discrete data



#We're creating some discrete x values between zero and 10 at a spacing of .001
x = np.arange(0,10,0.001)

#And with the probability mass function,I can actually specify the shape of␣
↪that data
#using two shape parameters, n and p, in this case, it's 10 and 0.5

n, p = 10, 0.5

#and we're saying I want to plot a binomial probability mass function using␣
↪that data
plt.plot(x,binom.pmf(x,n,p))
```

[16]: [<matplotlib.lines.Line2D at 0x7ff49c85a040>]



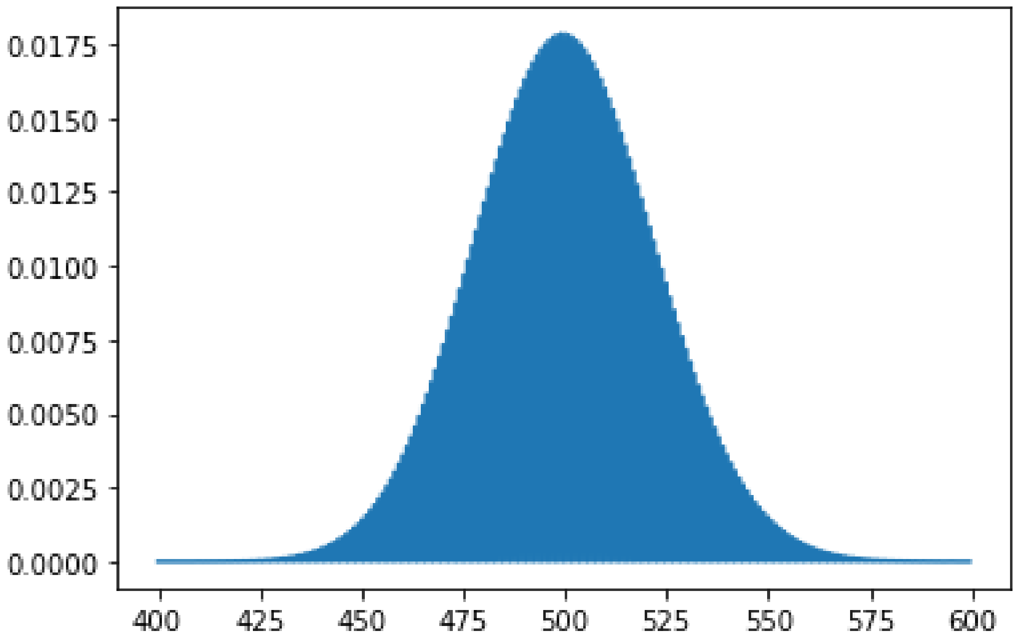
0.5 Possion Probability Mass Function

[18]:
```
#and this has a very specific application.Looks a lot like a normal␣
↪distribution,
#but it's a little bit different.

#The idea here is that if you have some information about the average number of␣
↪things that happen
#in some given time period This can give you a way to predict the odds of␣
↪getting some other value instead,
#on a given future day

#So as an example, let's say I have a website, and on average I get 500␣
↪visitors per day.
#I can use the Poisson probability mass function to estimate the probability of␣
↪seeing some other value
#on a specific day.So let's say I get an average of 500 visits per day,
#what's the odds of seeing 550 visitors on a given day,That's what a Poisson␣
↪probability mass function can give you

from scipy.stats import poisson
```

[19]:
```
#So, in this example, I'm saying my average is 500, mu.
mu = 500

#And I'm gonna set up some x values to look at between 400 and 600 with a␣
↪spacing of .5
#and I'm gonna plot that using the Poisson probability mass function.
x = np.arange(400, 600, 0.5)
```

[21]:
```
#And I can use that graph to look up the odds of getting any specific value␣
↪that's not 500,
#assuming a normal distribution

#So 550, it turns out, comes out to about .002 is the probability there, or .2%
plt.plot(x,poisson.pmf(x,mu))
```

[21]:
```
[<matplotlib.lines.Line2D at 0x7ff4bb663340>]
```


0.0175
0.0150
0.0125
0.0100
0.0075
0.0050
0.0025
0.0000
400
425
450
475
500
525
550
575
600

```
[ ]:
```